# Altova MapForce 2024 Professional Edition


## Manuel de l'utilisateur et de référence

# Altova MapForce 2024 Professional Edition
# Manuel de l'utilisateur et de référence



Published: 2024

# Table des matières

# 1 Introduction

Altova MapForce 2024 Professional 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 25 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 467 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 70 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. Pour plus de détails, voir Notions de base du projet 84.
- 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*). Pour les détails, voir Déployer les mappages vers FlowForce Server 867.
- 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*). Pour les détails, voir Paramètres de composant Base de données 263.
- 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 455.
- Prise en charge de .NET 8.0 pour la génération de code C# (*éditions Professional et Enterprise*). Pour les détails, voir Génération de code 70.
- 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*). Pour les détails, voir Bases de données 158.
- 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*). Pour plus d'information, voir `sleep` 698.
- Une prise en charge native a été ajoutée pour MySQL et MariaDB (*éditions Professional et Enterprise*). Pour plus d'informations concernant les bases de données prises en charge, voir bases de données 158.
- 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 57.

- 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 [127].
- Le système Aide [1102] a été réorganisé pour fournir une Aide en ligne par défaut, avec une option pour utiliser le manuel utilisateur PDF installé localement [1086] comme défaut alternatif.
- Il est désormais possible d'ajouter des commentaires « sticky-note-style » à un mappage. Pour plus d'information, voir Commentaires [39].
- Les paramètres ont été ajoutés pour définir les paramètres de réseau [1096].
- 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 [1101].
- 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* ). Pour plus de détails, voir Plug-in de MapForce pour Eclipse [914].
- 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 [498] basés sur base de données et de variables [414] 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*). Pour en savoir plus sur toutes les bases de données prises en charge, voir Bases de données [158].
- Prise en charge du Gestionnaire de schéma XML [141] 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* ). Pour des détails, voir Plug-in de MapForce pour Eclipse[914].
- Prise en charge de Visual Studio 2022 dans MapForce Plug-in pour Visual Studio et génération de code (*éditions Professional et Enterprise*). Pour plus d'informations, voir MapForce Plug-in pour Visual Studio[911] et génération de code.
- Prise en charge de .NET 6.0 dans génération de code (*éditions Professional et Enterprise*). Pour les détails, voir génération de code.
- 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*). Pour en savoir plus sur toutes les bases de données prises en charge, voir Bases de données[158].
- Il est désormais possible de prévisualiser des images dans la fenêtre **Projet**(*éditions Professional et Enterprise*). Pour plus d'informations, voir Notions de base de projet[84].
- 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* ). Pour des détails, voir Plug-in de MapForce pour Eclipse[914].
- Copy-all connections[60] 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*). Pour plus d'informations, voir Volets de sortie StyleVision[872].
- Prise en charge de JSON Schema dans les variables[413] et paramètres UDF[497] (*Enterprise Edition* ).
- Prise en charge pour bases de données NoSQL : MongoDB et CouchDB (*Enterprise Edition* ). Pour en savoir plus sur toutes les bases de données prises en charge, voir Bases de données[158].
- 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 70. 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 547.
- Lorsque vous générez le code C#, vous pouvez choisir .NET Core 3.1 et .NET 5.0 en tant qu'infrastructures cibles depuis les options de génération de code (ceci s'ajoute à la prise en charge existante pour les projets .NET Framework). Pour les détails, voir Générer le code C#.
- Mises à jour internes et optimisations.

### Version 2021

- Un mappage MapForce peut lire des données BLOB (binary large object) depuis des fichiers binaires et écrire des fichiers binaires sur le disque. Cela permet, par exemple de lire des champs BLOB depuis une base de données et de les enregistrer en tant que fichiers d'image sur le disque, ou de lire des fichiers binaires comme des PDF depuis le disque et de les enregistrer en tant que des champs `xs:base64Binary` dans le cadre d'un fichier XML. Voir Fichiers binaires 665 pour plus d'informations.
- De nouvelles versions de base de données sont prises en charge : MariaDB 10.4, 10.5
- De nouvelles versions d'Eclipse sont prises en charge : 2019.09, 2019.12, 2020.03, 2020.06
- Lorsque vous réunissez plusieurs tables ou modes de base de données en utilisant des composants « join » SQL dans un mappage, vous pouvez définir le mode « join » soit en tant que LEFT OUTER JOIN soit INNER JOIN, voir Modifier le Mode Join 350.
- Mises à jour internes et optimisations.

## 1.1.5 Version 2020

### Version 2020 Release 2

- Une nouvelle fenêtre Gérer la fenêtre des bibliothèques 28 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 496.
- 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 474. 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 ou XQuery, vous pouvez générer un code XSLT ou XQuery qui référence les fichiers de bibliothèque importés en utilisant un chemin relatif. La nouvelle option est disponible dans le dialogue Paramètres de mappage 79.
- L'API MapForce 1107 a été amélioré avec de nouveaux membres qui vous permettent de gérer des bibliothèques importées de manière programmatique (par exemple les ajouter ou les supprimer).
- Le code généré pour des bibliothèques de wrapper de schéma XML propose maintenant un plus grand contrôle sur les espaces de noms d'élément et des préfixes. De nouvelles méthodes sont disponibles pour déclarer ou contourner des espaces de noms pour un élément, ou pour apposer un élément avec un espace de noms préfixé. Voir Exemple : Purchase Order 990.

- De nouvelles versions de base de données sont prises en charge : PostgreSQL 12.1 et Informix 14.10.
- Mises à jour et optimisations internes

## Version 2020

- Prise en charge pour Visual Studio 2019 dans MapForce Plug-in pour Visual Studio 911 et génération de code.
- Prise en charge d'Eclipse 4.9 - 4.12, voir MapForce Plug-in pour Eclipse. 914
- Si un package Oracle contient des procédures ou des fonctions publiques stockées, celles-ci sont aussi disponibles dans le mappage, voir Ajouter des procédures stockées dans le mappage 324.
- Vous pouvez configurer un composant de base de données de manière à ce que les noms d'objet de base de données soient traités en tant que relatifs par rapport au schéma par défaut, mais non liés à un schéma particulier. Cela vous permet d'économiser du temps si vous avez besoin de passer à une base de données à l'avenir, voir Changer de base de données et de schémas 250.
- Vous pouvez déployer des Ressources globales créées dans MapForce vers FlowForce Server, voir Déployer des Ressources globales dans FlowForce Server 900.
- 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 451.
- Mises à jour et optimisations internes

# 1.2 MapForce, c'est quoi?

**Site web d'Altova :** Outil de mappage de données

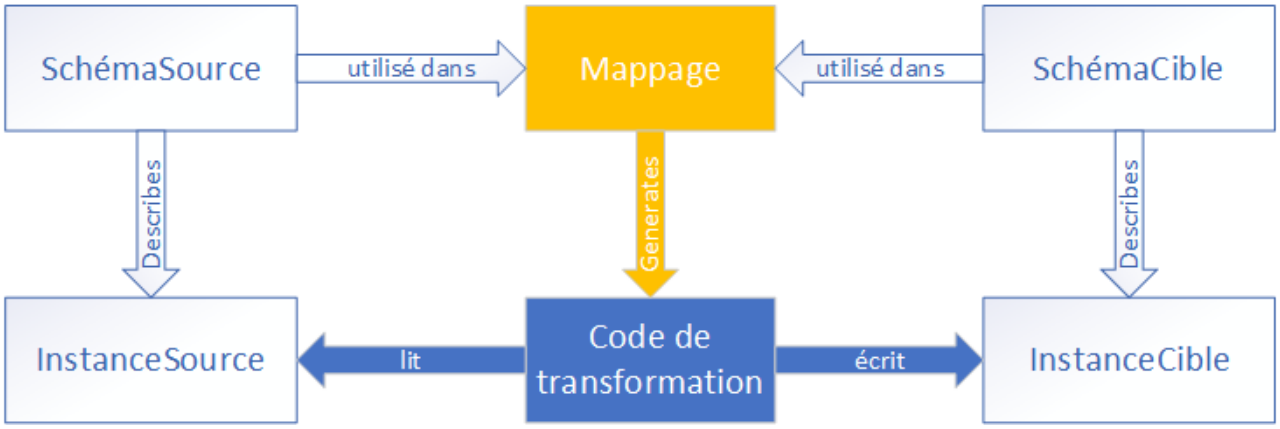
MapForce est un outil graphique puissant pour la conversion et l'intégration any-to-any. Voir Mappage : Sources et Cibles [21] pour une liste complète de formats de données disponibles. Un mappage typique consiste en une ou plusieurs sources de données et une ou plusieurs cibles de données [37]. Le mappage peut aussi inclure un ou plusieurs composants de transformation [38] 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 [23] et Tutoriels [88].

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 [22] 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 [89].




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 70.

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<br>• XSLT 2.0<br>• XSLT 3.0 | • XSLT 1.0<br>• XSLT 2.0<br>• XSLT 3.0<br>• BUILT-IN<br>• XQuery<br>• Java<br>• C#<br>• 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 1361.

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 30. 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[1361]). 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[855].

## 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[21].

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[89].
- Fusionner des sources de données multiples dans une cible. Pour plus d'informations, voir Tutoriel 2[100].
- 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[105].
- Mapper les sources multiples vers les cibles multiples. Voir le Tutoriel 4[113].

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[35] et Aperçu de l'interface d'utilisateur[25].

*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. Pour plus de détails, voir Projets[82]..

## 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*).

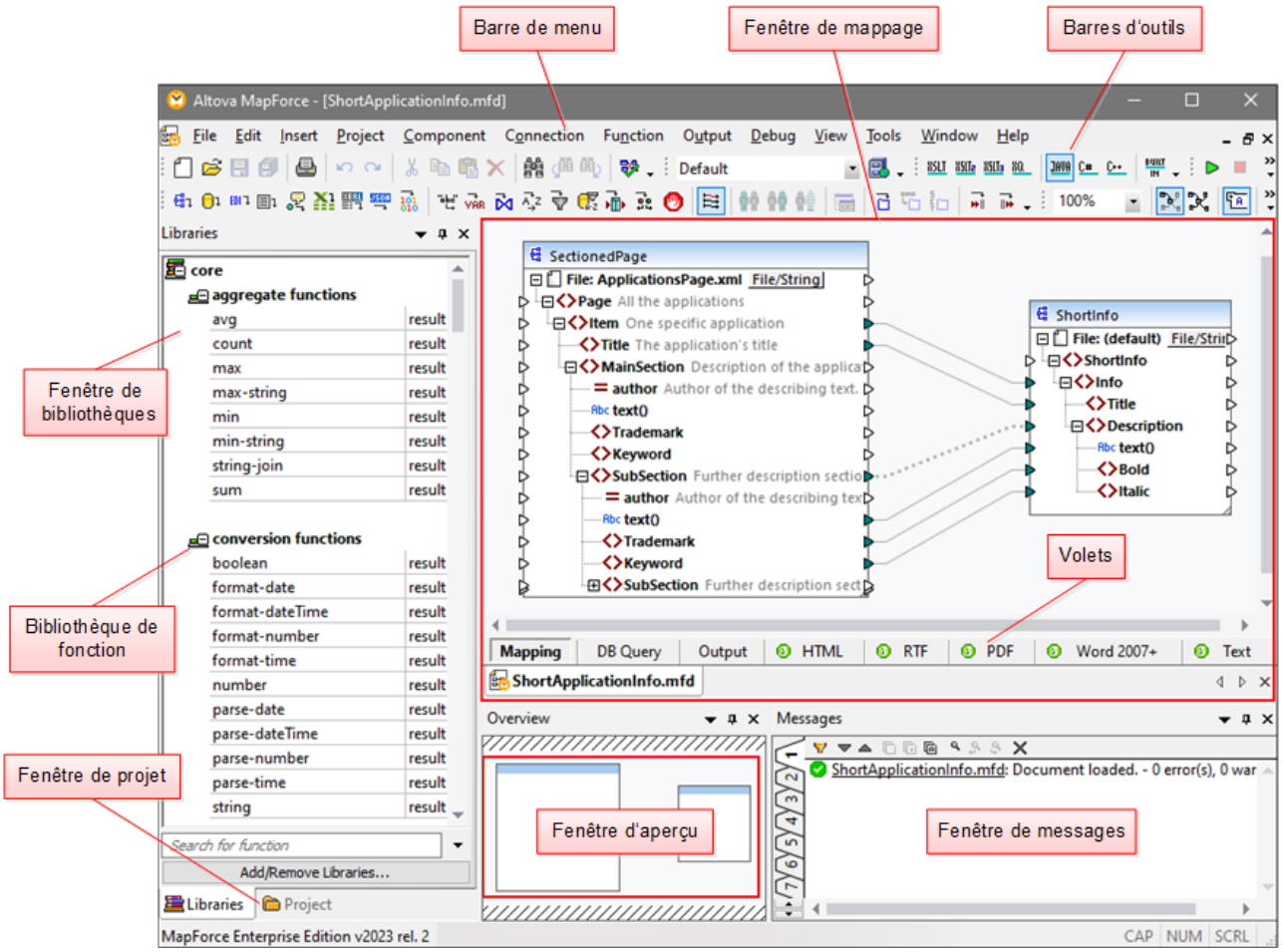
*XMLSpy*
Si XMLSpy est installé sur le même appareil, vous pouvez ouvrir et éditer tout type de fichier pris en charge en ouvrant XMLSpy directement depuis les contextes spécifiques de MapForce. Par exemple, la commande de menu **Composant | Éditer une définition de schéma dans XMLSpy** est disponible 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 855. Pour en savoir plus sur l'utilisation de MapForce en tant que plug-in, voir Plug-in pour Visual Studio 911 et Plug-in pour Eclipse 914.

## 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.




Dans la capture d'écran ci-dessus, les volets avec les logos verts sont des volets StyleVision. Pour les détails, voir Volets de sortie StyleVision 34.

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


- Barres 26
- Fenêtres 26
- Fenêtre de messages 30

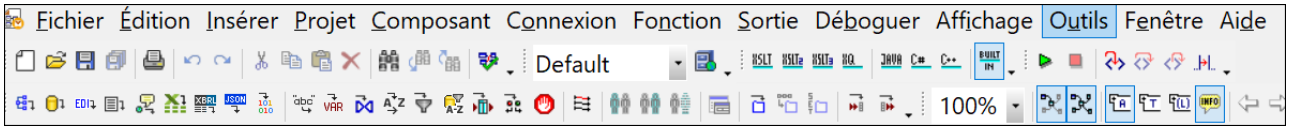
- Volets 31

## 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

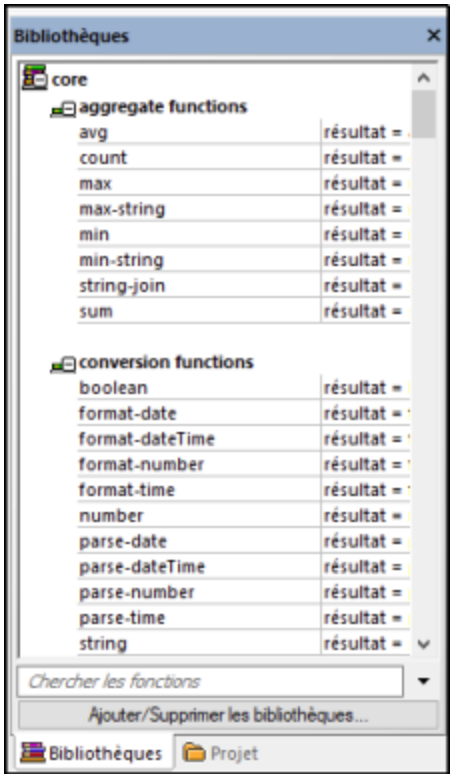
La barre de statut d'application apparaît en bas de la fenêtre d'application et montre l'information au niveau de l'application. Les info-bulles sont affichées quand vous déplacez la souris au-dessus d'un bouton de barre d'outils. Si vous utilisez la version 64-bit de MapForce, le nom d'application apparaît 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

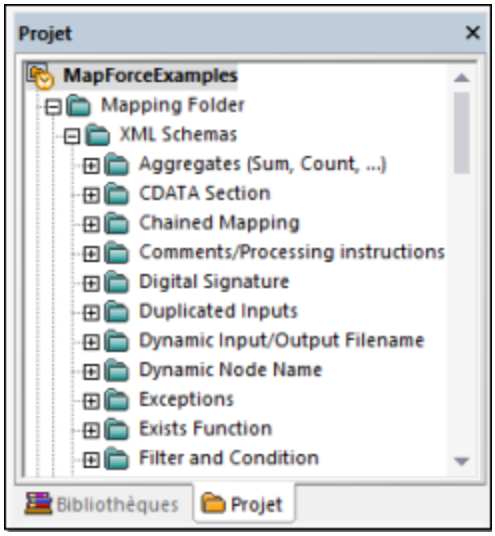
La fenêtre **Bibliothèques** recense les fonctions intégrées MapForce, qui sont organisées en bibliothèques. La liste des fonctions disponibles change sur la base du langage de transformation que vous avez choisi soit depuis le menu **Sortie**, soit depuis la barre d'outils **Sélection de langage**. Pour plus d'information, voir Langages de transformation 22. 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 467.

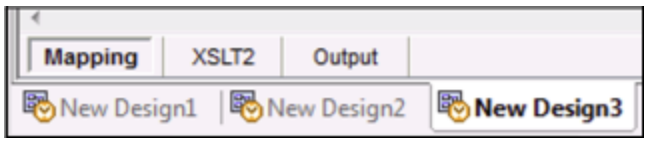
## Fenêtre Projet (Éditions Enterprise et Professional)

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**.Pour plus d'information, voir Projets 82.




## 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*).



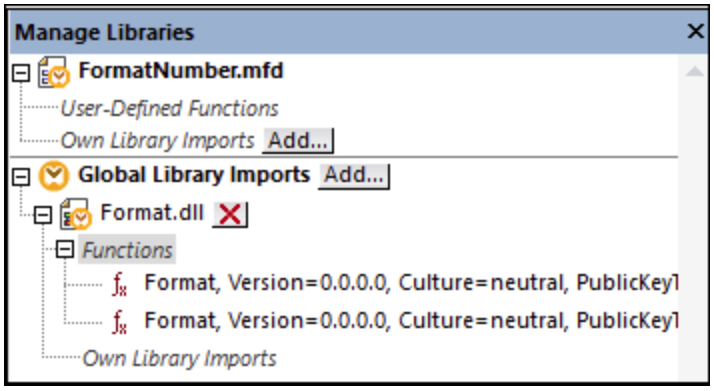

Vous pouvez ouvrir les options de gestion de la Fenêtre en utilisant la commande de menu **Fenêtre | Fenêtres**. Le dialogue **Fenêtres** vous permet de réaliser différentes actions y compris activer, enregistrer, fermer ou minimiser les fenêtres de mappage ouvertes. Pour choisir plusieurs fenêtres dans 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 (y compris les fichiers Java .class compilés et les fichiers .NET DLL assembly) 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.



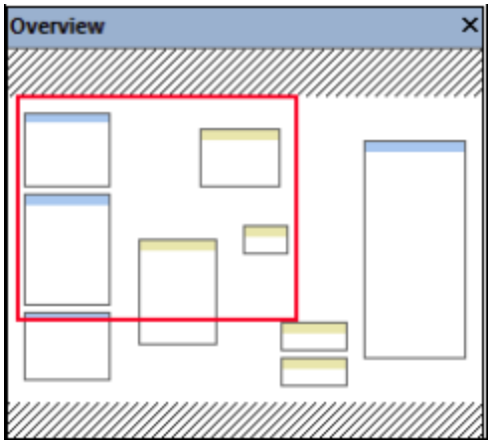

Vous pouvez choisir de consulter des UDF et des bibliothèques uniquement pour le document de mappage (fichier .mfd) qui est activé actuellement, ou pour tous les mappages ouverts. Pour consulter des fonctions et des bibliothèques importées pour tous les documents de mappages ouverts actuellement, cliquer 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 471.

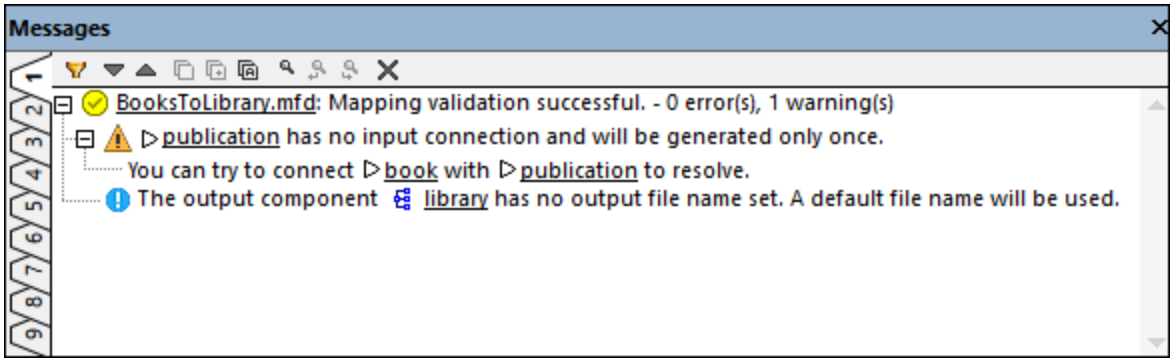
## Fenêtre de vue d'ensemble

La fenêtre **Aperçu** vous donne une vue d'ensemble du volet de mappage 31. 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 [68] 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

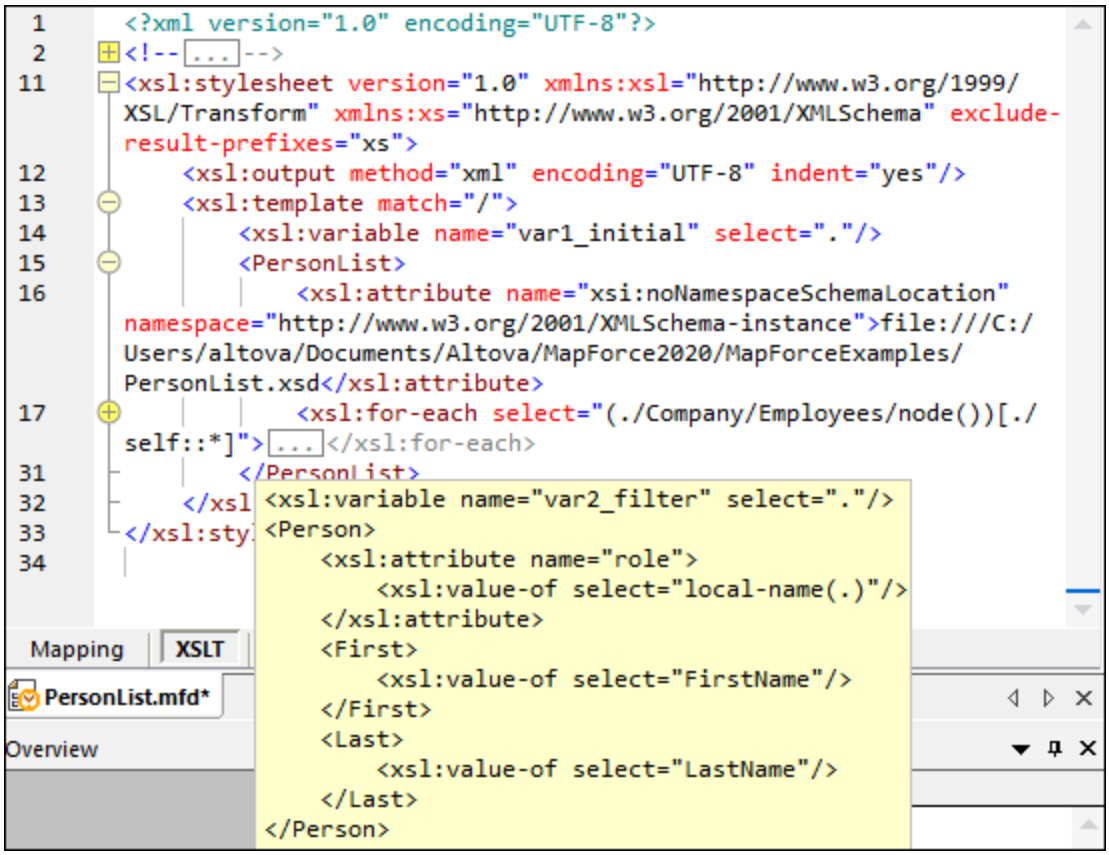
Le volet **Mappage** est la surface de travail dans laquelle vous configurez les mappages 68. Vous pouvez ajouter des composants de mappage (comme des fichiers, des schémas, des constantes, des variables, etc.) dans la zone de mappage à partir du menu **Insérer**. Pour plus d'information, voir Ajouter des composants au

mappage[41]. 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[468].

## 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[22], 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.

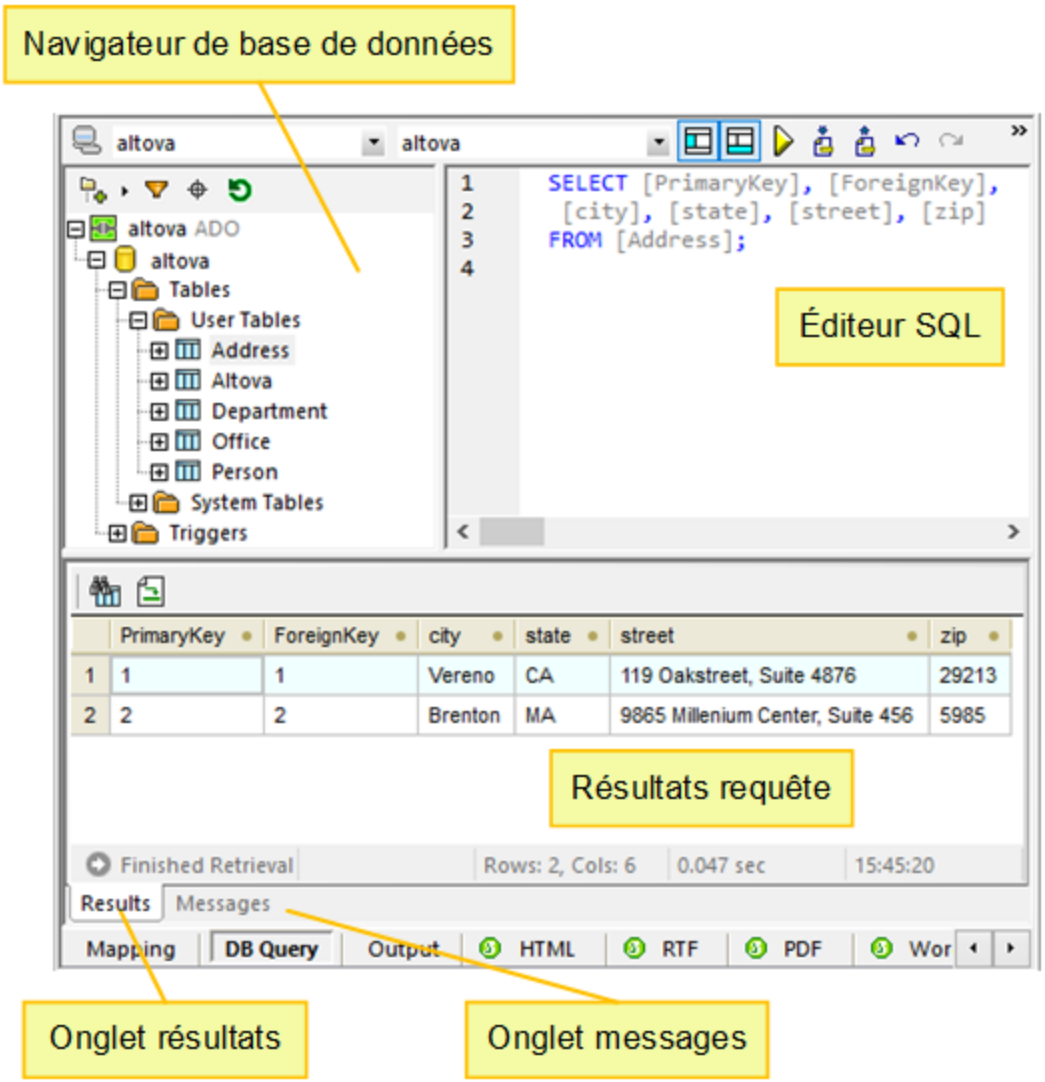
## Volet XQuery (Éditions Enterprise et Professional)

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. Pour plus d'informations, voir Parcourir et interroger des bases de données 300.




## 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).

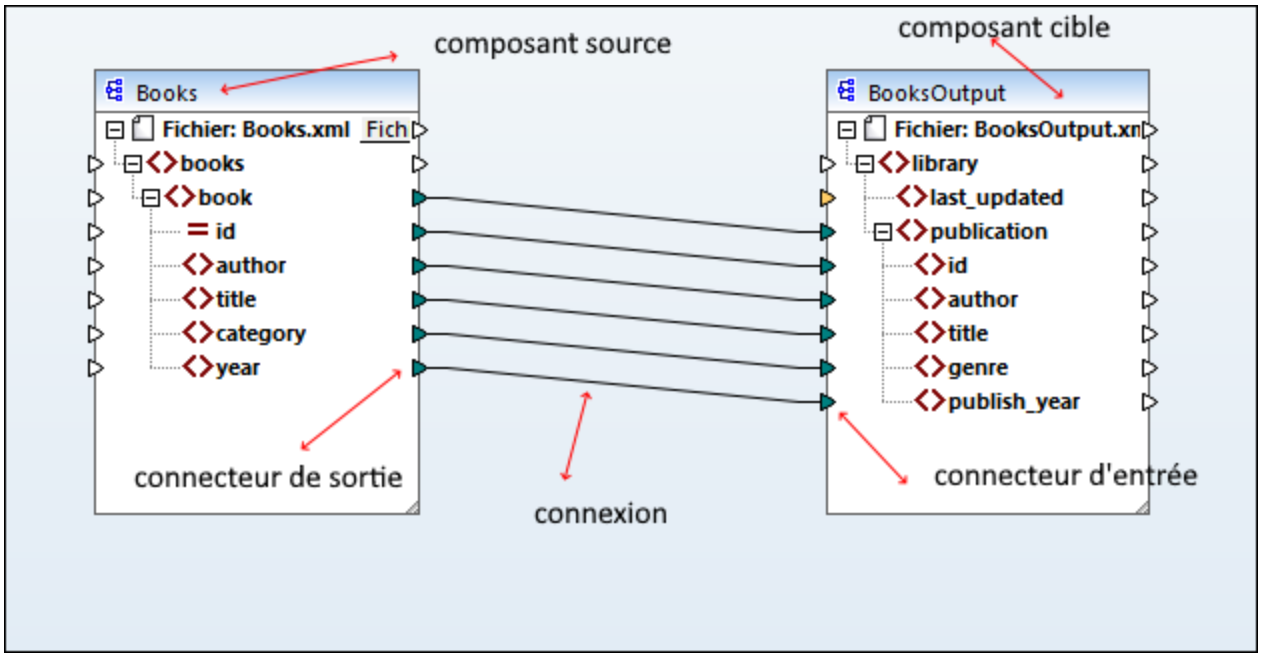
## Volets de sortie de StyleVision (Éditions Enterprise and Professional)

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**. Puis sélectionnez **Mappage** et cliquez sur **OK**. La prochaine étape consistera à ajouter des composants 41 au mappage et à créer les connexions 51.

## 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[122] et de transformation[397]

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[105]. 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[37].

*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

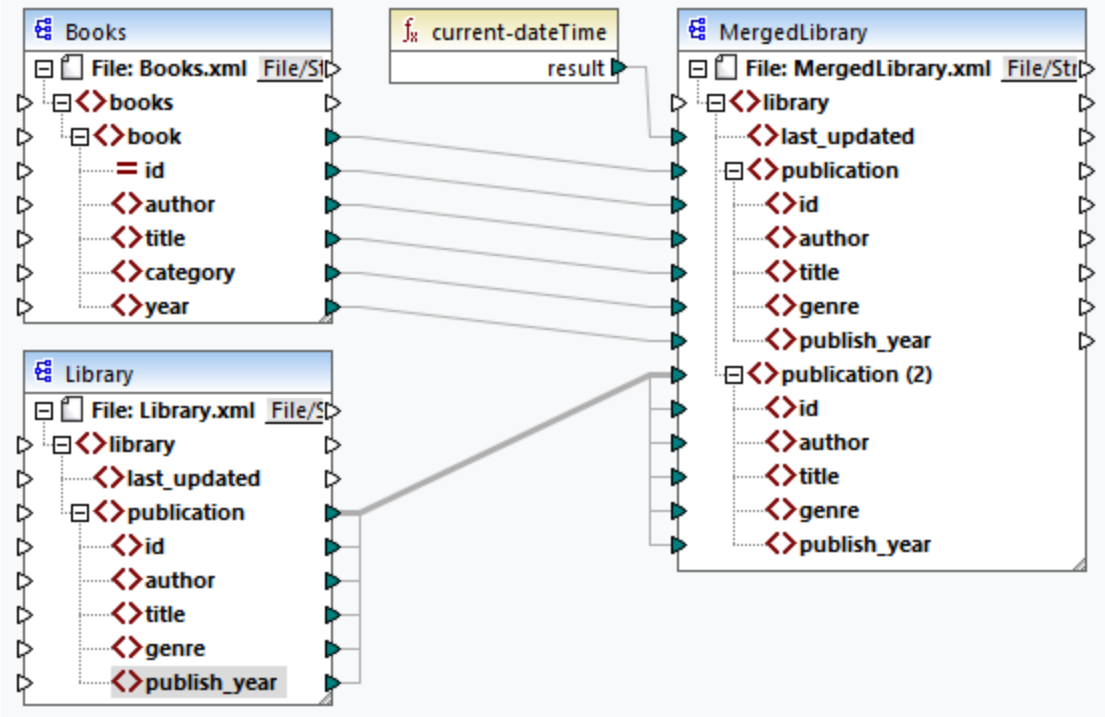
Cette section décrit les tâches et concepts les plus communs de MapForce. La section est organisée en sous-sections comme suit :

- Composants[37]
- Connexions[51]
- Procédures générales et fonctions[68]
- Projets[82]

# 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 35. 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 122. 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[467], stocker un résultat de mappage intermédiaire[413] pour un traitement ultérieur, remplacer une valeur par une autre valeur[451], trier[438], grouper[597], joindre[426], et filtrer[444] vos données. Vous pouvez également ajouter une exception[463], qui interrompt le processus de mappage et affiche une erreur quand une condition définie par un filtre apparaît. 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*) |

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

## Commentaires

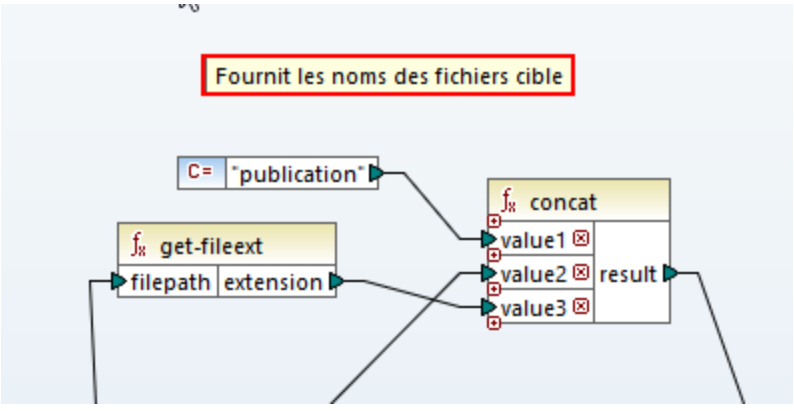
MapForce permet d'ajouter des commentaires en composants standalone et comme notes sous les composants existants. Commenter Texte est disponible non seulement dans un mappage, mais est aussi ajouté à la documentation de mappage générée 823. Dans la documentation de mappage générée, les commentaires de composant sont ajoutés directement sous le composant correspondant, et les composants de commentaire font partie de la section *Composants restants*.

*Composants commentaires*

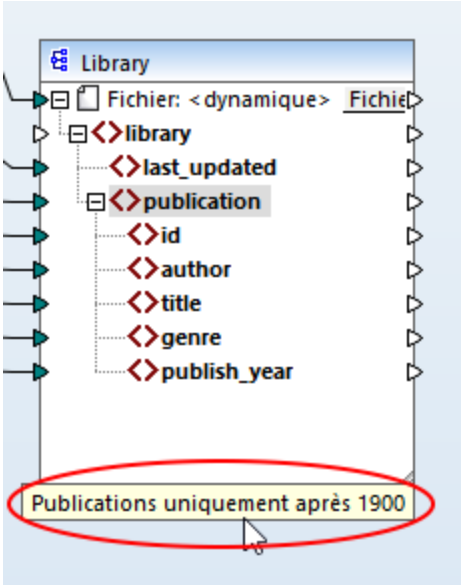
Les composants commentaires sont des boîtes autoportantes et ne peuvent pas être connectés avec tout autre composant. Pour ajouter un composant commentaires, vous pouvez sélectionner une des options suivantes :

- 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 [1086].

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 35, 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*)

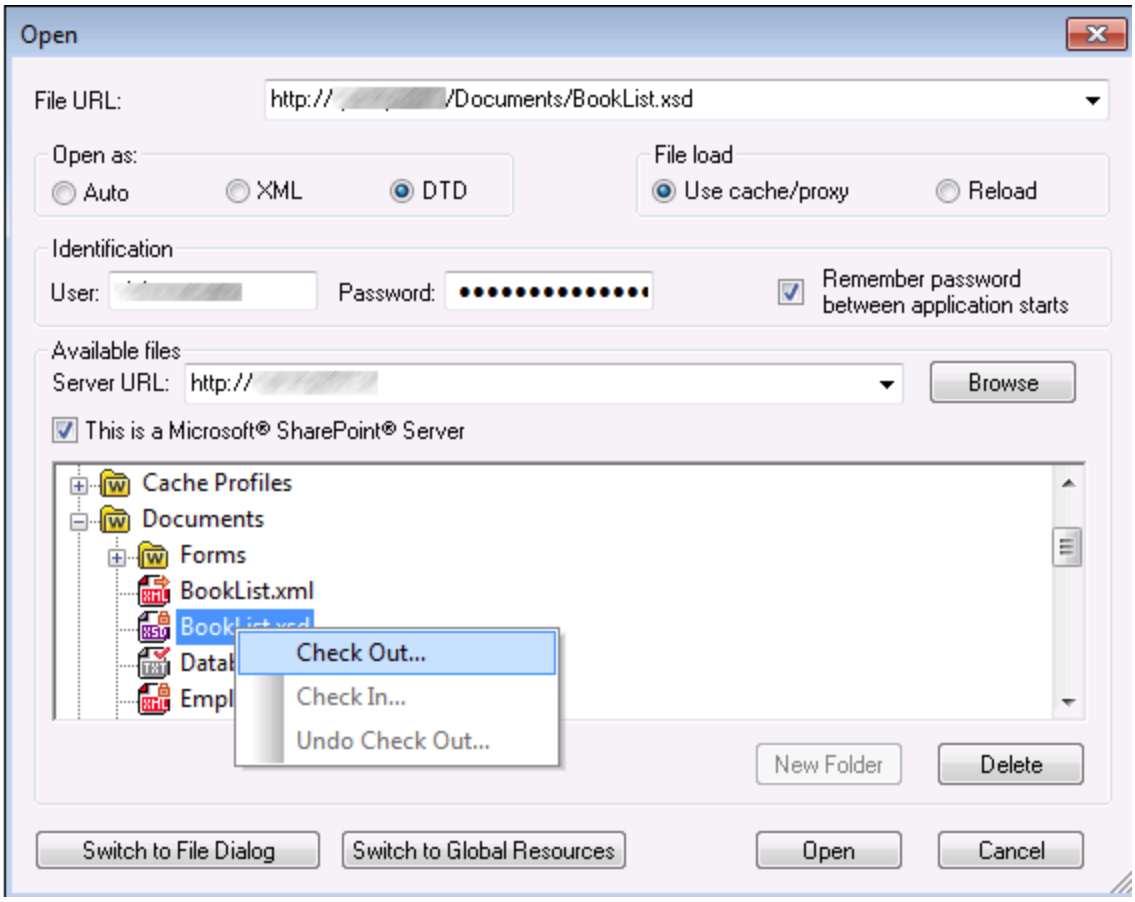

Chaque type de composant a un but et une fonction spécifiques. Pour obtenir un aperçu des composants, voir Composants 37. Si vous souhaitez en savoir plus sur les structures de données qui peuvent être utilisées comme sources et cibles de, voir les composants de structure 122. Pour des informations concernant les composants intégrés de MapForce utilisés pour stocker des données temporairement ou les transformer, voir Composants de transformation 397.

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 35. 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 ll'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**.

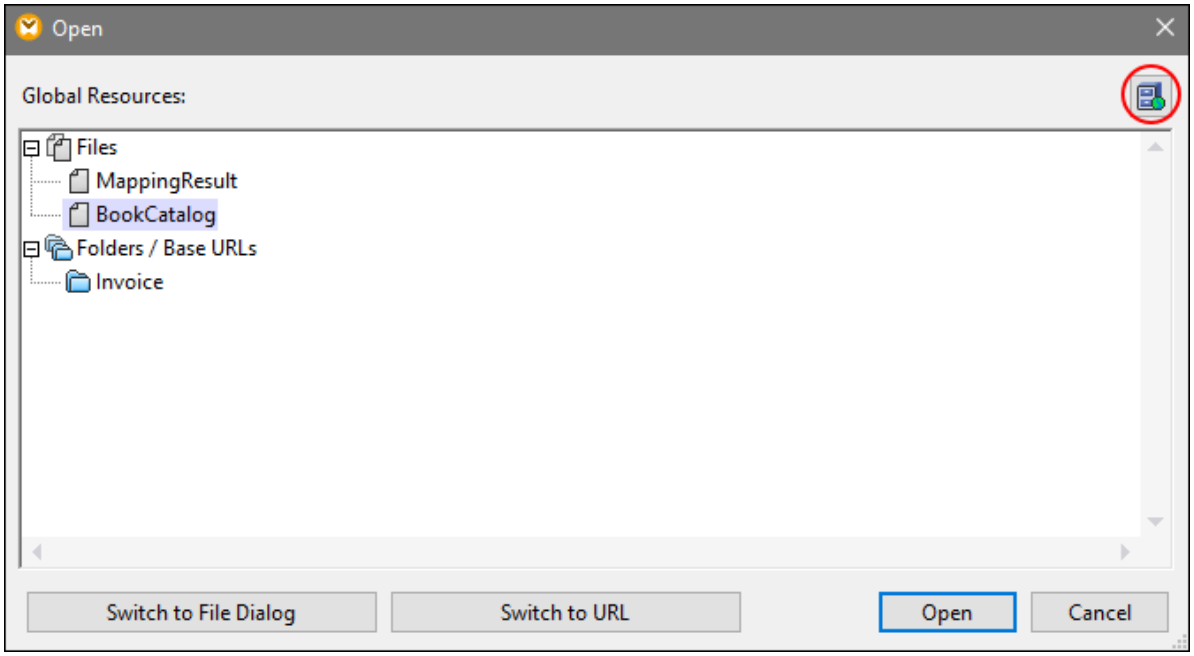
- *Passer au dialogue Fichier :* En cliquant sur le bouton, vous serez redirigé vers le dialogue dans lequl vous pouvez sélectionner un fichier local.

- *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 881. 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**).

Si vous avez créé une base de données comme ressource globale et que vous voulez l'ajouter à votre mappage, suivez les étapes décrites dans Ressources globales 192.

## 2.1.2 Les bases de composant

Cette rubrique explique comment définir, chercher et manipuler les composants de structure[37]. 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**.

Veuillez noter que les options disponibles dépendent du type de composant. La liste des paramètres pour chaque type de composant est donnée ci-dessous :

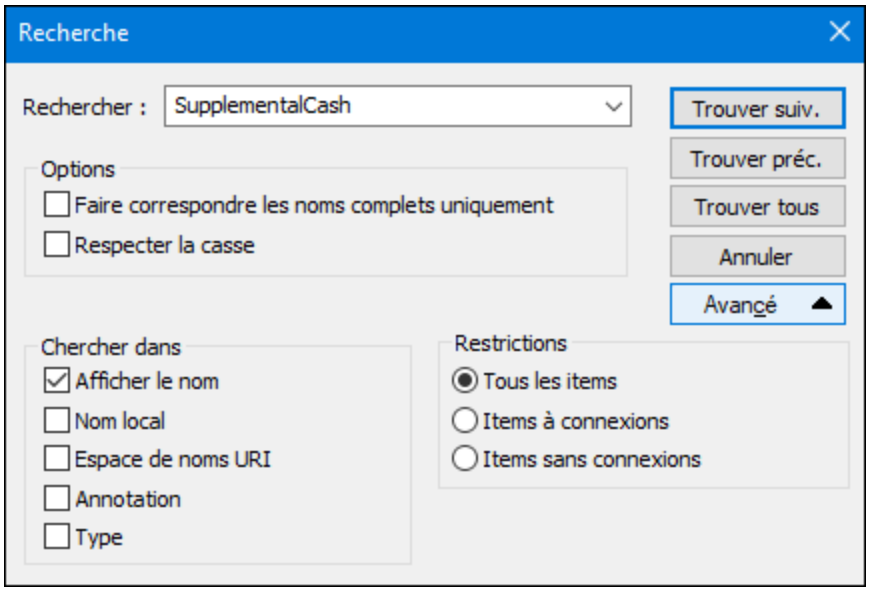
- Paramètres de composant XML[124]
- Paramètres de composant de base de données[256]
- Paramètres de composant CSV[382]
- Paramètres de composant du Champ à longueur fixe[390]

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[787]. Ce bouton vous permet également de définir les options pour les strings de parsage et les données de sérialisation vers les strings[794].

### Chercher un composant

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

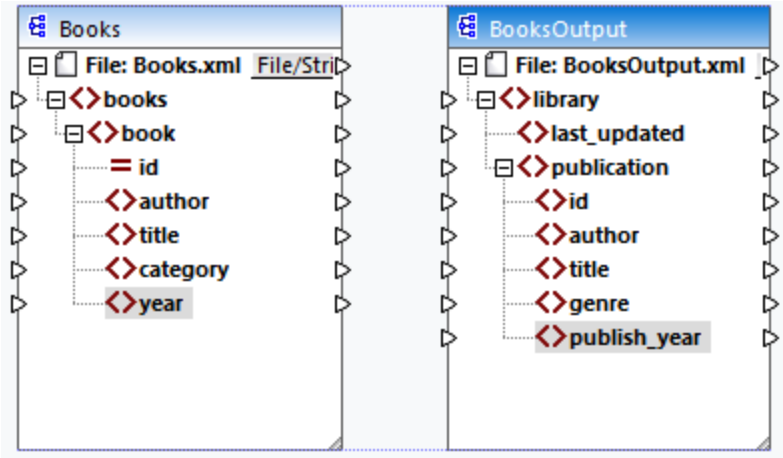
1. Cliquez sur le composant que vous souhaitez parcourir et appuyer sur les touches **CTRL+F**.
2. Saisissez un terme de recherche et cliquez sur **Trouver suivant/précédent/tout** (*voir la capture d'écran ci-dessous*).




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**.

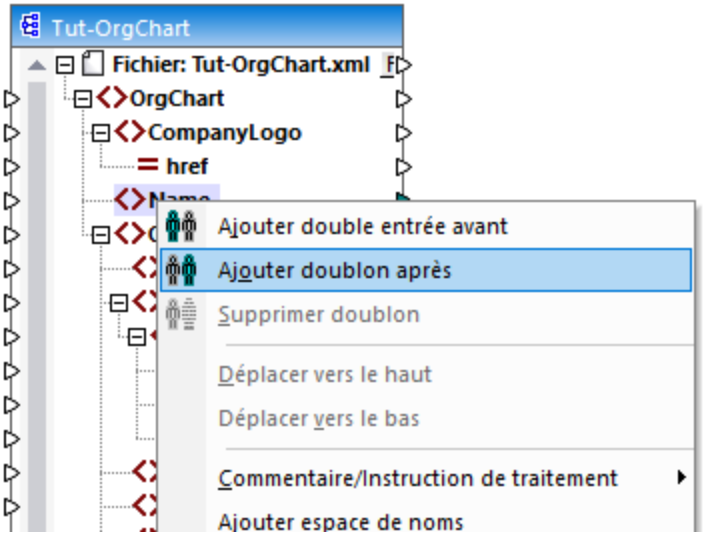
## Dupliquer l'entrée

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[100].

*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 44.

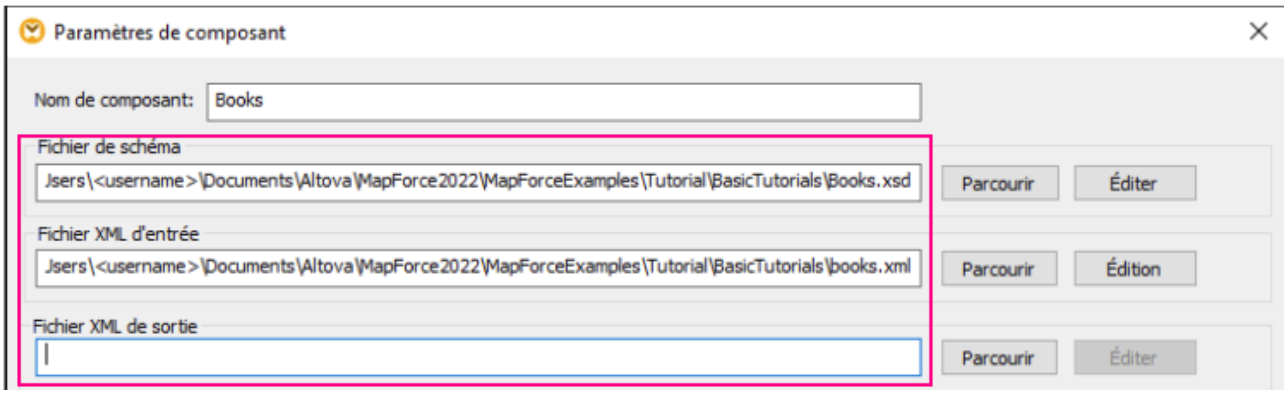
#### Exemple : Composant XML

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 [35] et ajouter un composant de structure [41] : 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 :

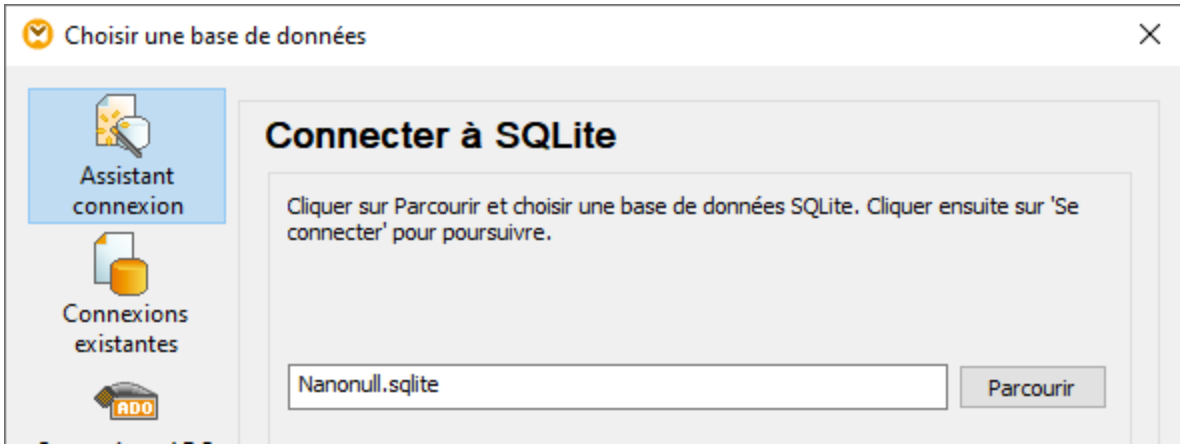
- Fichiers de structure utilisés par des paramètres complexes d'entrée ou de sortie des fonctions définies par l'utilisateur et des variables de type complexe ;
- Fichiers plats d'entrée ou de sortie (*éditions Professional et Enterprise*) ;
- Fichiers de schéma référencés par les composants de base 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**. Pour plus d'information sur la connexion à une source de base de données, voir Démarrer l'assistant de connexion de la base de données 163.




**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 79. Pour en savoir plus, voir À propos des chemins dans le code généré 49.

## 2.1.3.2 Les chemins dans des environnement d'exécution différents

Si vous générez du code depuis les mappages, compilez les mappages vers les fichiers d'exécution de MapForce Server (**.mfx**), ou déployez les mappages vers FlowForce Server, les fichiers générés sont exécutés dans l'environnement cible de votre choix : par exemple, RaptorXML Server, MapForce Server, ou une application C#. 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. |

| Langage cible | Chemin de base |
|---|---|
| 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[79].

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[471].

* *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[79].
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

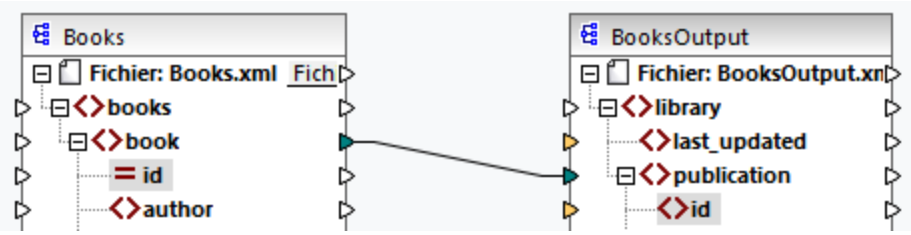
Pour créer une connexion entre deux lignes, appuyez et restez sur le connecteur de sortie[36] d'un nœud source et glissez-le vers un nœud de destination. Un connecteur d'entrée accepte *une seule* connexion. Si vous tentez d'ajouter une deuxième connexion à la même entrée, MapForce vous demandera de remplacer la connexion par une nouvelle ou de dupliquer l'item d'entrée[45]. 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*

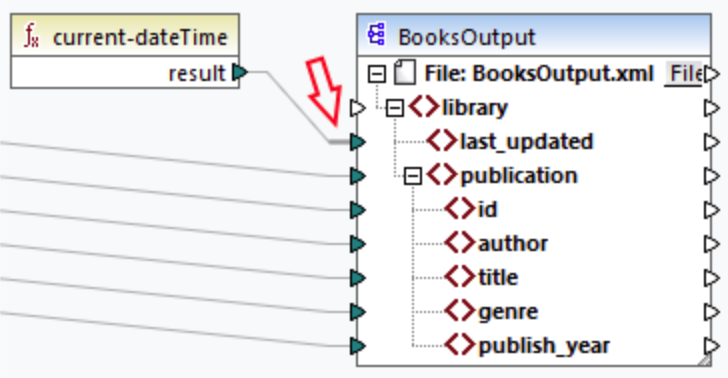
Lorsque vous créez des connexions entre les nœuds source et cible manuellement, MapForce analyse automatiquement les résultats possibles de mappage. Si vous vous connectez aux deux nœuds enfant sans les connecter à leurs nœuds parent, vous verrez un message de notification qui suggère la connexion du 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

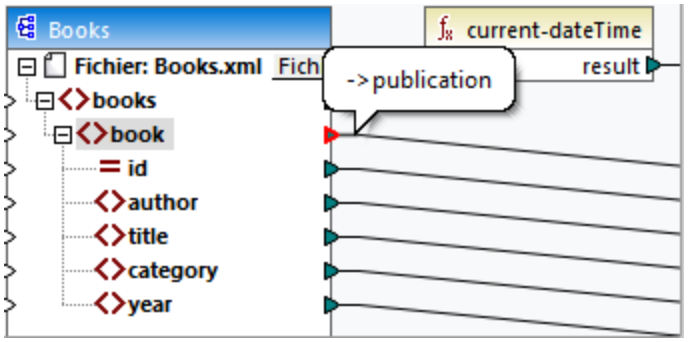
Pour déplacer une connexion vers un nœud différent, appuyez et tenez appuyée la section épaisse à la fin de la connexion (*voir la capture d'écran ci-dessous*) et glissez-la vers la destination sélectionnée.




## 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 61.

## 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.

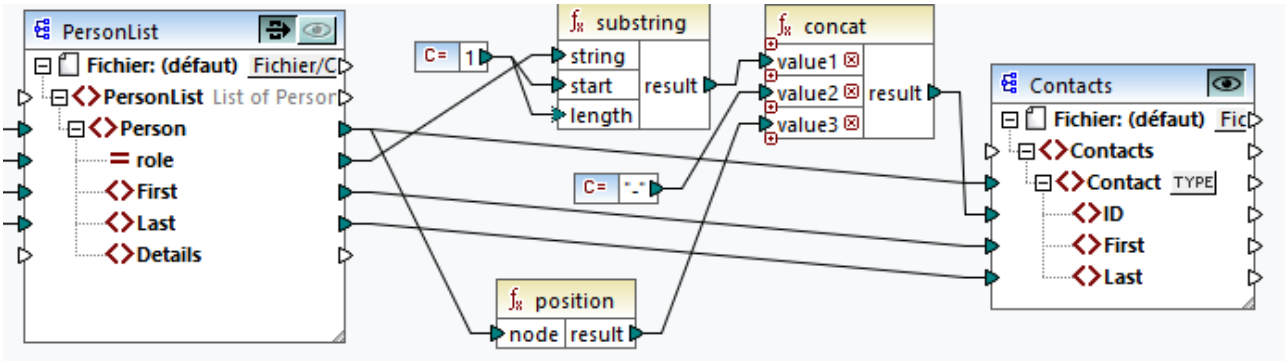
*Connexions directes et indirectes*

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.

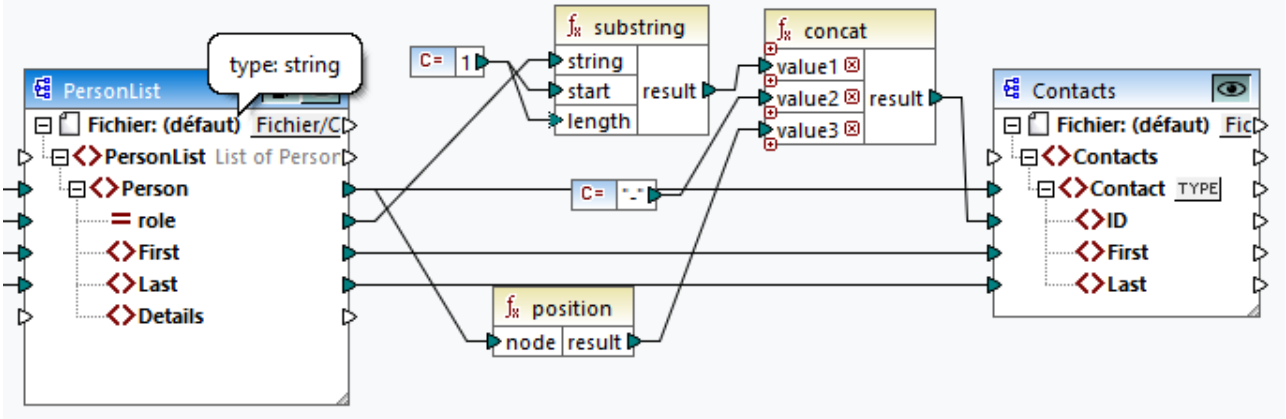
Pour comprendre comment ces deux options fonctionnent, voir l'exemple ci-dessous.

*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[38] (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[37] (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 :



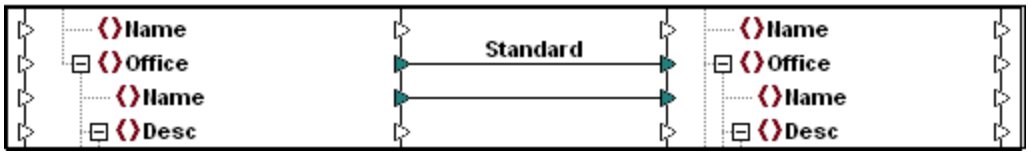
## 2.2.1 Types de connexion

Les types de connexion suivants sont disponibles dans MapForce :

- Connexions orientées vers la cible[55] (Standard) ;
- Connexions orientées vers la source[55] (Contenu mixte) ;
- Connexions d'enfants correspondants[57]
- Connexions copy-all[60] (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 55 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 21.

*Instance XML source*

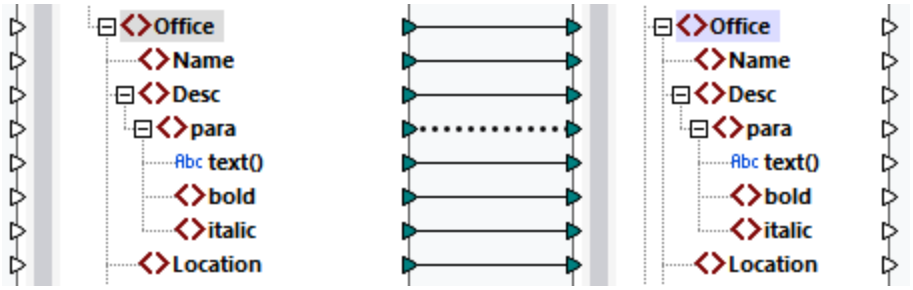
Un snippet du **Tut-OrgChart.xml** est affiché ci-dessous. Dans cet exemple, nous nous concentrerons sur l'élément du contenu mixte <para> avec ses nœuds enfant <bold> et <italic>. L'élément <para> contient aussi une instruction de traitement (<?sort alpha-ascending?>) ainsi qu'un commentaire (<!--Company 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 57 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 57). Supprimez la case à cocher **Créer des connexions copy-all** dans les pour les connexions d'enfants correspondants 57. Ceci empêchera MapForce de créer les connexions copy-all 60 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.

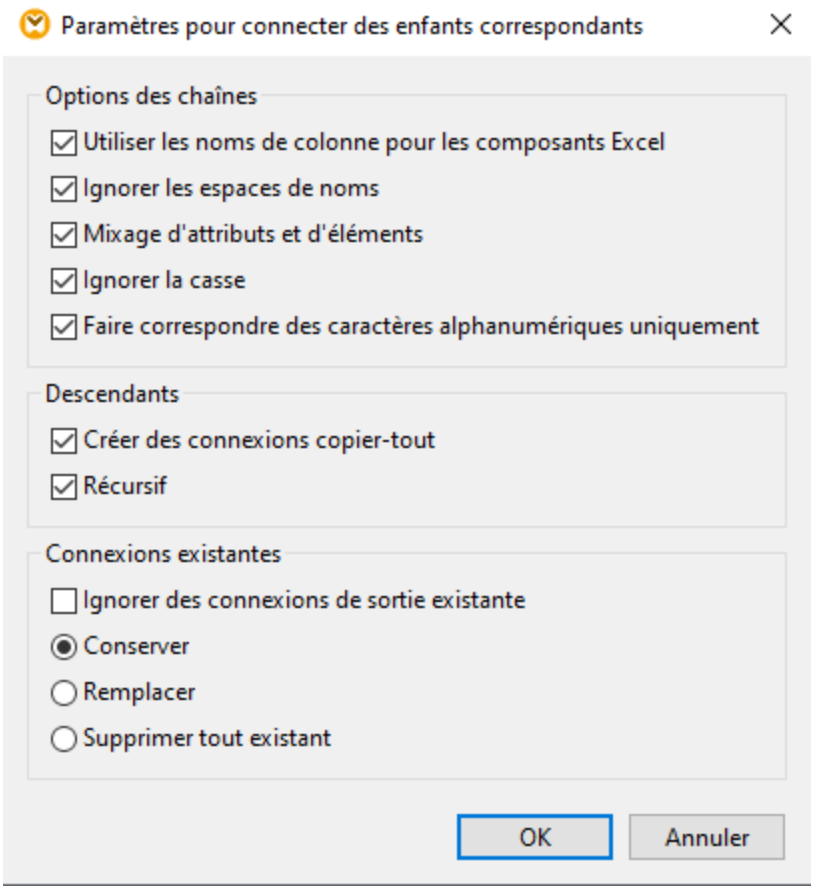
## 2.2.1.2 Connecter des enfants correspondants

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[60], 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.

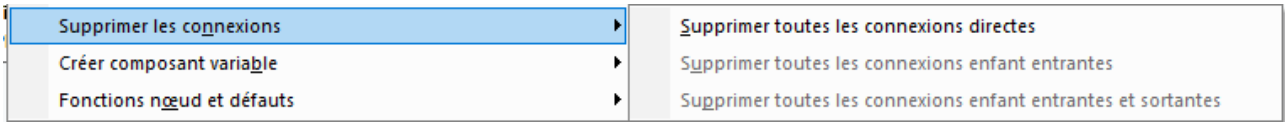
*Connexions existantes*
La section *Connexions existantes* précise quoi faire des connexions existantes. Les options suivantes sont disponibles :

- *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 :

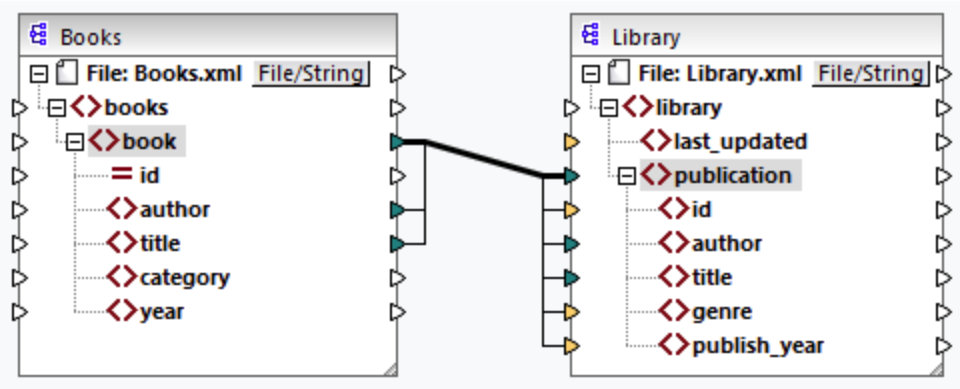
1. Allez au menu **Connexion**.
2. Cliquez sur **Paramètres pour connecter des enfants correspondants**.
3. Vérifiez la case **Créer des connexions copier-tout** et cliquez sur **OK**.
4. Appuyez sur la touche de la barre d'outils **Toggle auto-connexion d'enfants**. De manière alternative, allez au 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 21.
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 94.
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).

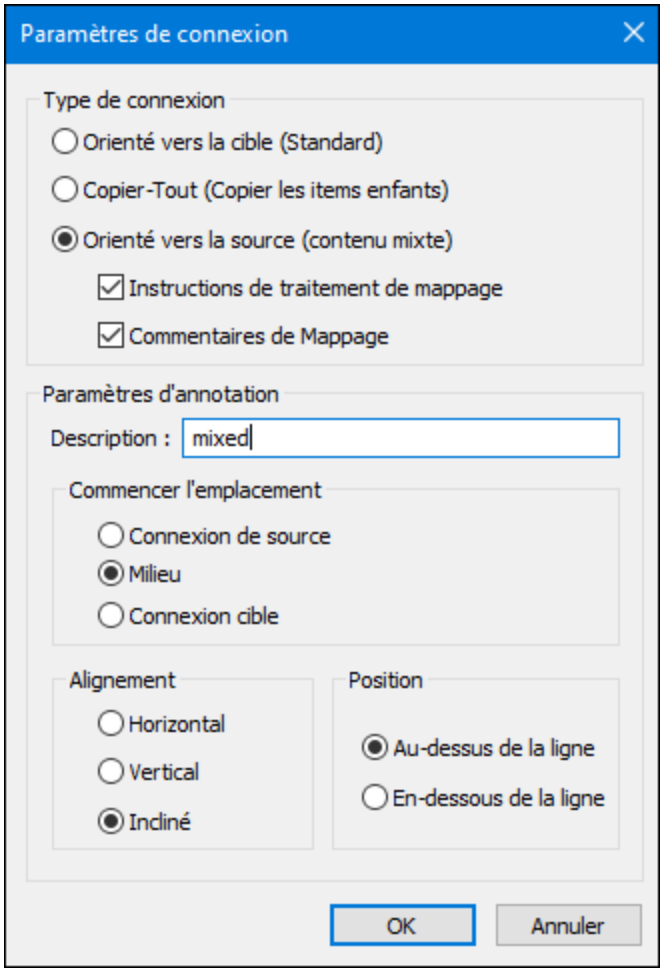
La duplication de l'entrée[45] n'a de sens que si vous souhaitez que la cible accepte des données depuis plus d'une entrée, ce qui n'est pas nécessaire ici. Si vous choisissez de remplacer la connexion "Copier-tout", un message vous invite à nouveau soit à résoudre, soit à supprimer la connexion "Copier-tout". Cliquez sur **Résoudre la connexion copier-tout** si vous souhaitez remplacer la connexion copier-tout par des connexions orientées vers la cible[55] 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[491], 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

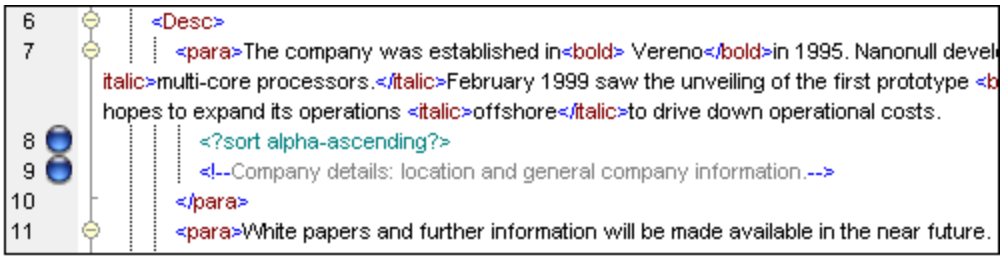
La boîte de dialogue **Paramètres de connexion** définit les paramètres d'une connexion. Pour ouvrir cette boîte de dialogue, double-cliquez sur la connexion. De manière alternative, cliquez avec la touche de droite sur une connexion et sélectionnez **Propriétés** depuis le menu contextuel. Les paramètres 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)[55] sont adaptées pour la plupart des scénarios de mappage.
- Les connexions Copy-all (Copier tous les items enfant)[60] : 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)[55] 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

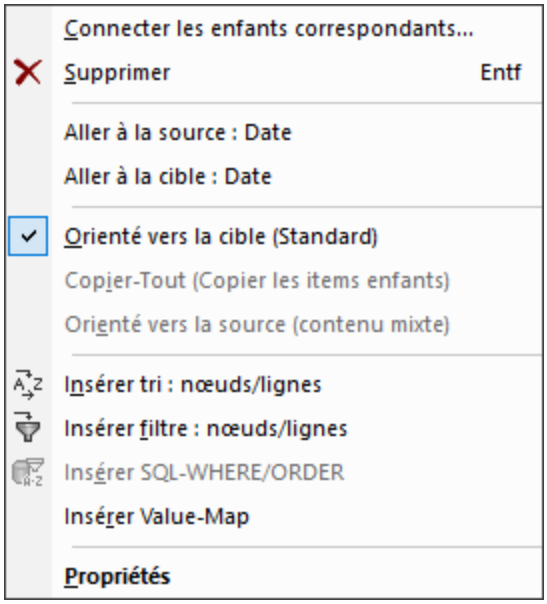
La section des **paramètres d'annotation** section vous permet d'étiqueter la connexion. Cette option est disponible pour tous les types de connexion. Pour annoter une connexion, suivez les instructions ci-dessous :

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[57]. 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[36] de la connexion sélectionnée.
- *Aller à la cible : <item name>* Met en surbrillance le connecteur d'entrée[36] de la connexion sélectionnée.

*Types de connexion*

Voir les détails sur les types de connexion dans Types de connexion[54] et Paramètres de connexion[61].

*Commandes d'insertion*

- *Insérer Tri : Nœuds/Lignes :* Ajoute un composant de tri[438] entre un nœud source et un nœud cible.
- *Insérer Filtre : Nœuds/Lignes :* Ajoute un composant de filtre[444] 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*). Pour tout détail, voir Filtrer et trier des données de base de données[365].
- *Insérer Value-Map :* Ajoute une value-map[451] entre le nœud source et le nœud cible.

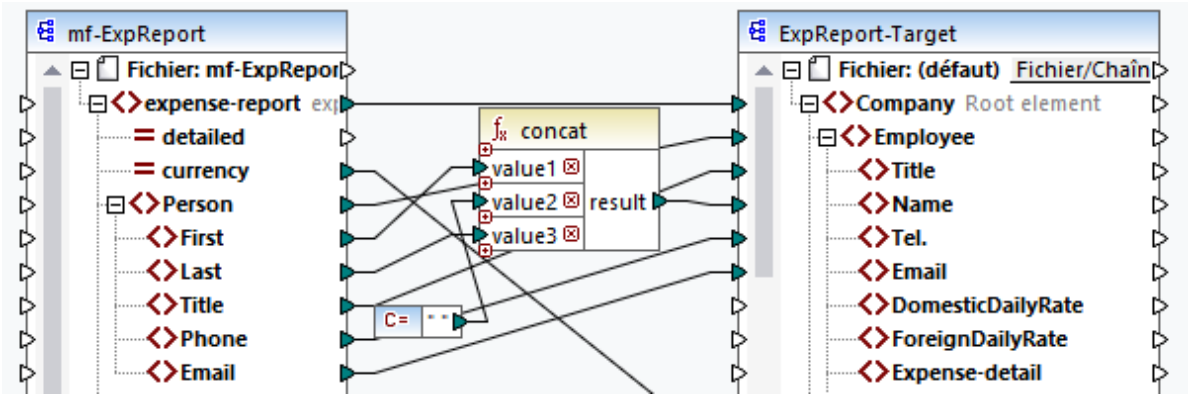
*Propriétés*

Ouvre la boîte de dialogue des Paramètres de connexion[61].

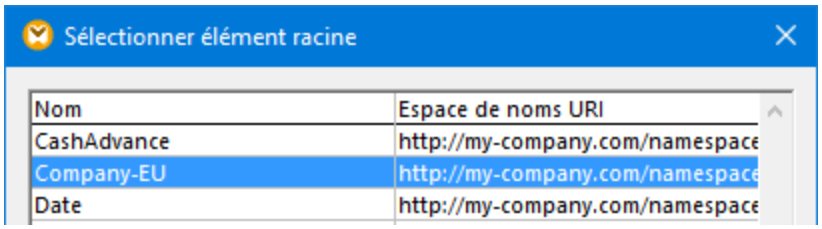
## 2.2.4 Connexions incorrectes

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.

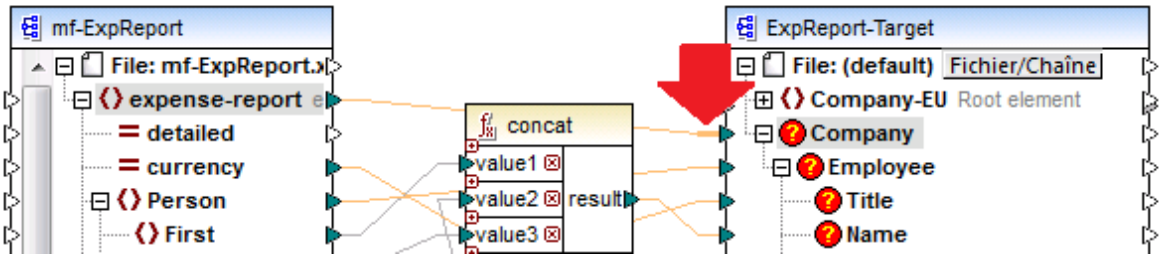
1. Ouvrez **`Tut-ExpReport.mfd`** disponible sous le dossier du Tutoriel[21]. La partie de ce mappage est affiché ci-dessous.




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 [44] 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

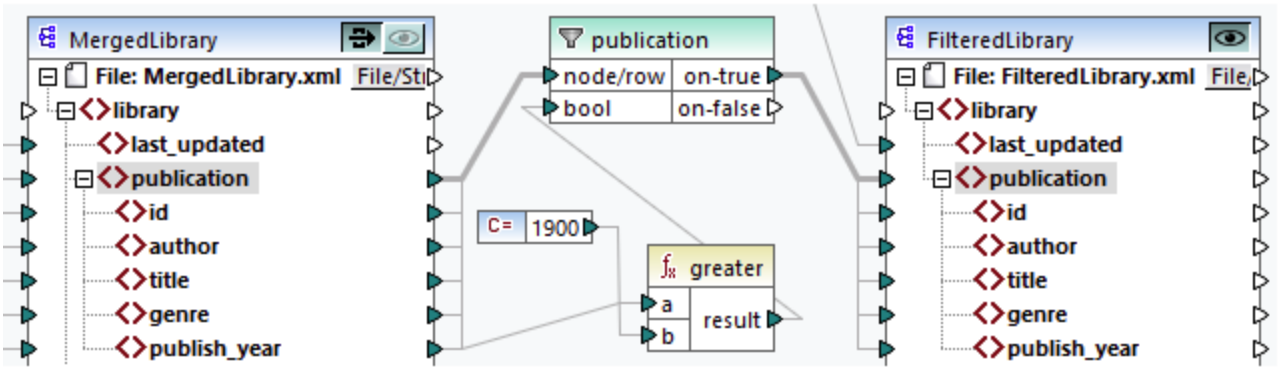
MapForce vous permet de garder les connexions même après avoir supprimé quelques composants de transformations [37] : par ex., les variables, composants de tri et de filtre, value-maps, entrées simples, composants SQL/NoSQL-WHERE/ORDER. Les connexions peuvent être simples ou multiples. Garder les connexions 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 21.

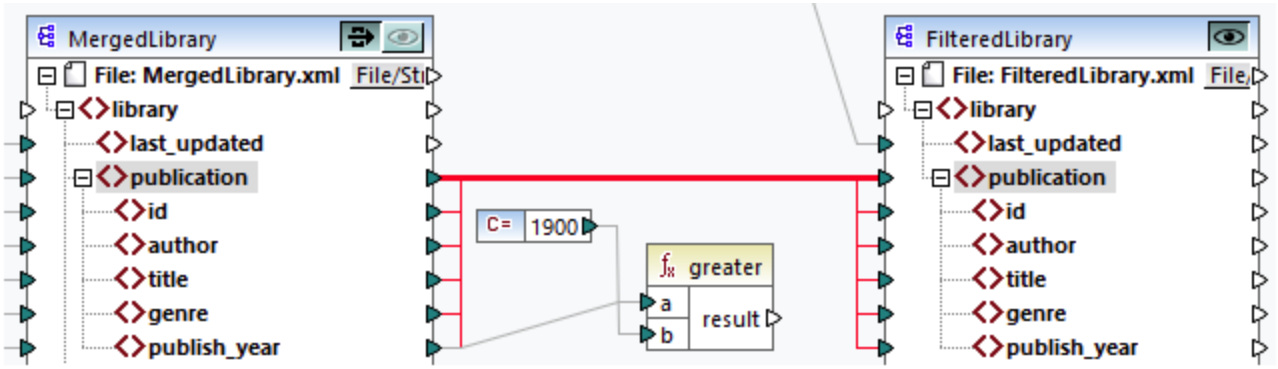
*Avant la suppression*

La capture d'écran ci-dessous montre que les connexions copy-all 60 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 30. La fenêtre **Messages** vous permet également de prendre des actions liées aux messages 31. 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 38 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[33].

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[76]. Tout avertissement, erreur ou messages d'information lié à l'exécution de mappage est affiché dans la fenêtre Messages[30].

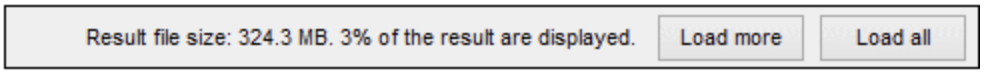
Si vous sélectionnez C++, C# ou Java (*éditions Professional et Enterprise*) en tant que langage de transformation[22], MapForce exécute le mappage utilisant son moteur de transformation built-in et affiche le résultat dans le volet **Sortie**.

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[1086].

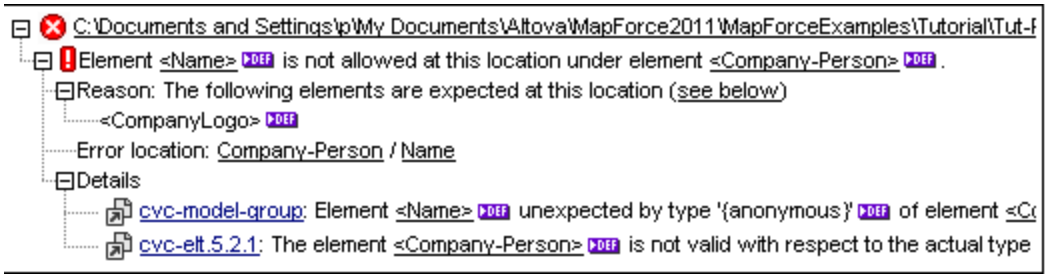



*Valider la Sortie*

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 [22] 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 [1362].
- Pour plus d'information concernant les chemins de handling dans le code généré, voir Chemins dans les environnements d'exécution [49].
- *Éditions Professional et Enterprise :* Vous pouvez changer les options générales de génération de code dans la section *Génération* du dialogue **Options** . Pour les détails, voir Génération [1089].
- *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. Pour plus de détails, voir Mappages de base de données dans des environnements d'exécution variés [159].

*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<br><br>Frameworks cible :<br><br>• .NET Framework<br>• .NET Core 3.1<br>• .NET 5.0<br>• .NET 6.0<br>• .NET 8.0 | Java SE JDK 8, 11, 17, 21 (y compris OpenJDK)<br>Eclipse 4.4 ou plus<br>Apache Ant |
| **mises en œuvre XML DOM** | MSXML 6.0<br>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 79. Pour les détails sur les paramètres de génération de code dans le dialogue **Options**, voir Génération 1089.
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 30.

## 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 1362).

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. Pour plus d'information sur les projets et paramètres liés au projet ainsi que les procédures, voir Projets 82.

### 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. Pour plus d'information sur ces procédures, voir le Générateur de Code 935. Vous pouvez aussi modifier le code Java, C# et C++ généré et l'intégrer dans votre code personnalisé. Pour les détails, voir Intégrer le code généré 943.

*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 79. `<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 44 et Chemins de bibliothèque dans du code généré 50.
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.

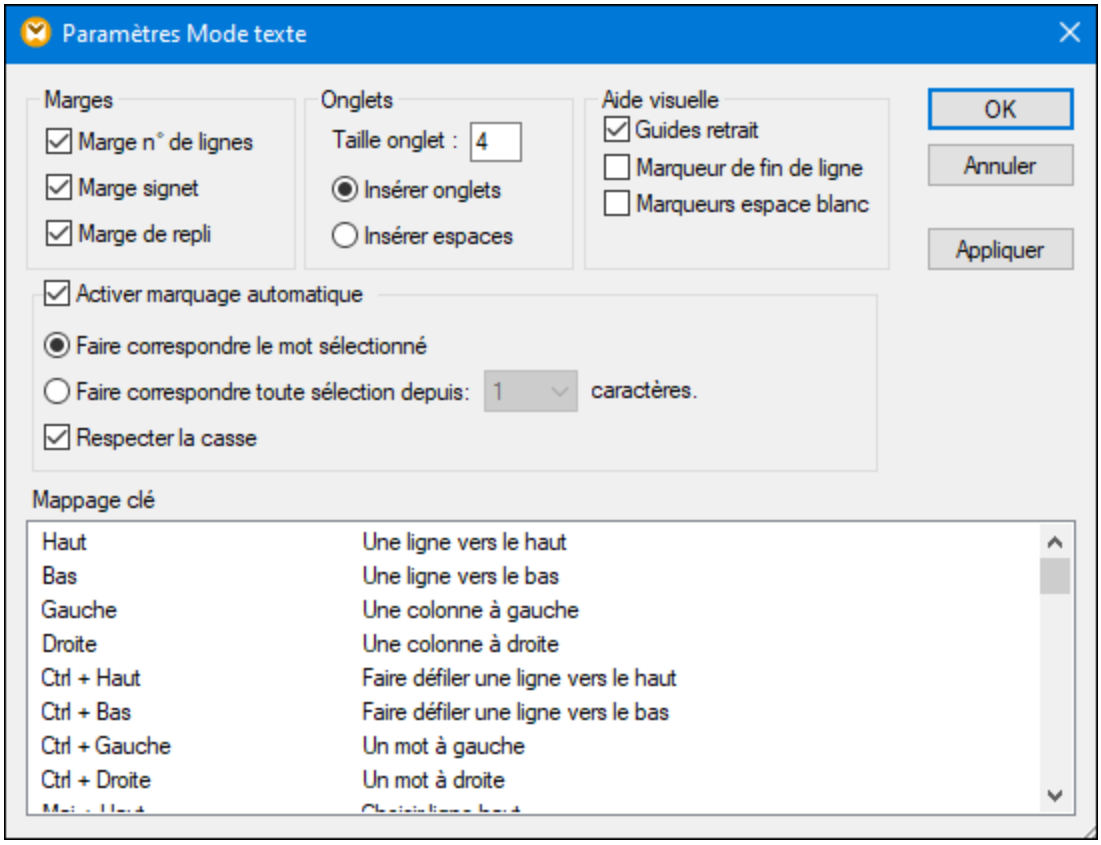
Si votre mappage est chained 105, un fichier de transformation séparé sera généré pour chaque composant cible.

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 33, volet XQuery 32, et volet XSLT 32 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

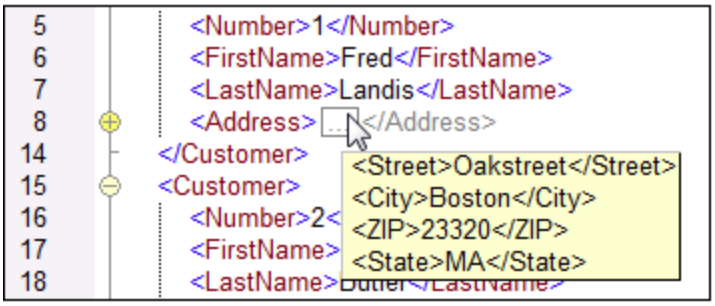
sélectionnée, les signets sont affichés dans la marge à signet (*voir la capture d'écran ci-dessous*). Si la case à cocher **marge à signet** n'est pas sélectionnée, les lignes marquées d'un signet de couleur cyan.


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.

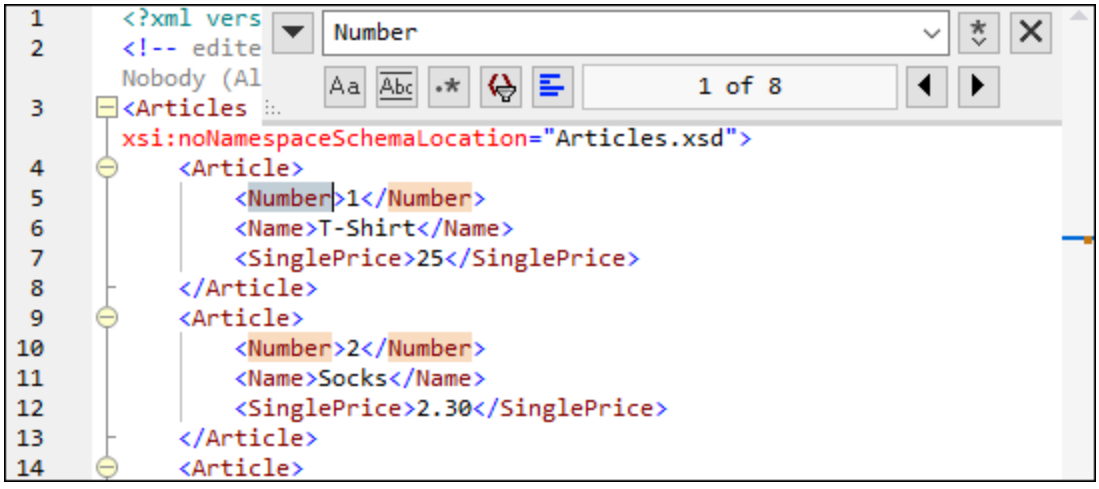
*Retour automatique à la ligne*
Le retour à la ligne aide à afficher une liste de codes à l'intérieur des bords de l'espace de travail. Si le paramètre de retour à la ligne n'est pas activé, certaines parties de texte peuvent ne pas être entièrement visibles dans l'espace de travail. Pour activer le 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**, le volet **Xquery,**, 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.



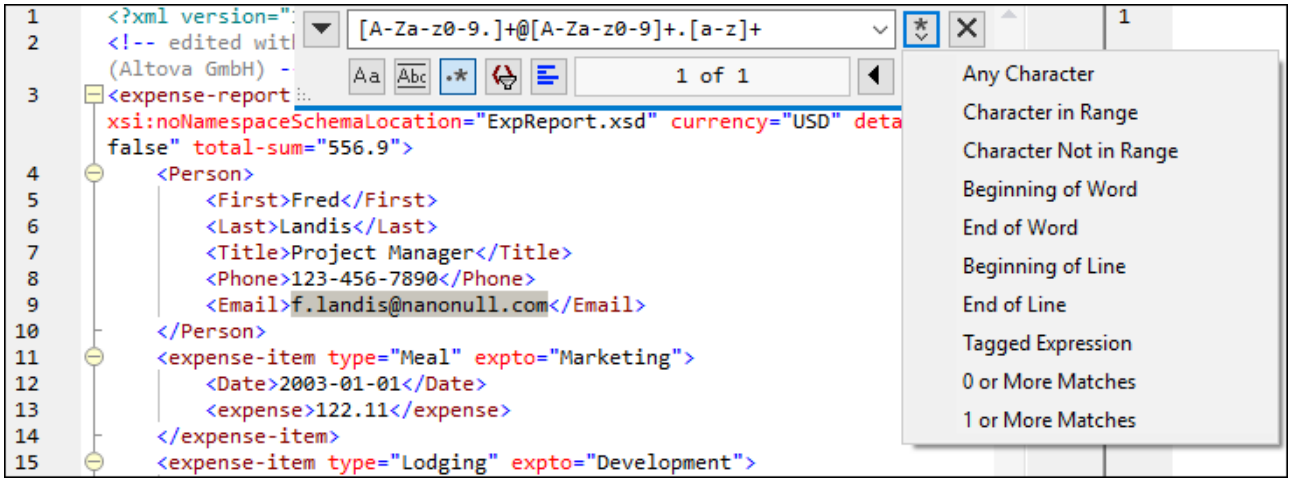
## Options de Recherche

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.<br><br>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

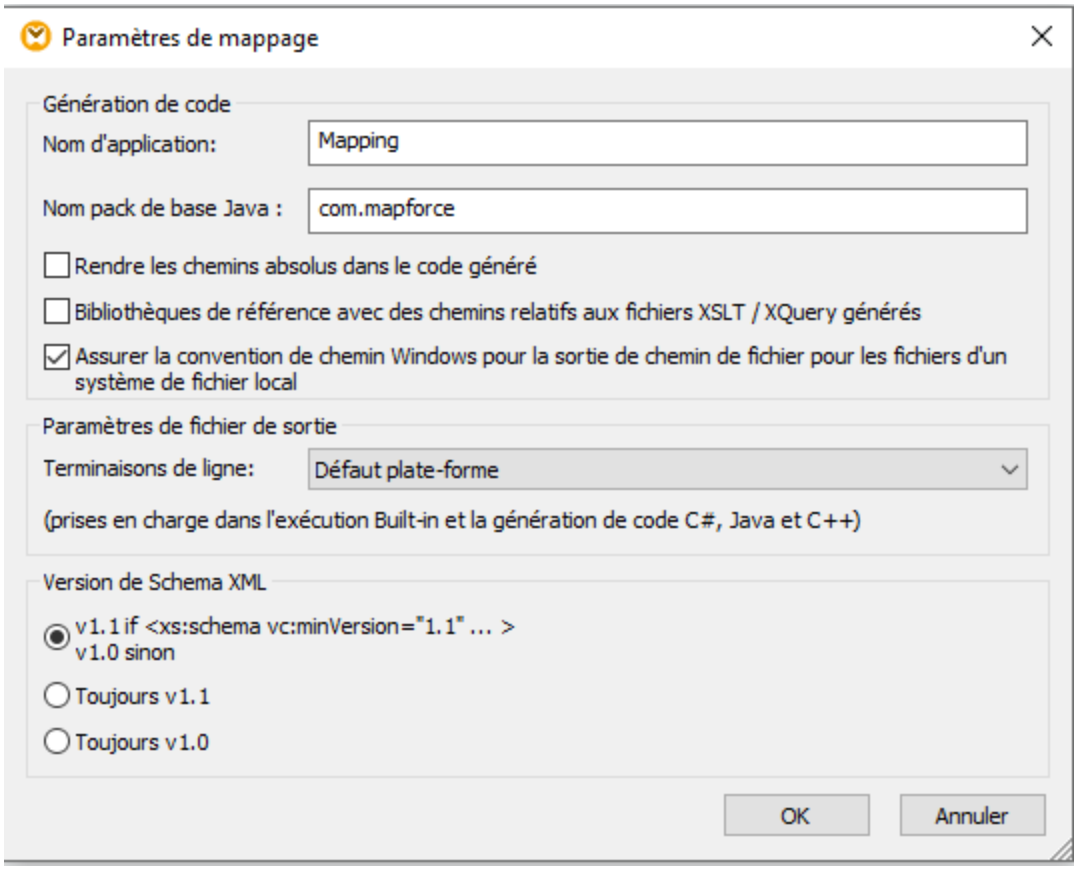
Si l'option **Utiliser des expressions régulières** est activée, vous pouvez rechercher n'importe quel des caractères spéciaux suivants au sein du texte :

- `\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, dans les fichiers d'exécution de MapForce Server (`.mfx`) et dans les fonctions de mappage déployées dans FlowForce Server. Pour plus d'informations, voir Chemins dans des environnements d'exécution 49.

- *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 50.

- *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

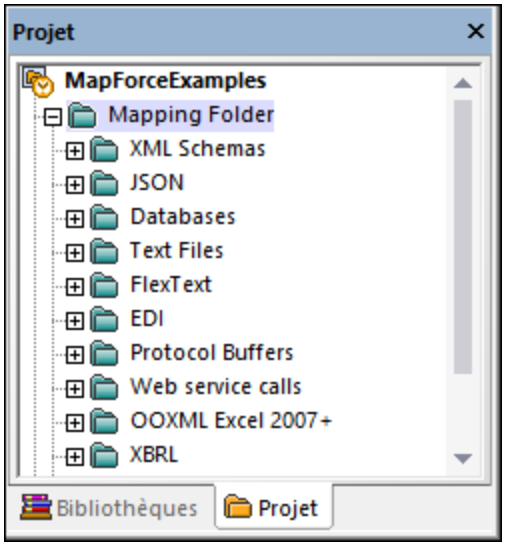
numéro de version du document . Changer ce paramètre dans un mappage existant entraîne un rechargement de tous les schémas de la version de schéma XML sélectionnée, et peut aussi modifier sa validité.

- 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.

## 2.4 Projets

Outre le fait de créer des mappages individuels, vous pouvez aussi créer des projets de mappage qui contiennent plusieurs mappages. Des mappages qui sont ajoutés à un projet sont facilement accessibles depuis la fenêtre **Projet** (*voir la capture d'écran ci-dessous*).




L'avantage principal des projets est que vous pouvez définir des paramètres de génération de code[84] commun (comme le langage cible et le répertoire de sortie) pour tous les mappages, y compris ceux contenus dans ce projet en particulier. Vous pouvez aussi créer des dossiers dans des projets et spécifiez des paramètres de génération de code personnalisés pour chaque dossier individuel dans un projet. Les fichiers de projet de MapForce sont enregistrés avec l'extension **.mfp**.

Dans l'édition MapForce Enterprise, vous pouvez aussi créer des projets de Service Web. De tels projets vous permettent de générer du code de programme Java ou C# qui met en place les services Web SOAP, sur la base de fichiers WSDL (Web Services Description Language) existants.

### 2.4.1 Notions de base du projet

Les sous-sections ci-dessous vous aideront à démarrer avec le projet. Les procédures associées aux projets peuvent être largement divisées en (i) créant un projet, (ii) organisant un projet, et en (iii) réalisant différentes actions. Voir les informations concernant ces procédures dans les sous-sections ci-dessous.

#### Nouveau projet

Pour créer un nouveau projet, procédez comme suit :

1. Cliquez sur le bouton dans la barre d'outils. En alternative, allez au menu **Fichier** and cliquez sur **Nouveau**.
2. Sélectionnez **Fichier de projet**, puis cliquez sur **OK**.

3. Saisissez le nom du projet dans le dialogue **Enregistrer le projet sous** et cliquez sur **Enregistrer**. Le nouveau projet est maintenant affiché dans la fenêtre de **Projet**.

Pour fermer un projet, allez au menu **Projet** et cliquez sur **Fermer Projet**.

## Organisation du projet

*Ajouter un mappage à un projet*
Si vous voulez ajouter un mappage actuellement actif au projet, suivre une des étapes suivantes :

- Allez au menu **Projet**, cliquez sur **Ajouter Fichier actif au projet**.
- Cliquez avec la touche de droite sur le projet pertinent dans la fenêtre **Projet** et sélectionnez **Ajouter Fichier actif au projet** depuis le menu contextuel.

Pour ajouter des fichiers de mappage existant au projet, suivez une des étapes suivantes :

- Allez au menu **Projet**, cliquez sur **Ajouter Fichier au projet**.
- Cliquez avec la touche de droite sur le projet pertinent dans la fenêtre **Projet** et sélectionnez **Ajouter Fichiers au projet**.

**Astuce :** si vous souhaitez ajouter de multiples fichiers, tenez appuyée la clé **Ctrl** tout en sélectionnant les fichiers dans la boîte de dialogue **Ouvrir**.

*Supprimer un fichier depuis un projet*
Pour supprimer un fichier ou un dossier d'un projet, suivez une des étapes suivantes :

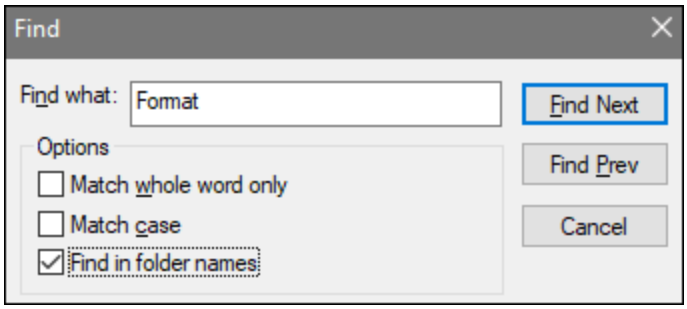
- Sélectionnez le fichier que vous souhaitez supprimer dans la fenêtre **Projet**. Cliquez avec la touche de droite sur le fichier et sélectionnez **Supprimer** depuis le menu contextuel.
- Sélectionnez le fichier pertinent dans la fenêtre **Projet** et appuyez sur **Supprimer**.

Les fichiers de projet de MapForce ont l'extension `.mfp`. Vous pouvez ouvrir des projets de MapForce de la même façon que les mappages : Allez au menu **Fichier** et cliquez sur **Ouvrir**. Par défaut, quand vous exécutez MapForce pour la première fois, vous verrez le projet `MapForceExamples.mfp` dans la fenêtre **Projet**.

## Actions liées au projet

*Rechercher un projet*
Pour rechercher un projet pour des fichiers, suivez les instructions ci-dessous :

1. Dans la fenêtre **Projet**, cliquez sur le projet ou le dossier que vous recherchez.
2. Appuyez sur **Ctrl + F**. La boîte de dialogue **Recherche** vous permet de définir les options de recherche. Par exemple, si vous voulez inclure les noms de dossier dans la recherche, sélectionnez l'option **Chercher dans les noms de dossier** (*voir la capture d'écran ci-dessous*).

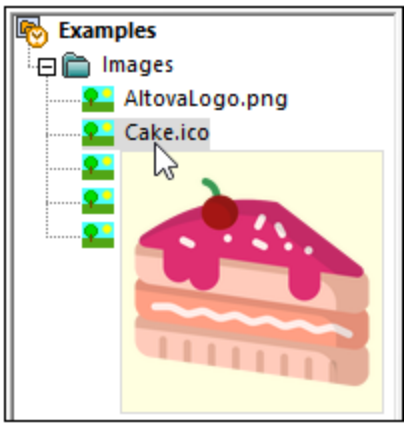



*Générer le code pour votre projet*
Dans les projets, vous pouvez générer le code pour des (i) mappages individuels, (ii) un dossier spécifique ou (iii) tout le projet.

Pour générer un code pour un mappage ou un dossier dans votre projet, cliquez avec la touche de droite sur le mappage pertinent ou dossier et sélectionnez **Générer du code** ou **Générer du code dans**. Si vous sélectionnez **Générer du code**, le code sera généré dans le langage spécifié dans les paramètres de projet[85]. Vous pouvez aussi choisir de générer le code dans une des langues disponibles dans votre édition de MapForce. Pour plus d'information, voir Génération de code[70].

Pour générer du code dans tout le projet, allez au menu **Projet** et sélectionnez **Générer le Code pour tout le projet**. En alternative, cliquez avec la touche de droite sur le nom du projet dans la fenêtre **Projet** et sélectionnez **Générer le code**. Le code sera généré dans le langage spécifié dans les paramètres de projet[85]. En ce qui concerne tout le projet, vous pouvez aussi sélectionner un langage pour générer le code dans le menu **Projet** ou dans le menu contextuel du projet. Le choix des langages dépend de votre édition de MapForce. Pour plus d'information, voir Génération de code[70].

*Prévisualiser les images*
La fenêtre de **Projet** vous permet de prévisualiser des images dans les formats suivants : **`.png`**, **`.jpeg`**, **`.gif`**, **`.bmp`**, **`.tiff`** et **`.ico`** (*voir la capture d'écran ci-dessous*). En double-cliquant sur le fichier de l'image, vous l'ouvrirez dans une application externe, qui dépend de l'association du fichier dans Windows.




*Regardez des tutoriels vidéo et ajoutez des liens Web*
Outre de multiples fichiers échantillon, le projet **`MapForceExamples`** contient également des liens vers divers du site web d'Altova. Double-cliquez sur un lien ouvrira la page correspondante dans votre navigateur Web par défaut.

Vous pouvez également insérer votre propre lien externe comme suit :

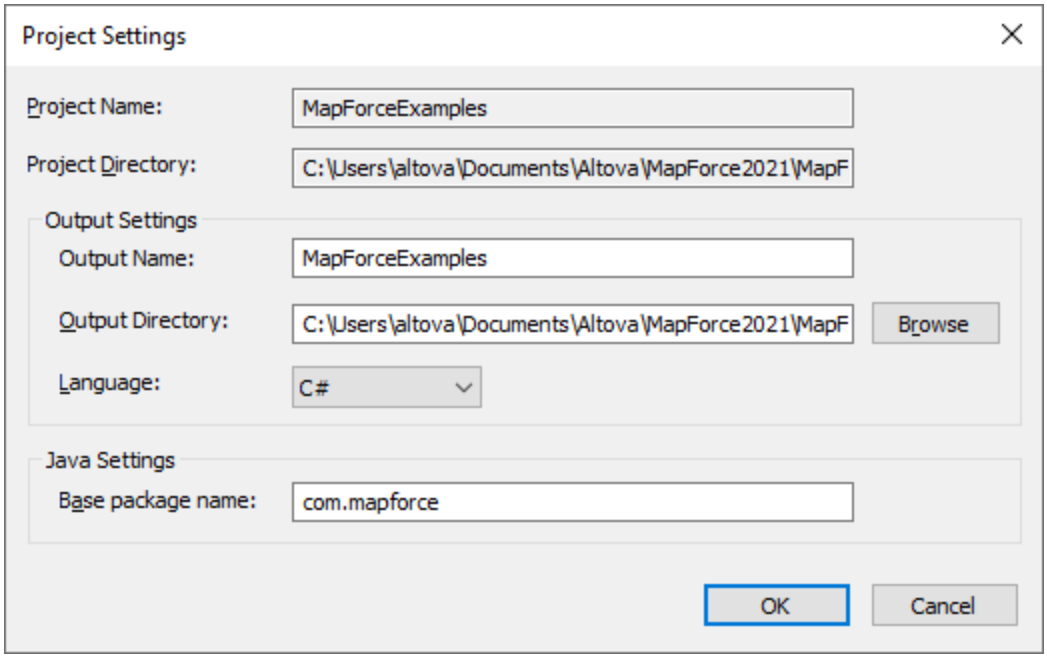
- En cliquant avec la touche de droite sur le nom du projet ou un dossier d'intérêt et sélectionnant **Créer Lien Web** depuis le menu contextuel.
- En sélectionnant le nom du projet d'intérêt, puis en cliquant sur **Créer Lien Web** dans le menu **Projet**.

Chacune des méthodes fait que le dialogue **Propriétés Lien Web** s'ouvre. Saisissez le nom du lien qui sera visible dans l'interface et l'URL de cette ressource. Vous devez aussi sélectionner une icône pour votre ressource : comme lien Web général ou lien vers une vidéo. Vous pouvez toujours modifier l'emplacement d'un lien, en glissant le lien vers l'emplacement désiré. Vous pouvez également utiliser la combinaison de clés **Ctrl + C** et **Ctrl + V** pour copier un lien et le coller dans l'emplacement souhaité.

## 2.4.2 Paramètres de projet

Pour tout projet, vous pouvez spécifier les paramètres de génération de code qui toucheront tous les mappages dans un projet. Pour ouvrir le dialogue **Paramètres de projet** (*voir la capture d'écran ci-dessous*), suivez une des étapes suivantes :

- Cliquer avec la touche de droite sur le nom du projet dans la fenêtre **Projet** et sélectionnez les **Propriétés** à partir du menu contextuel
- Allez au menu **Projet** et cliquez sur **Propriétés**.




Les paramètres disponibles sont recensés ci-dessous. Veuillez noter que le nom du projet et le répertoire du projet ne peuvent pas être changés une fois le projet créé.

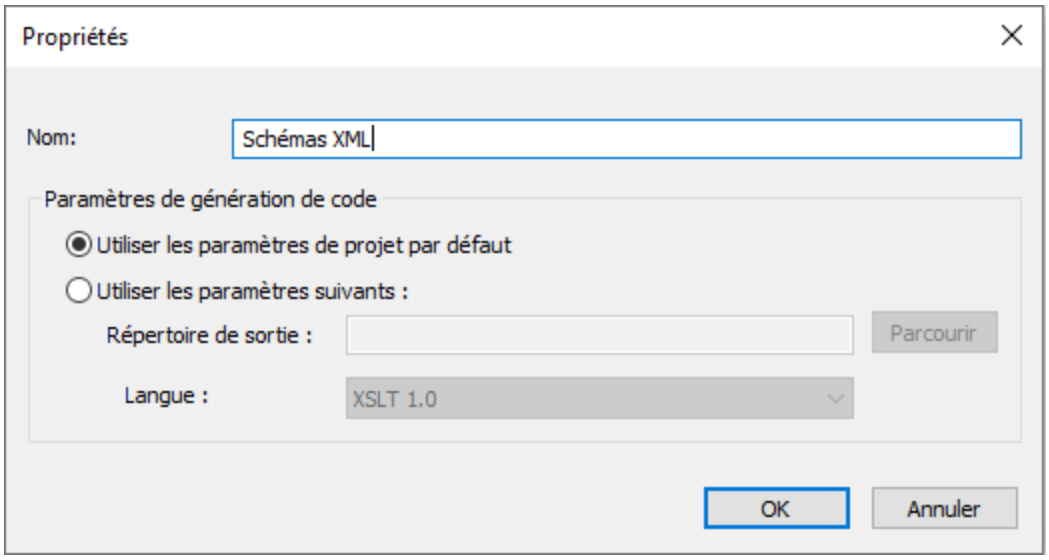
- *Nom de sortie :* La valeur saisie dans le champ de texte détermine les noms du projet/de la solution généré(e) et autres objets du code généré.

- *Répertoire de sortie :* Définit le dossier Windows où le code généré provenant de tous les mappages dans ce projet sera enregistré. Par défaut, la sortie est enregistrée dans le répertoire **`MapForceExamples\output\`**.
- *Langage :* Définit le langage de génération de code pour tous les fichiers de mappage dans ce projet. Pour les détails concernant la génération de code, voir Génération de code 70.
- *Nom de package de base :* Ce paramètre s'applique si Java est sélectionné en tant que langage de transformation. Le paramètre définit le nom du paquet de base dans le projet Java généré.

## 2.4.3 Dossiers de projet

MapForce vous permet d'organiser les mappages à l'intérieur d'un projet dans les dossiers. Vous pouvez créer autant de dossiers dont vous avez besoin et y ajouter des mappages. De tels dossiers sont virtuels et existent uniquement à l'intérieur d'un projet MapForce : Ces dossiers ne correspondent pas aux dossiers de votre système d'exploitation. L'un des avantages de créer des dossiers dans un projet est que vous pouvez définir des paramètres de génération de code communs pour tous les fichiers de mappage dans ce fichier particulier. Pour créer un dossier à l'intérieur d'un projet de MapForce, procédez comme suit :

1. Allez au menu **Projet**, et cliquez sur **Créer dossier**. En alternative, cliquez avec la touche de droite sur le projet dans la fenêtre **Projet** et sélectionnez **Créer dossier**.
2. Dans le dialogue **Propriétés**, (*voir la capture d'écran ci-dessous*)saisissez les paramètres de génération de code requis et cliquez sur **OK**.




La liste ci-dessous décrit les paramètres que vous pouvez définir dans la boîte de dialogue des **Propriétés**.

- *Nom :* Il s'agit du nom du dossier dans votre projet.
- *Utiliser les paramètres de projet par défaut :* Cette option signifie que les paramètres de génération de code dans le dossier actuel sont les mêmes que pour le projet entier. C'est pourquoi, lorsque vous générez du code depuis votre projet, MapForce utilisera les paramètres de génération de code définis dans les paramètres de projet 85. Si votre dossier nécessite des paramètres de génération de code personnalisés, choisissez **Utiliser les paramètres suivants** et spécifiez le répertoire de sortie du code et le langage, le cas échéant.

- *Répertoire de sortie :* Il s'agit du dossier où le code généré provenant de tous les mappages dans ce projet sera enregistré.
- *Langage :* Cette option définit le langage de de génération de code pour tous les fichiers de mappage dans ce dossier.

# 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 21. 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 89 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 100 montre comment fusionner des données depuis de multiples fichiers XML source à un fichier cible.
*Mappages enchaînés*
Dans ce tutoriel 105, 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 113 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

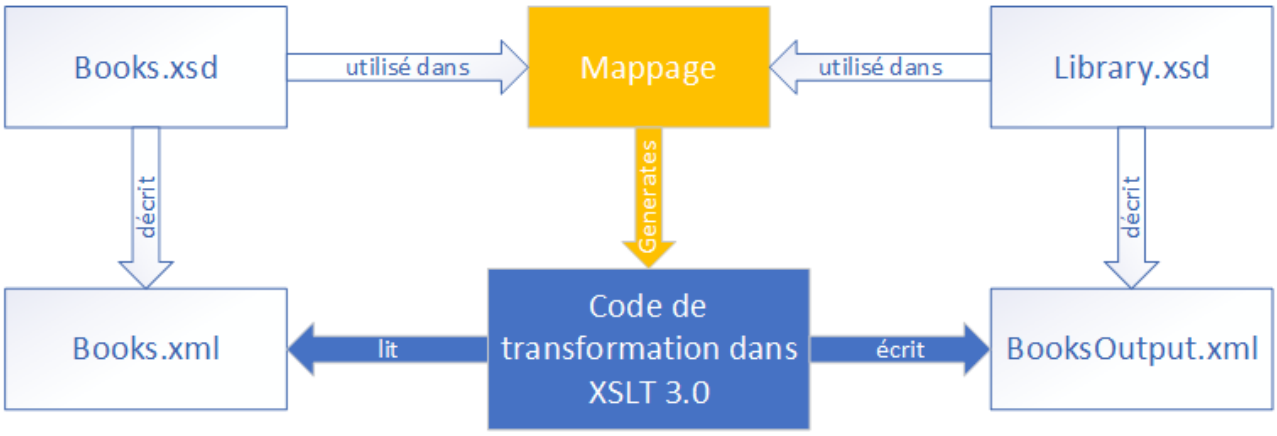
Ce tutoriel décrit comment créer un mappage pour un des scénarios les plus basiques. Notre objectif est de transférer les données depuis un fichier XML avec un schéma XML A y assigné) et mettre ces données dans un fichier XML B (avec un schéma XML B y assigné). Les grandes lignes de notre méthode seront 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[22], 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**. Puis sélectionnez **Mappage** et cliquez sur **OK**.
- Cliquez sur dans la batte d'outils. Puis sélectionnez **Mappage** et cliquez sur **OK**.

### Sélectionner un langage de transformation

Dépendant de votre édition de MapForce, différents langages de transformation[22] sont disponibles. Pour ce tutoriel, nous avons choisi XSLT3. Vous pouvez sélectionner ce langage de transformation d'une des manières suivantes :

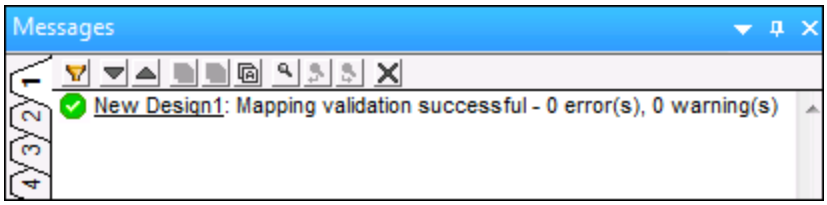
- Cliquez sur **XSLT3** dans la barre d'outils.
- Ouvrez le menu **Sortie** et cliquez sur **XSLT 3.0**.

### 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.

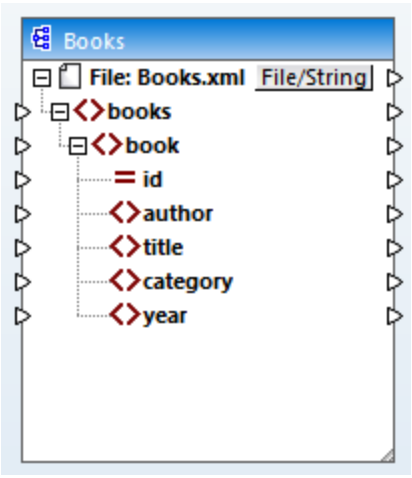
Si vous sélectionnez une des deux premières options, le dialogue **Insérer un fichier Schéma XML** proposera un choix entre un schéma pré-packagé ou un fichier local ou à distance. Dans nos tutoriels, tous les fichiers sont en local. Pour plus d'information sur l'ajout de fichiers XML, voir XML et Schéma XML[123].

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[35]. 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 :



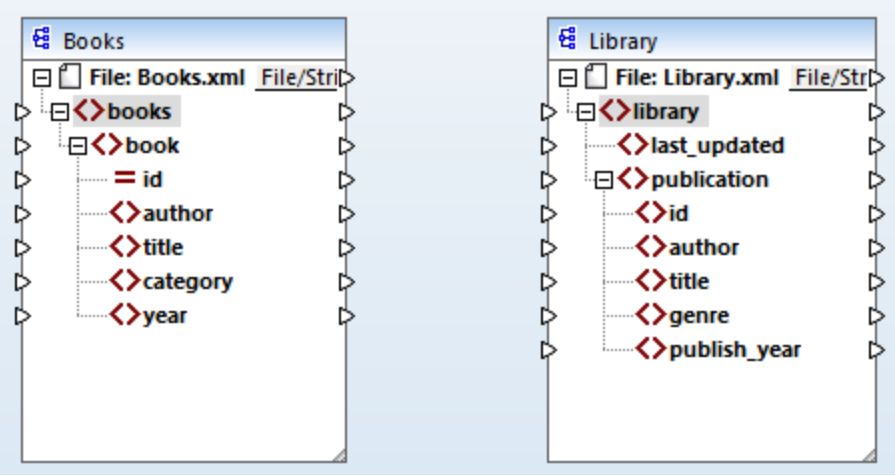

Le nom de l'en-tête (`Books`) fait uniquement référence au nom de composant et non pas au nom de schéma sur lequel ce fichier est basé. Pour voir le nom du schéma et d'autres propriétés du composant, double-cliquez sur l'en-tête du composant. Ceci ouvrira la boîte de dialogue Paramètres de composant[124].

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 35.

## 3.1.3 Ajouter composant cible

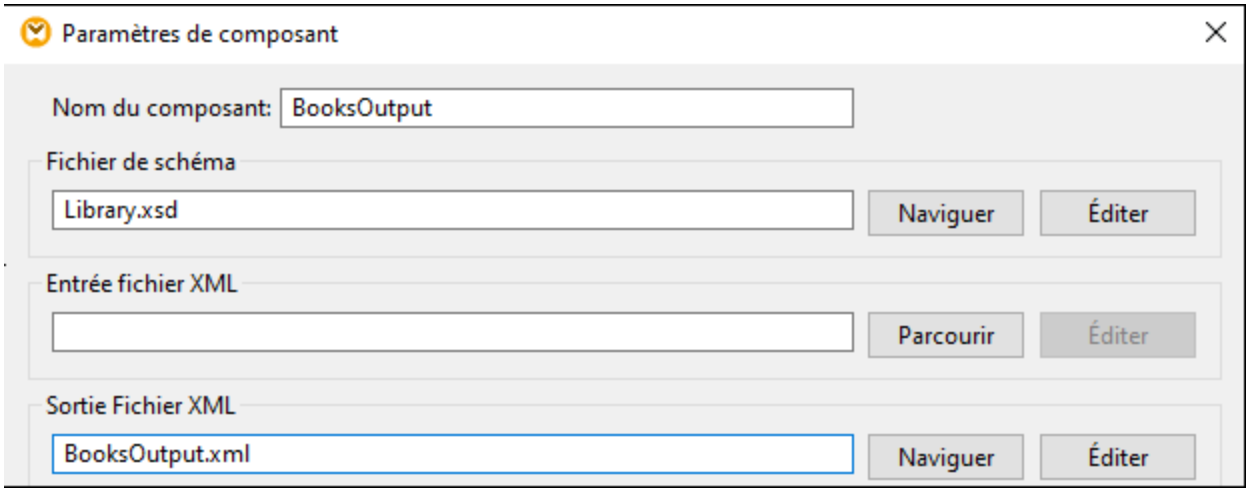
La prochaine étape est d'ajouter un composant cible et de définir ses paramètres. Ajoutez le fichier cible **`Library.xsd`** au mappage. Cliquez sur **Ignorer** lorsque vous êtes invité par MapForce à fournir un fichier d'instance. À cette étape, le design de mappage ressemble à l'exemple suivant :



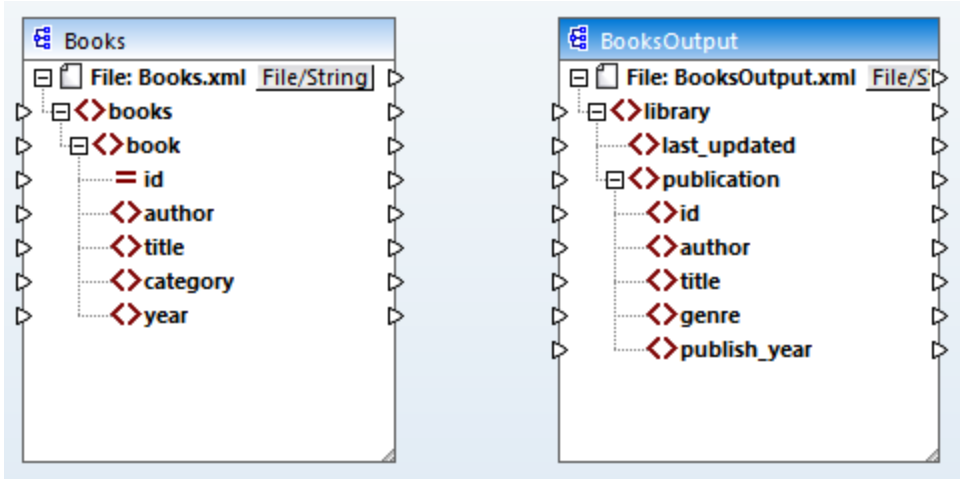

Veuillez noter que lorsque vous ouvrez **`library.xsd`**, il est affiché en tant que fichier XML file dans le composant. En fait, MapForce crée uniquement une référence vers le fichier XML dénommé **`Library.xml`**, mais ce fichier XML lui-même n'existe pas encore. Donc notre compostant cible a un schéma 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 124 (*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

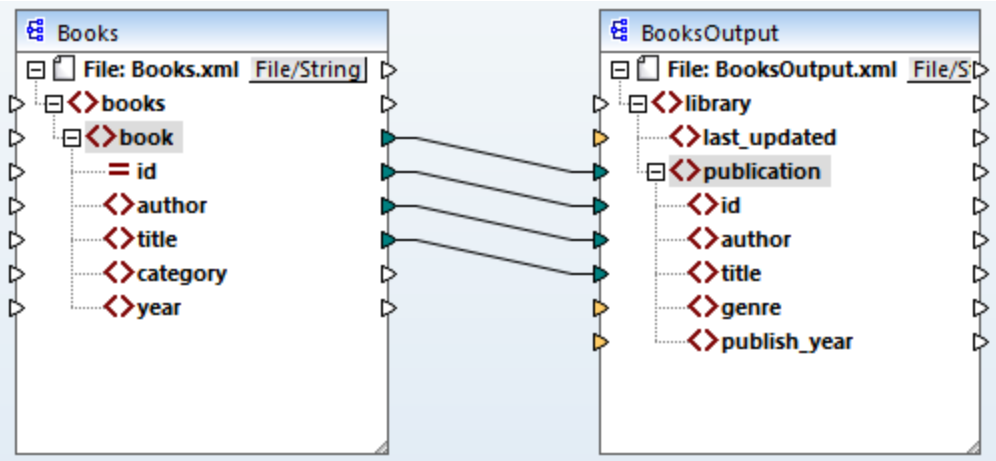
À cette étape, nous allons mapper les données dans le fichier source au fichier cible. Nous fournirons également des informations sur la date et l'heure actuelles utilisant la fonction XPath `current-dateTime`[706].

### 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* [57] 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

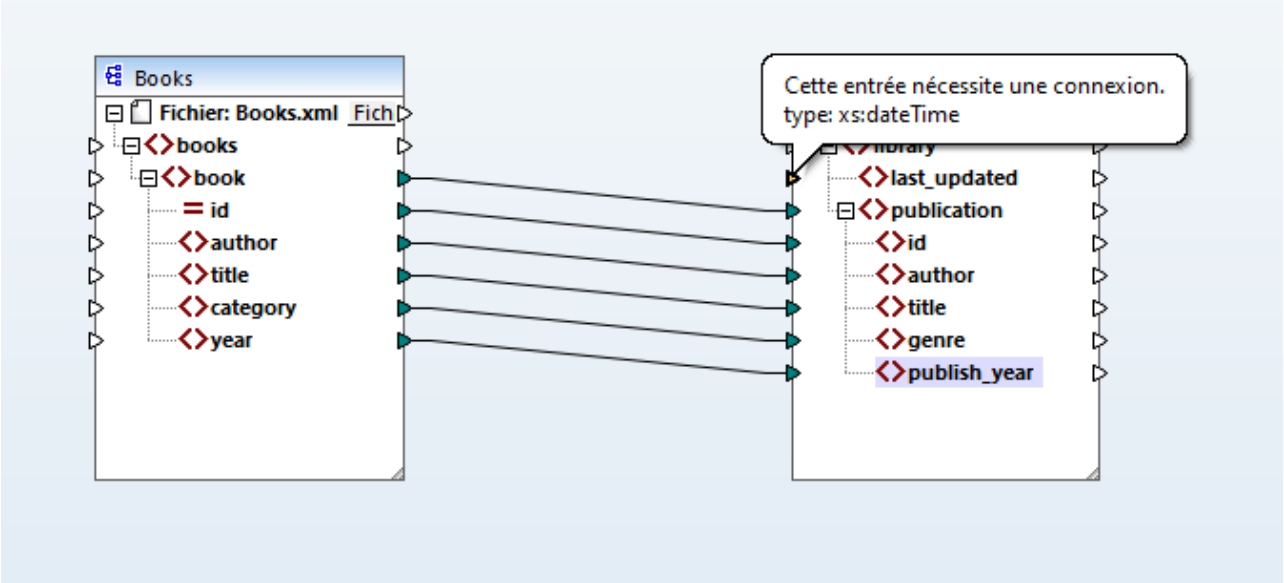
Veuillez noter que certains des connecteurs d'entrée dans le composant cible ont été marqués en orange par MapForce, ce qui indique que ces items sont obligatoires. Ces items sont obligatoires car ils ont été définis de cette manière dans le schéma de fichier. Afin de garantir la validité du fichier 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.

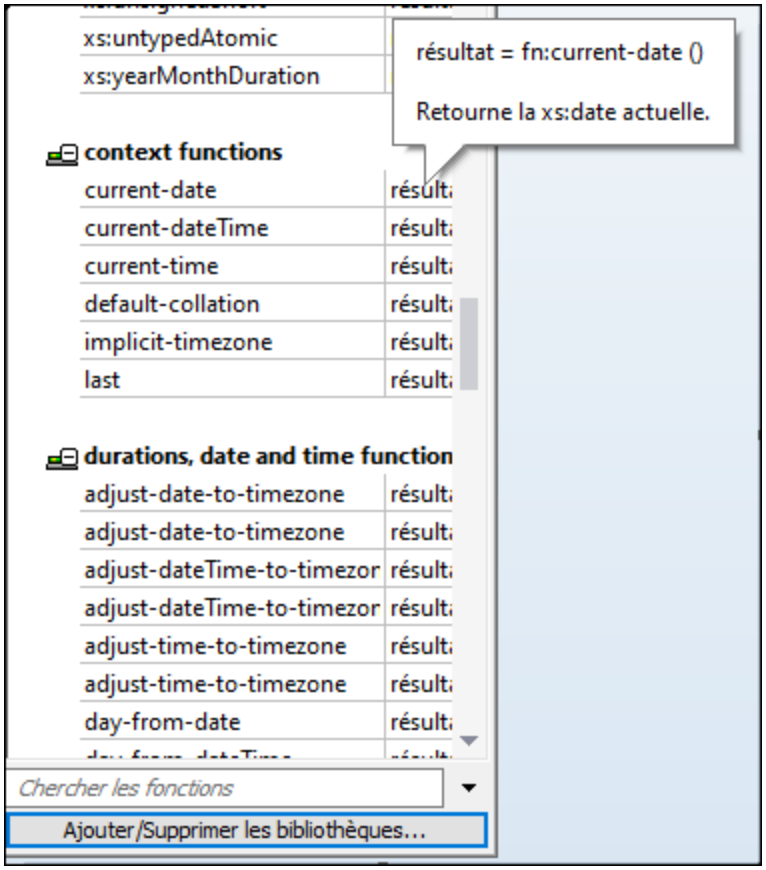
## Ajouter la date et l'heure actuelles

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*).

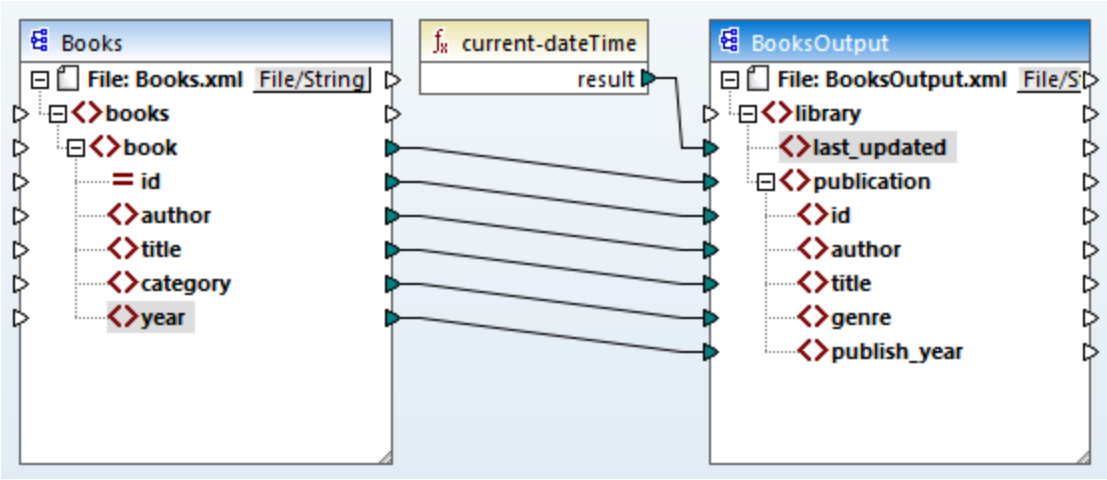
Pour voir les conseils, appuyez sur le bouton de la barre d'outils . En cliquant (**Afficher types de données**) dans la barre d'outils, vous pouvez aussi rendre le type de données visible à tout moment.




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[26] (*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 90.

## 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.

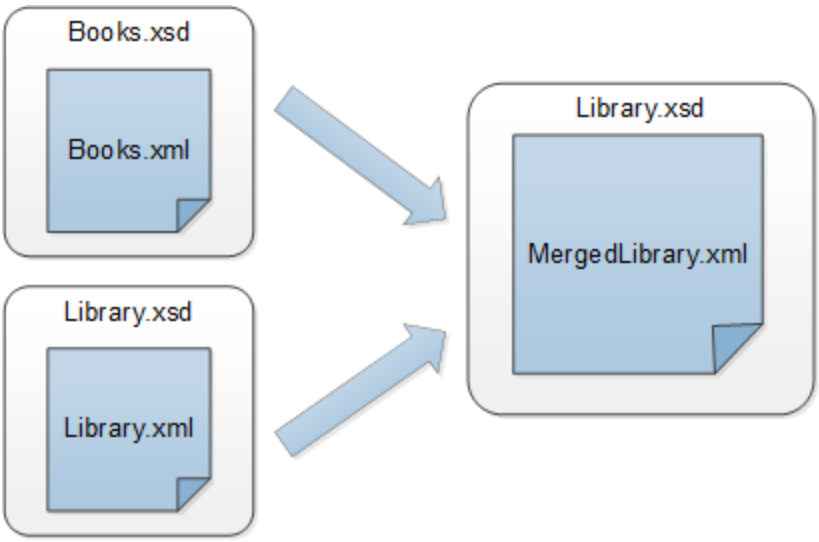
Pour configurer MapForce afin qu'il écrive la sortie directement sur les fichiers finaux au lieu des fichiers temporaires, rendez-vous sur **Outils | Options | Général**, et cochez la case *Écrire directement sur les fichiers de sortie finaux*. Veuillez noter qu'il n'est pas recommandé d'activer cette 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 113). 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>
```

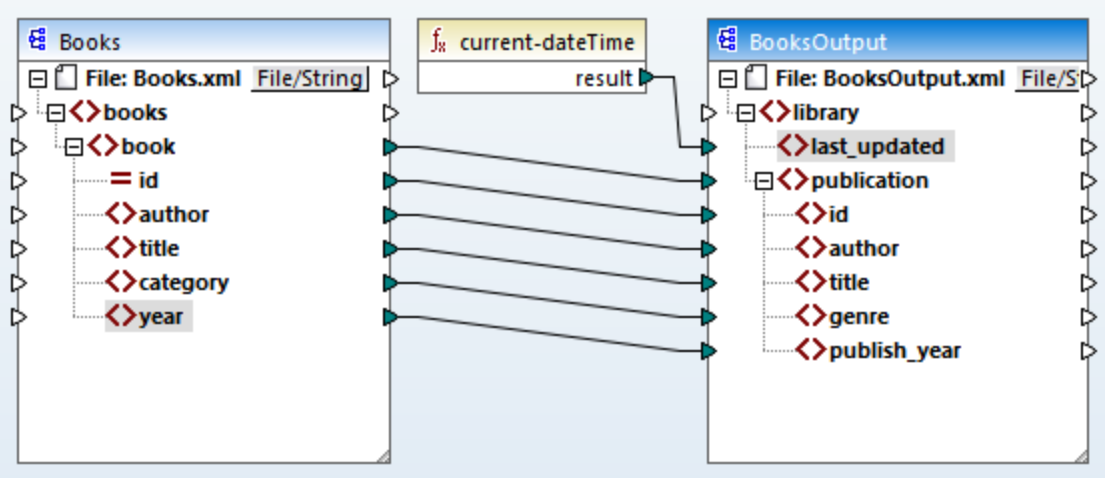
C'est ainsi que nous voulons que nos données fusionnées aient l'air dans le fichier cible dénommé **merged_library.xml** :

```
<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 [89]. Avant d'effectuer un changement au mappage, assurez-vous d'enregistrer ce design sous un nouveau nom dans le dossier **BasicTutorials**.

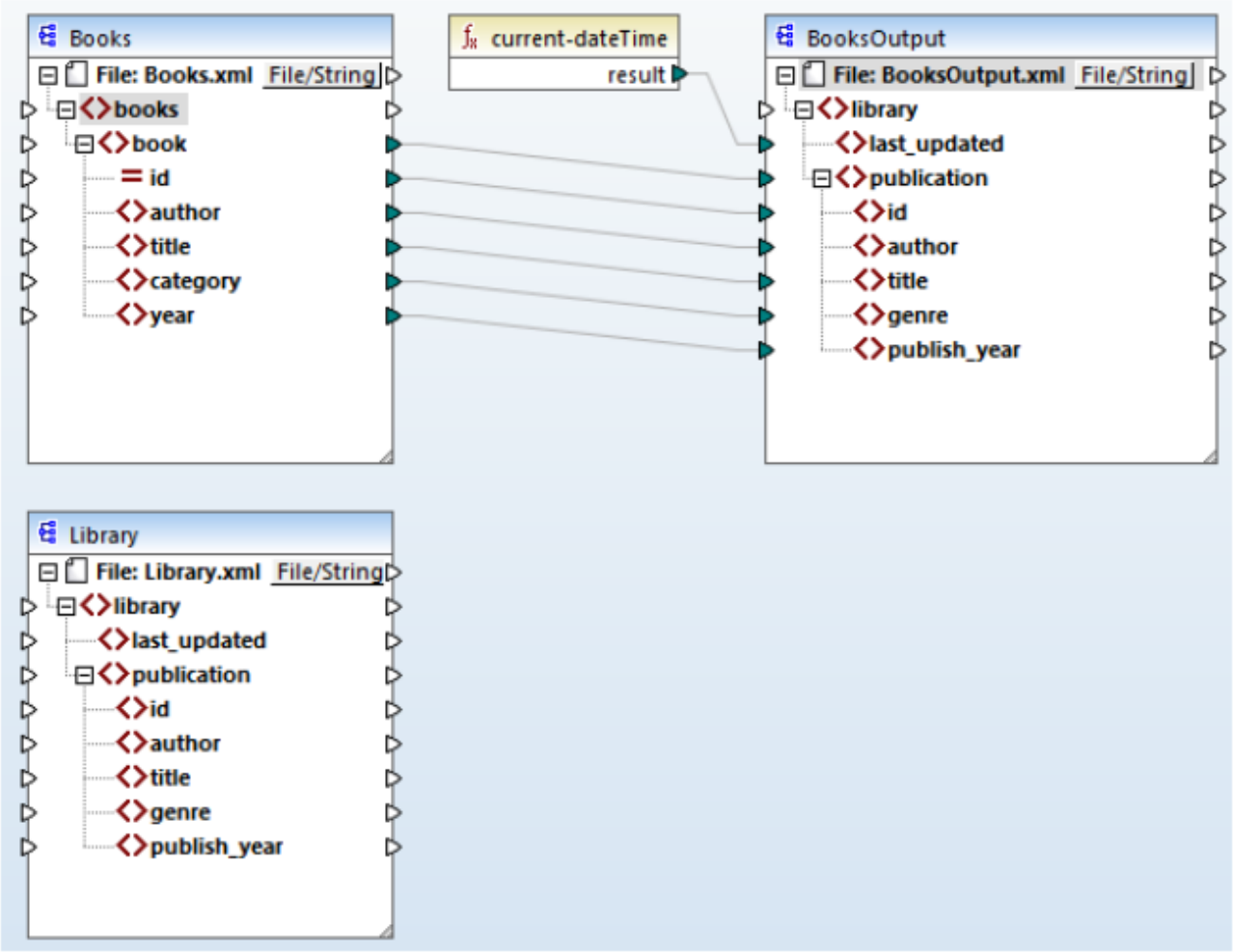


## 3.2.2 Ajouter seconde source

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[124].

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.

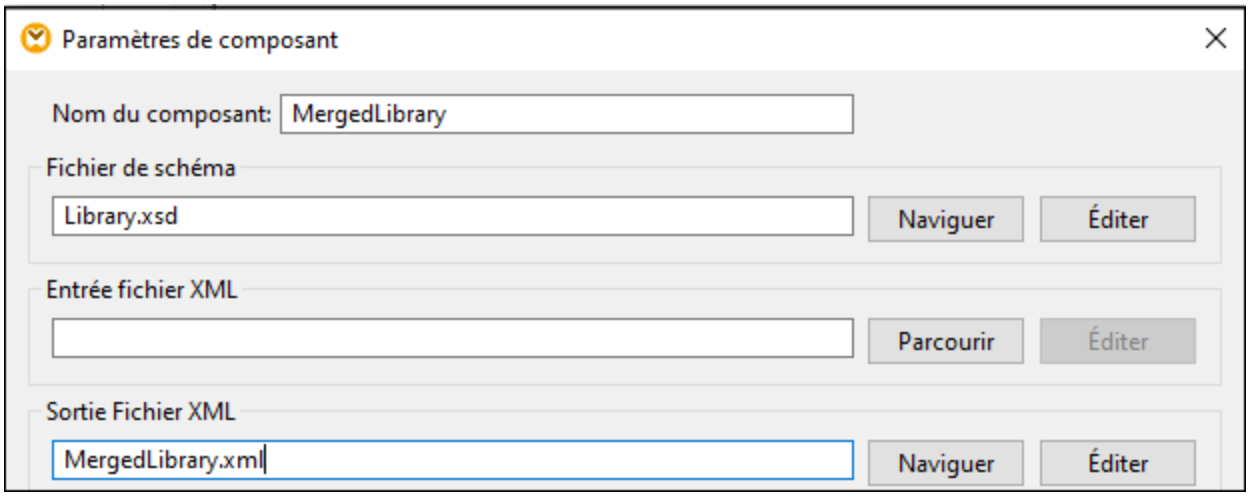
À cette étape, le design de mappage a l'air de ceci :



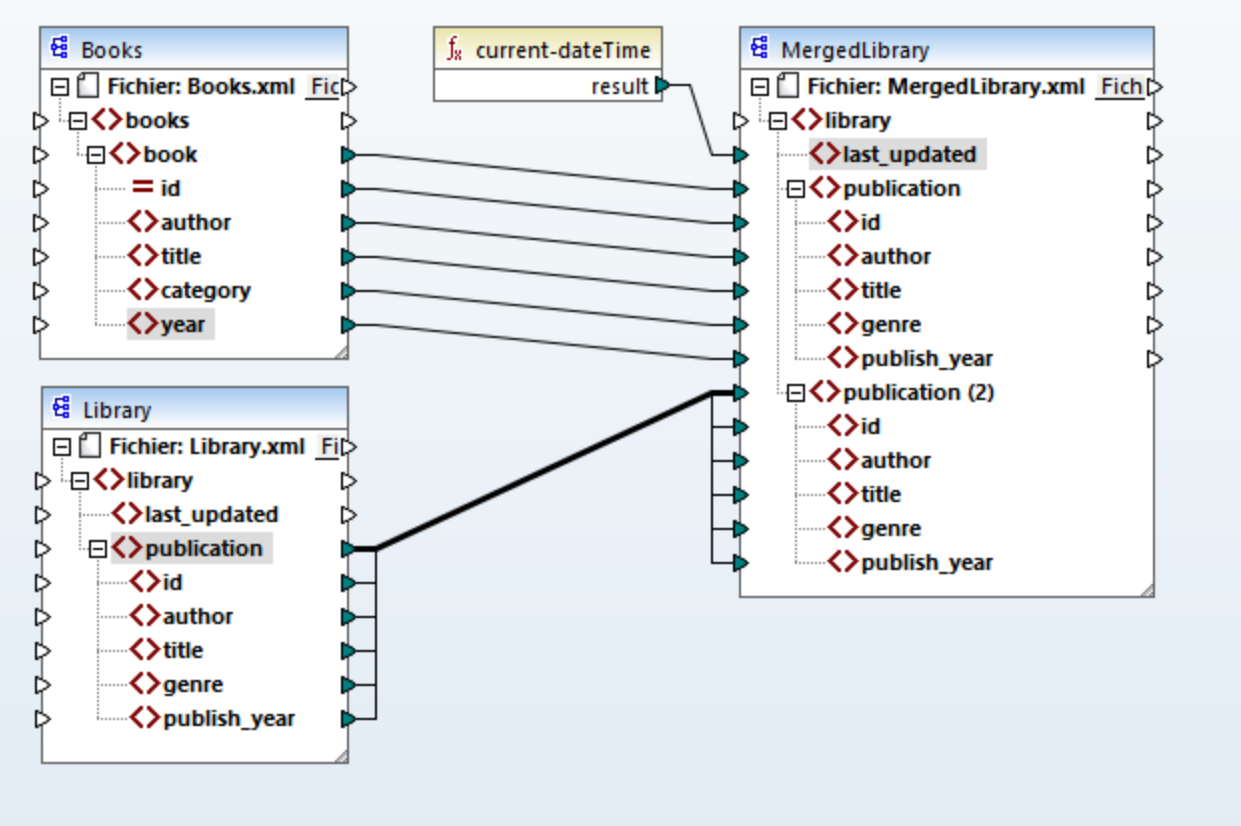
### 3.2.3 Configurer Sortie

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[124]. 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*[60], qui rend le mappage plus compréhensible.

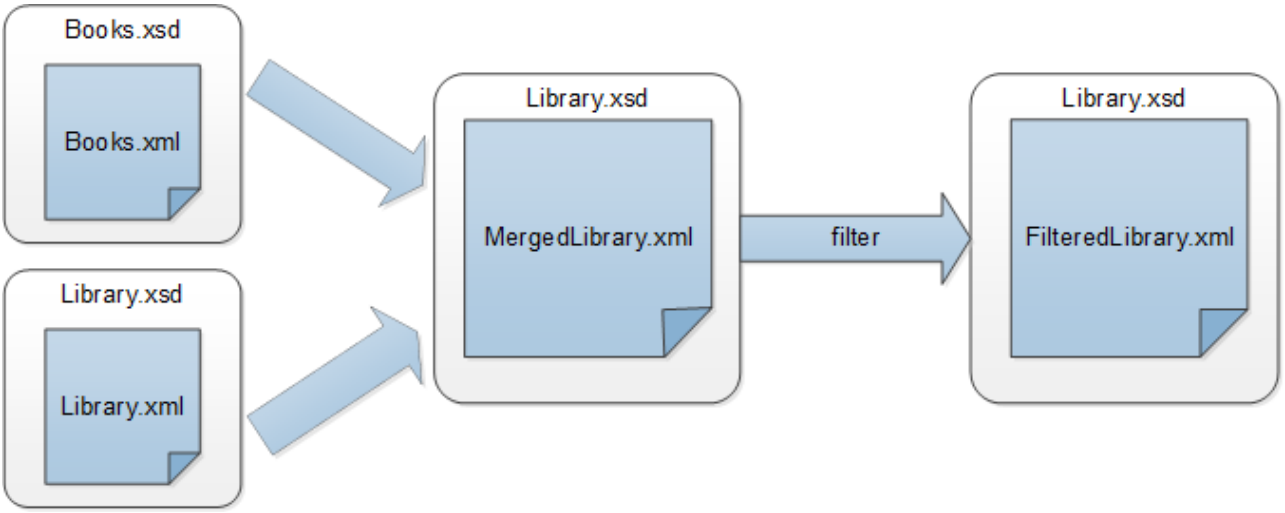
*Consulter la sortie*

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[105].

# 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.



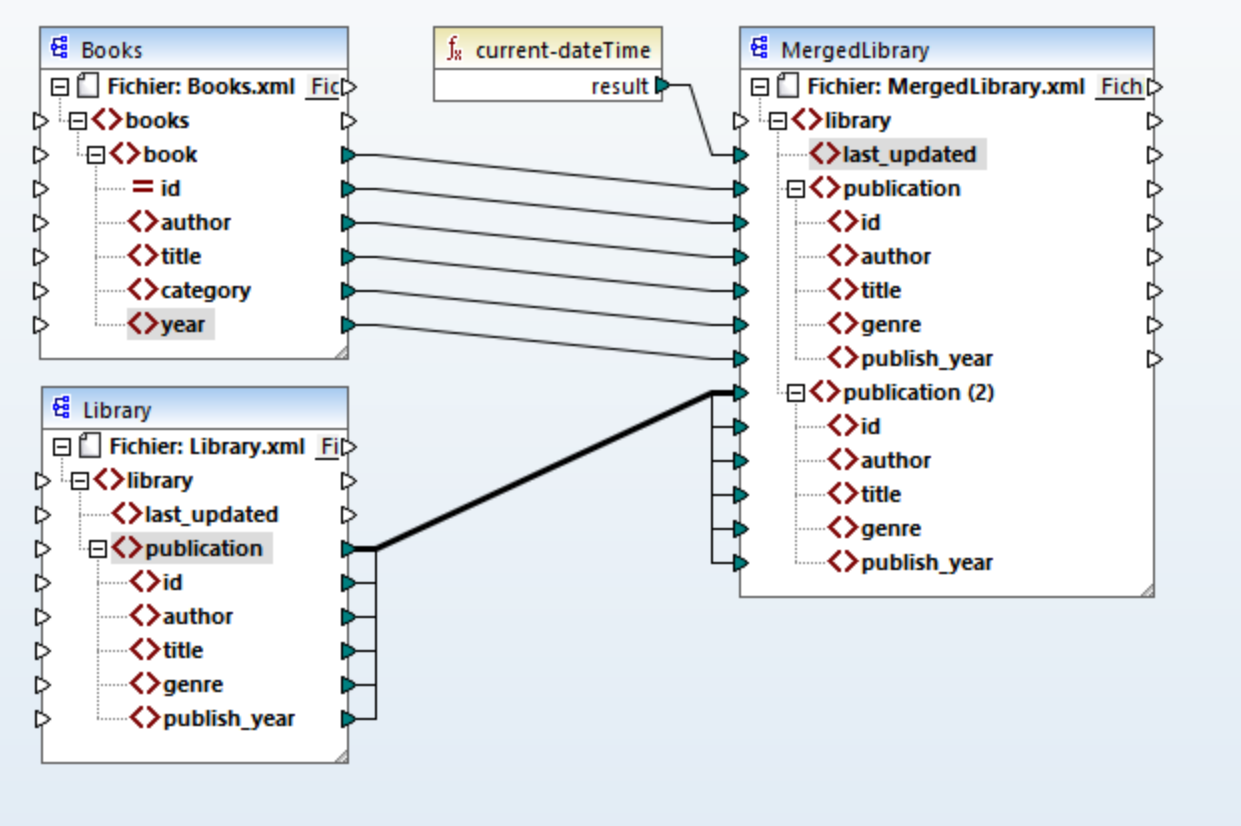

Dans le diagramme ci-dessus, les données sont tout d'abord fusionnées depuis deux fichiers source (**`Books.xml`** et **`Library.xml`**) dans un seul fichier cible appelé **`MergedLibrary.xml`**. Ensuite, les données sont transformées avec une fonction de filtre et sont passées au composant suivant appelé **`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 100. Avant d'effectuer un changement au mappage, assurez-vous d'enregistrer ce design sous un nouveau nom dans le dossier **`BasicTutorials`**.

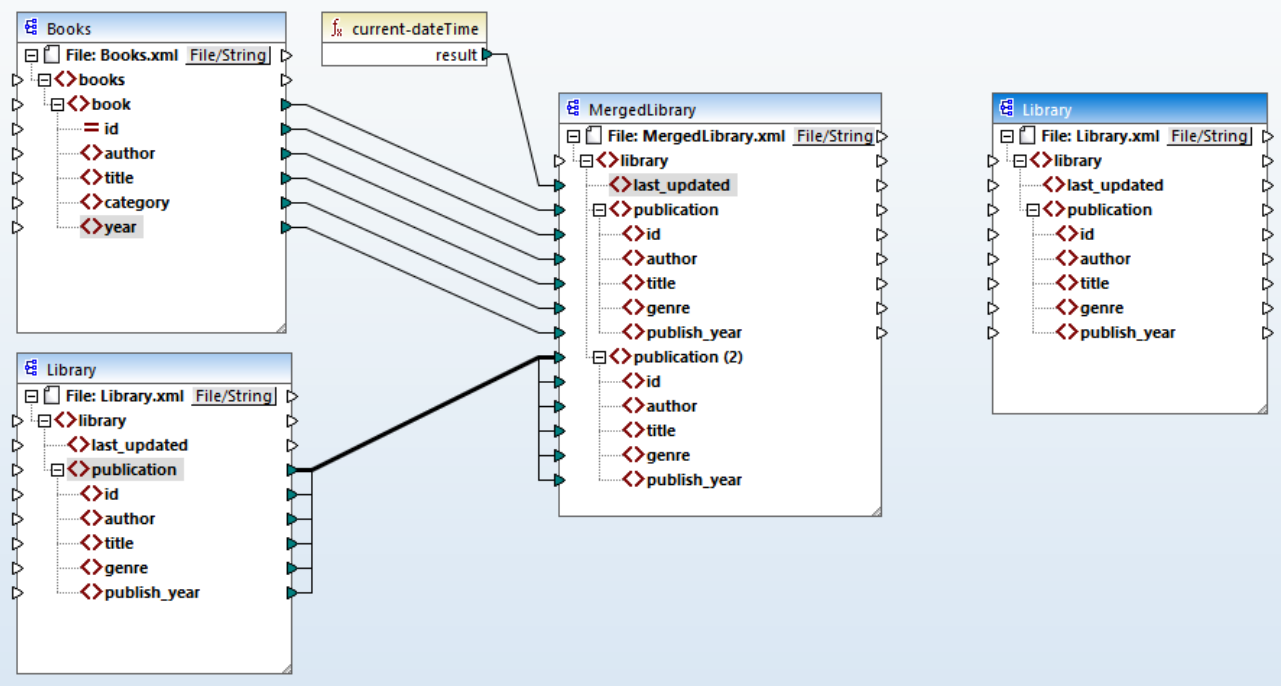



## 3.3.2 Configurer la deuxième cible

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`**.

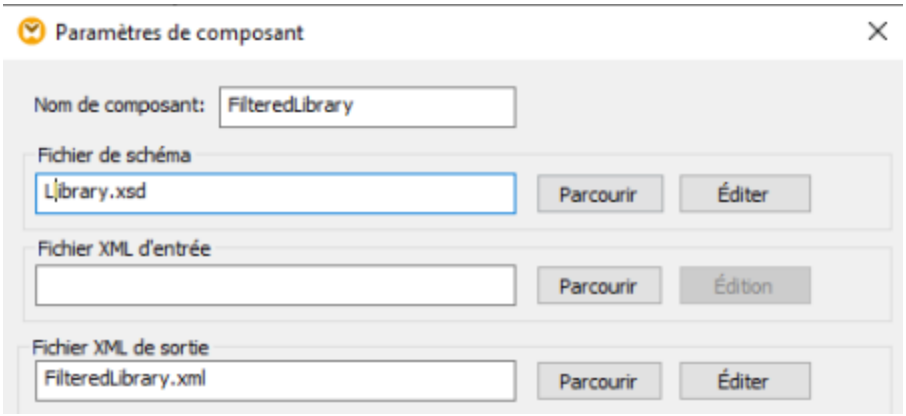
### Ajouter le deuxième composant cible

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 :



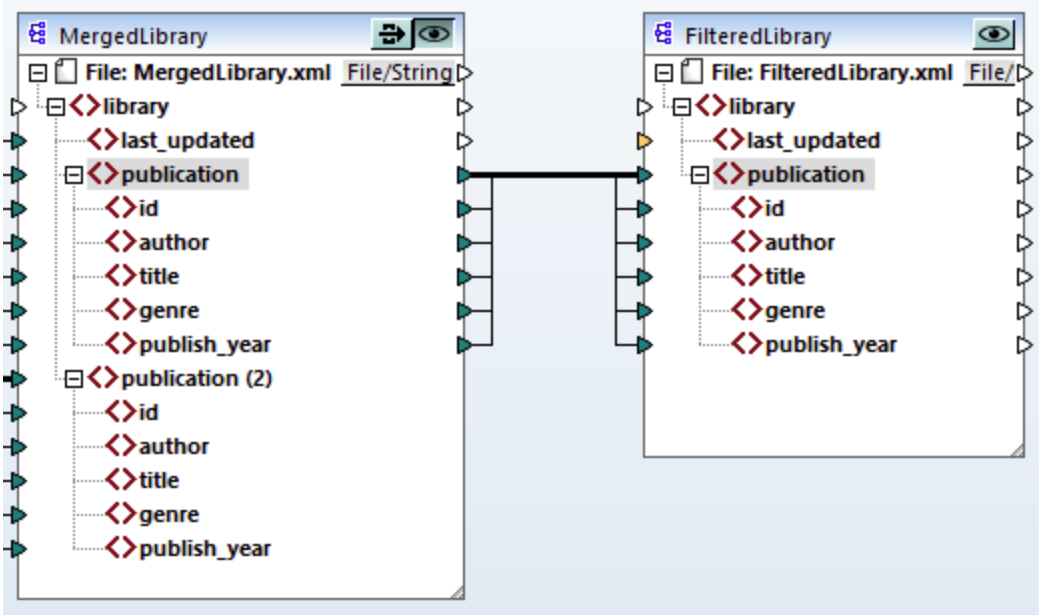

## Configurer le deuxième composant cible

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[124] :



## 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.

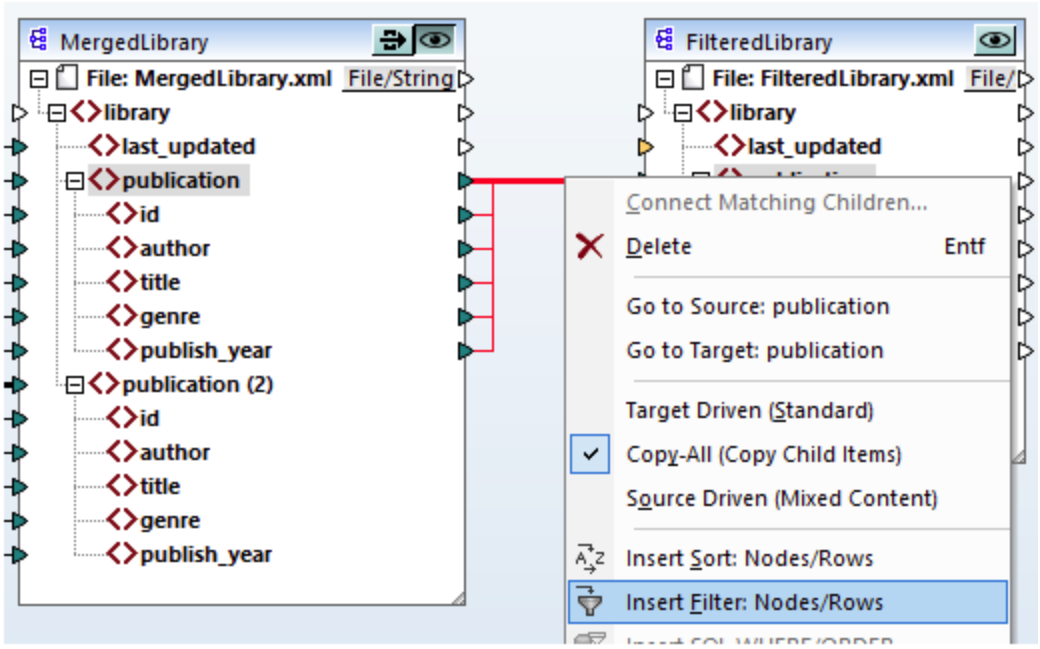



## 3.3.4 Filtrer les données

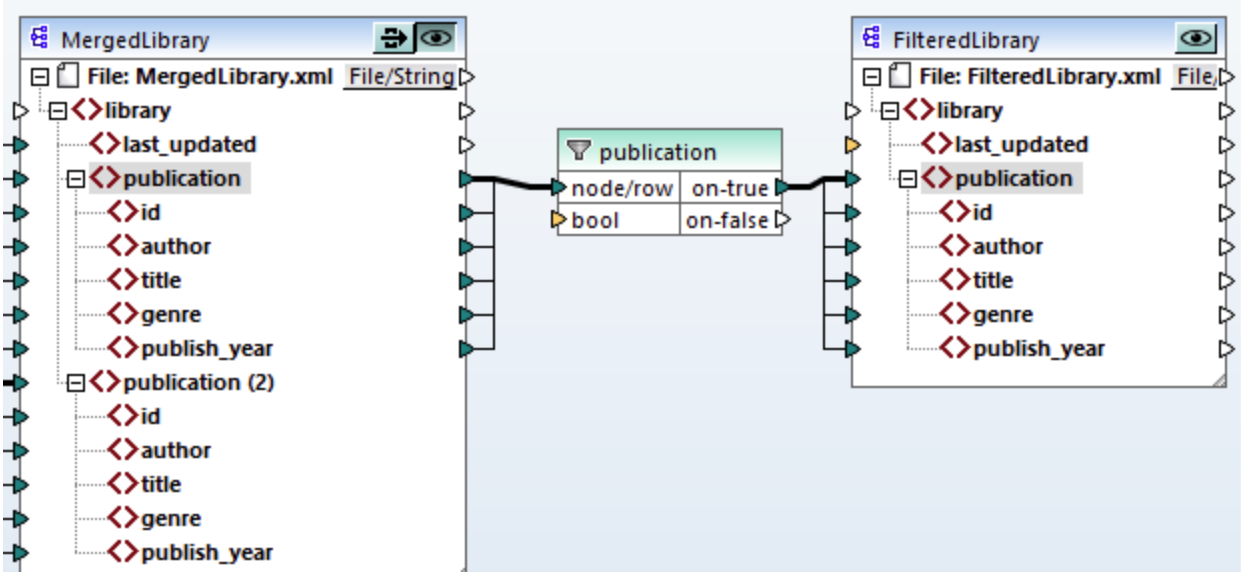
À cette étape, nous allons filtrer les données de `MergedLibrary` de telle façon que seuls les livres publiés après 1900 seront passés au composant `FilteredLibrary`. Nous allons utiliser le composant Filtre à cette fin.

### Ajouter un filtre

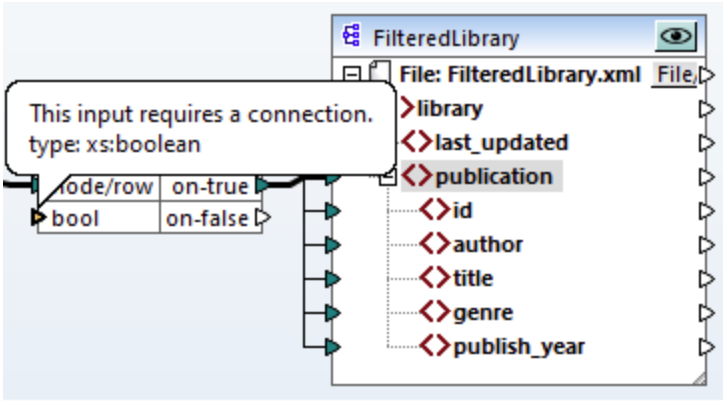
Pour ajouter un filtre, cliquez avec le bouton droit entre `MergedLibrary` et `FilteredLibrary` et sélectionnez **Insérer filtre : Nœuds/Lignes** depuis le menu contextuel(*capture d'écran ci-dessous*).




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



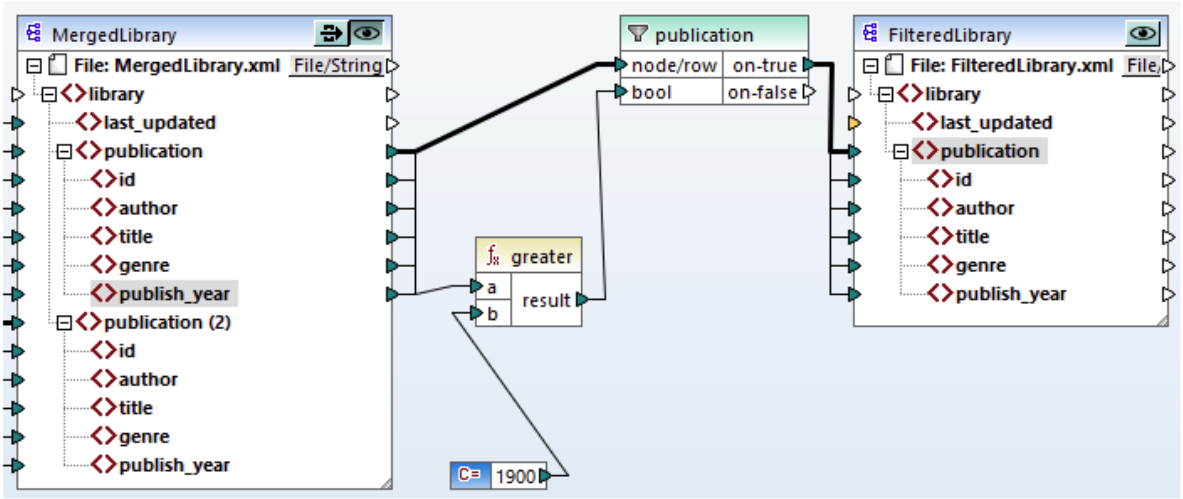

Dans la capture d'écran ci-dessus, le connecteur d'entrée `bool` est en surbrillance orange, ce qui signifie que cette entrée est obligatoire. Si vous passez sur un connecteur, vous pouvez voir si une entrée de type `xs:boolean` est requise (*voir la capture d'écran ci-dessous*). Pour voir les informations,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

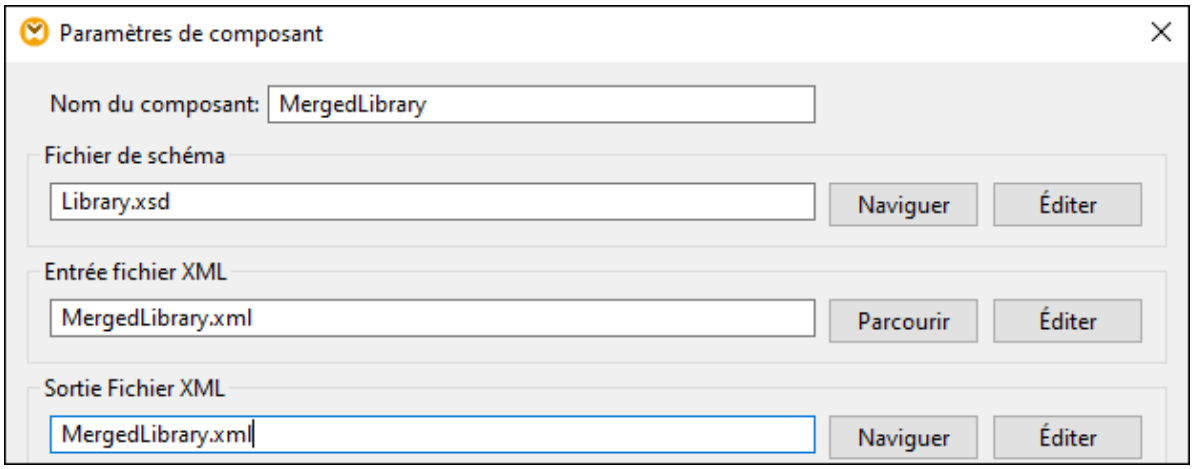
1. Désactivez la touche **Pass-though** du composant `MergedLibrary` si la touche a précédemment été activée.
2. Cliquez sur la touche **Aperçu** dans le composant `MergedLibrary` .
3. Ouvrez le volet Sortie.
4. Cliquez sur le bouton **Enregister fichier de sortie** dans la barre d'outils pour enregister 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**[1086]) 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.

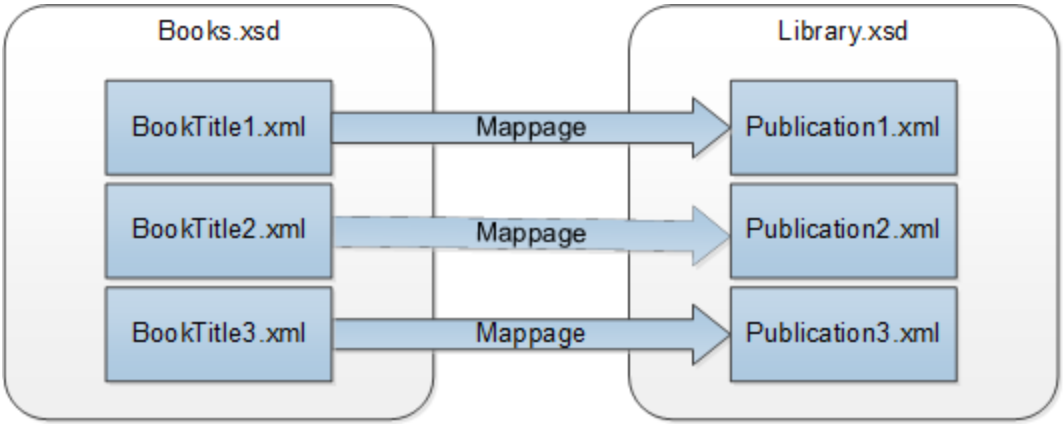
Vous avez désormais terminé de concevoir le mappage qui produit deux fichier de sortie. À votre convenance, le design de mappage est enregistré dans ce tutoriel en tant que **`Tut3_ChainedMapping.mfd`**.

# 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 :

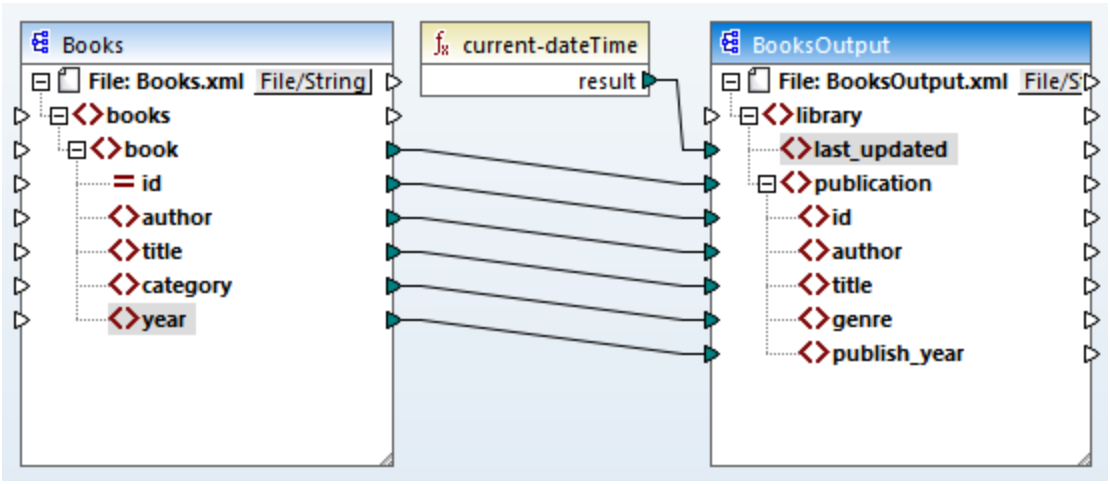
1. Pour lire des données depuis des fichiers XML multiples situé dans le même répertoire. Les fichiers sont basés dans le même schéma source.
2. Pour chaque fichier source XML, générez un nouveau fichier cible XML. Les fichiers cible seront basés dans un schéma cible.

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 89 (*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**

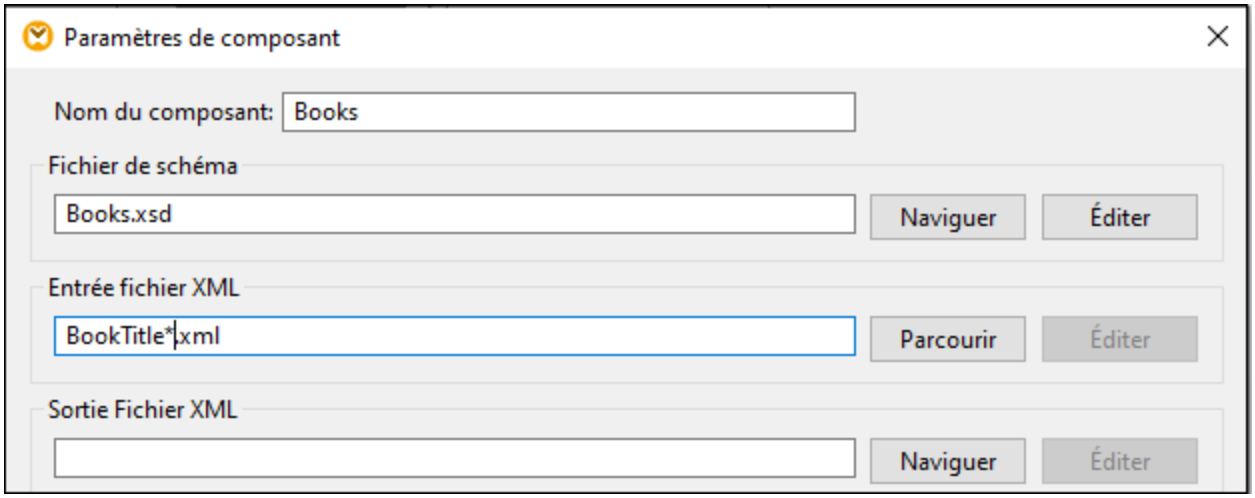
```
      <library>
         <publication>
            <id>3</id>
            <author>Herman Melville</author>
            <title>Moby Dick</title>
            <genre>Fiction</genre>
            <publish_year>1851</publish_year>
         </publication>
      </library>
```

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.

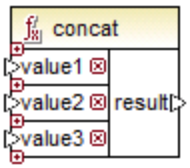



## 3.4.2 Configurer la Sortie, Partie 1

Dans cette étape, nous allons créer le nom de fichier de chaque fichier de sortie. Pour ce faire, nous utiliserons la fonction **concat** 629 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 :

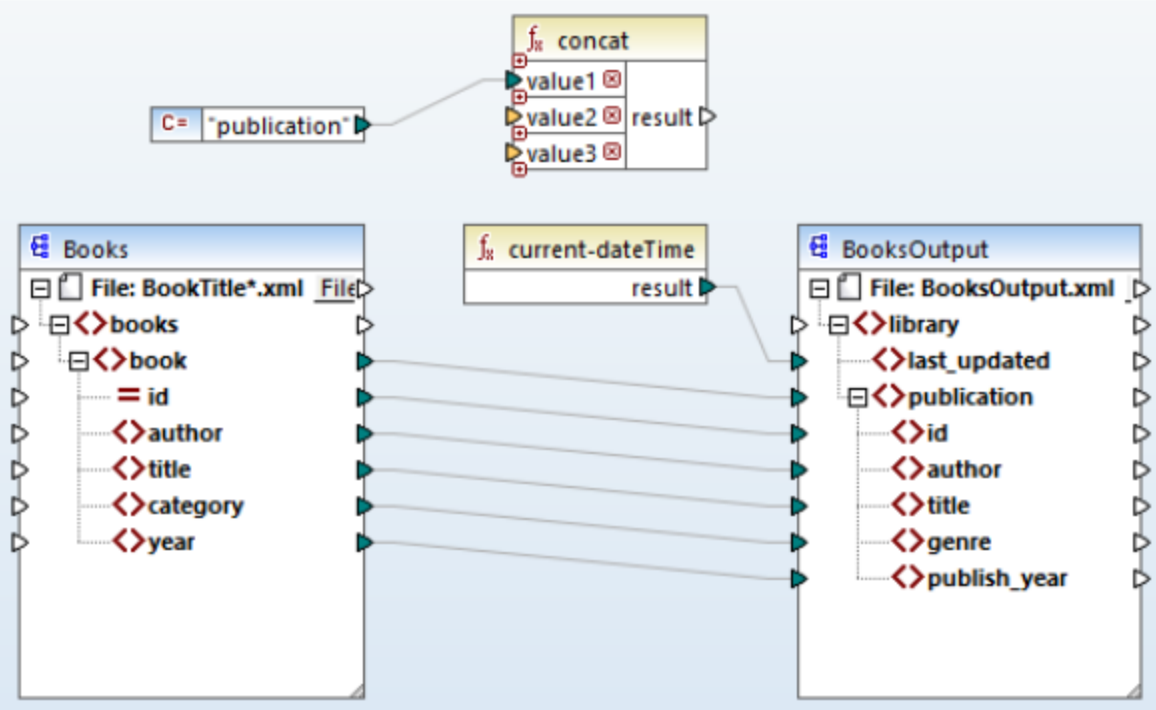
### Étape 1 : Ajouter la fonction de concaténation

Ajoutez 94 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.



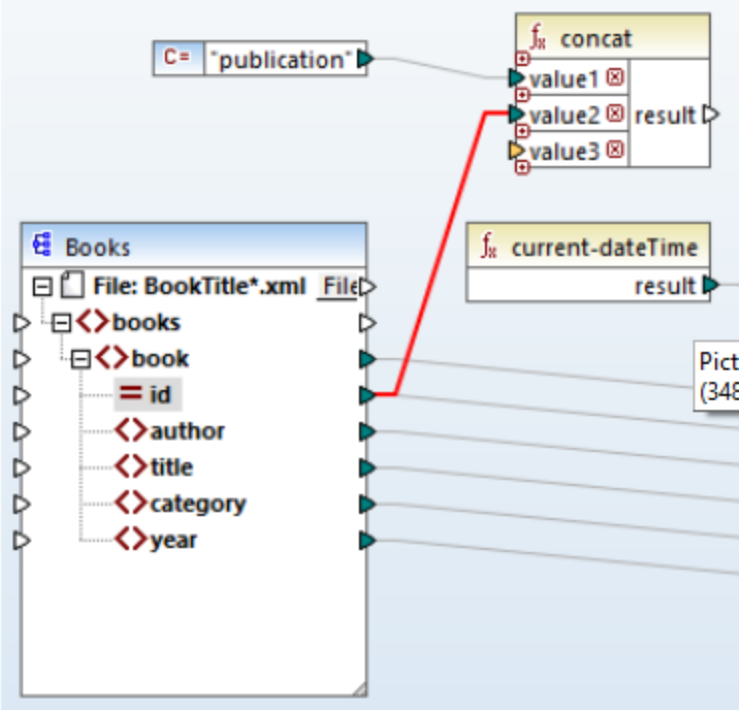

### Étape 2 : Insérer une constante

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

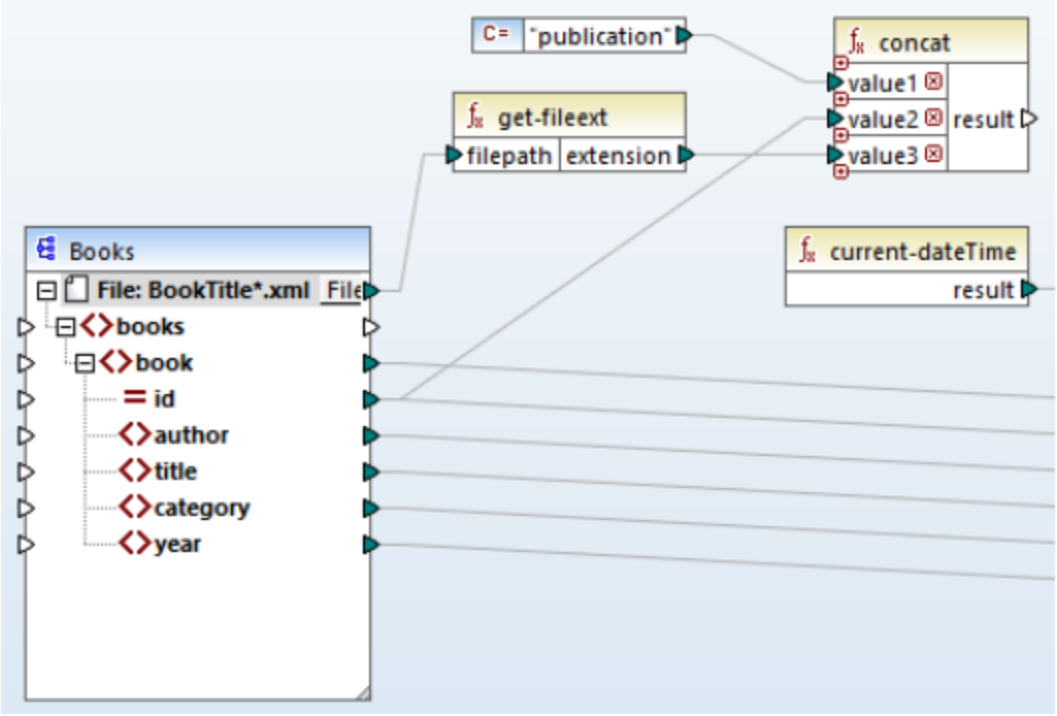
Connectez l'attribut `id` du composant source avec `value2` de la fonction **`concat`** (*capture d'écran ci-dessous*) L'attribut `id` du fichier XML source fournit une valeur d'identifiant unique pour chaque fichier. Cela sert à empêcher que les fichiers soient générés avec le même nom. La connexion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




## Étape 4 : Extraire l'extension de fichier

Ajouter **`get-fileext`** [572] 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.

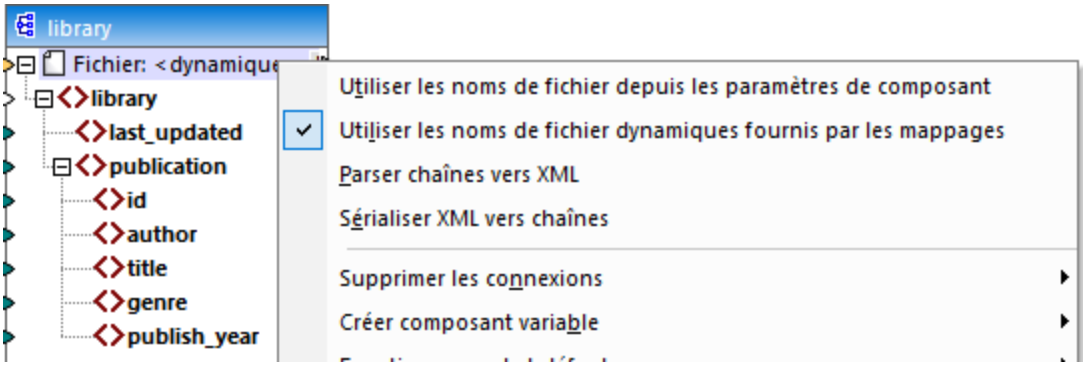



## 3.4.3 Configurer la Sortie, Partie 2

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 787.

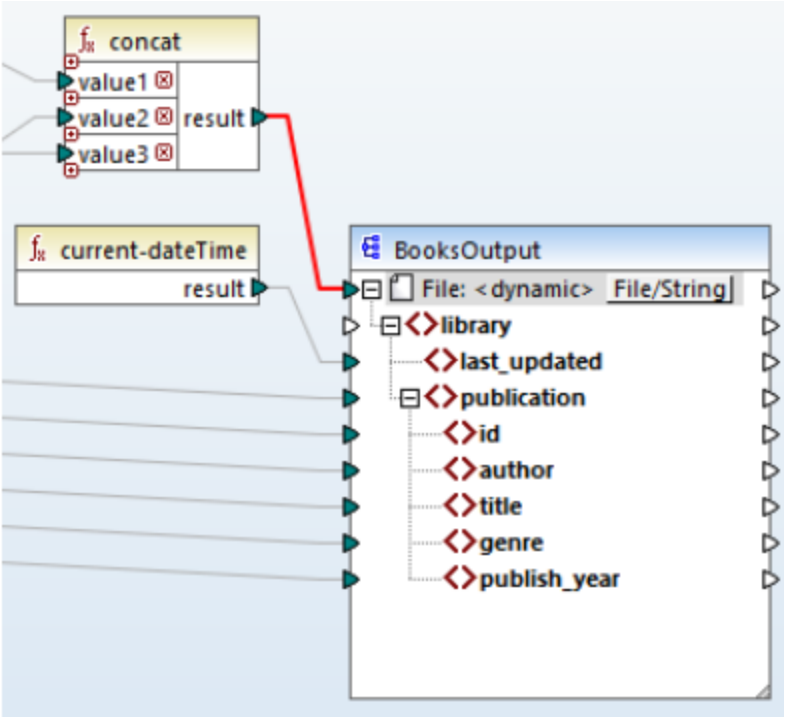
### Étape 1 : Définir noms de fichiers dynamiques

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*).



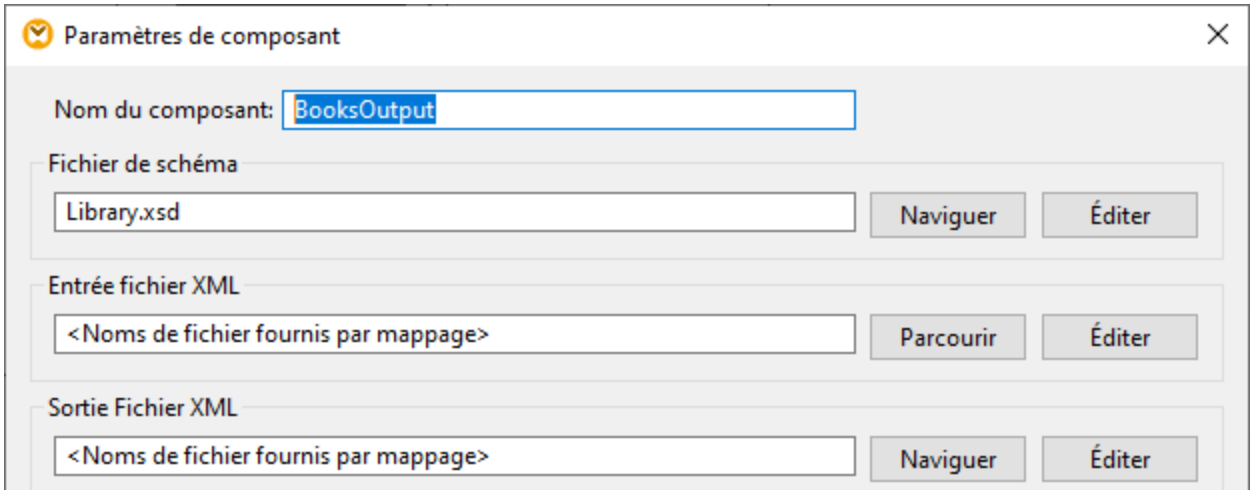

## Étape 2 : Connecter fonction concat et nœud dynamique

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



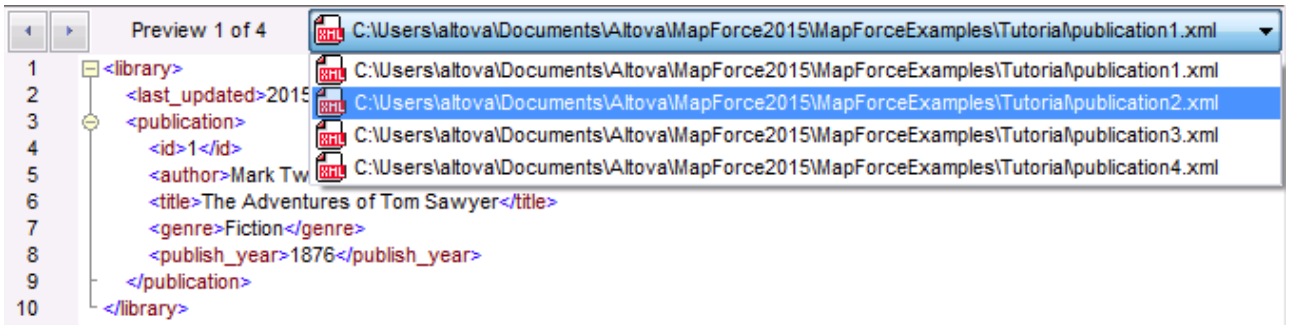

## Étape 3 : Vérifier paramètres de composant

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 123
- Bases de données 158
- Fichiers CSV et Texte 373

MapForce vous permet également de mapper les données vers et des fichiers binaires (données BLOB). Pour plus d'informations, voir les fonctions `lang | file` 665.

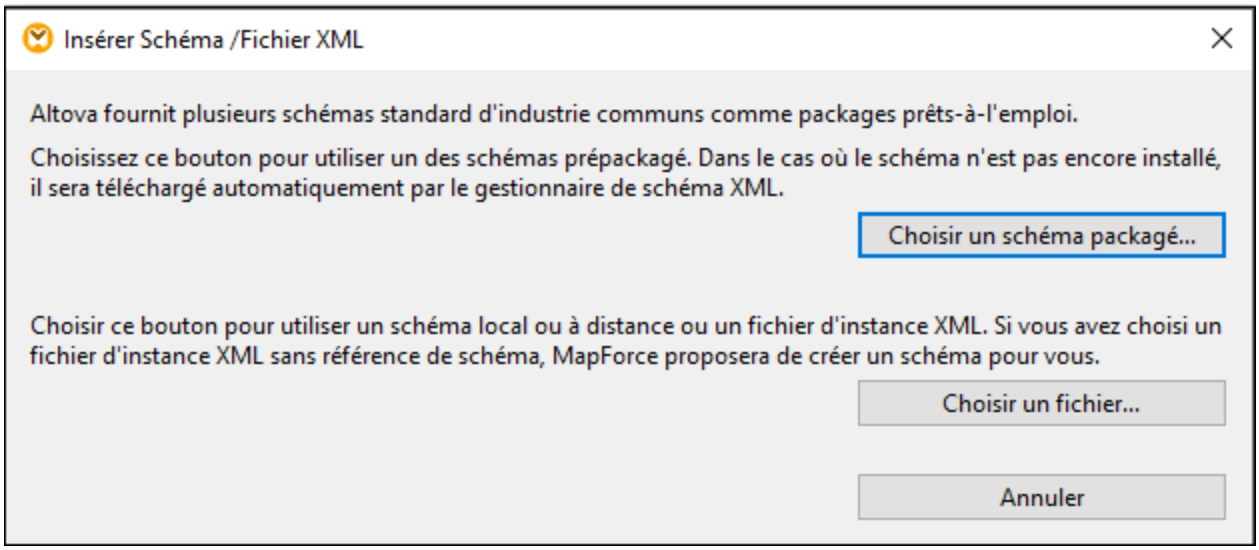
# 4.1 XML et schéma XML

**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[37] qui peuvent être utilisés comme sources de données et cibles. Pour des informations sur les scénarios de transformation des données de base, voir Tutoriels[88].

## 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[141] 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 455.

### Dans cette section

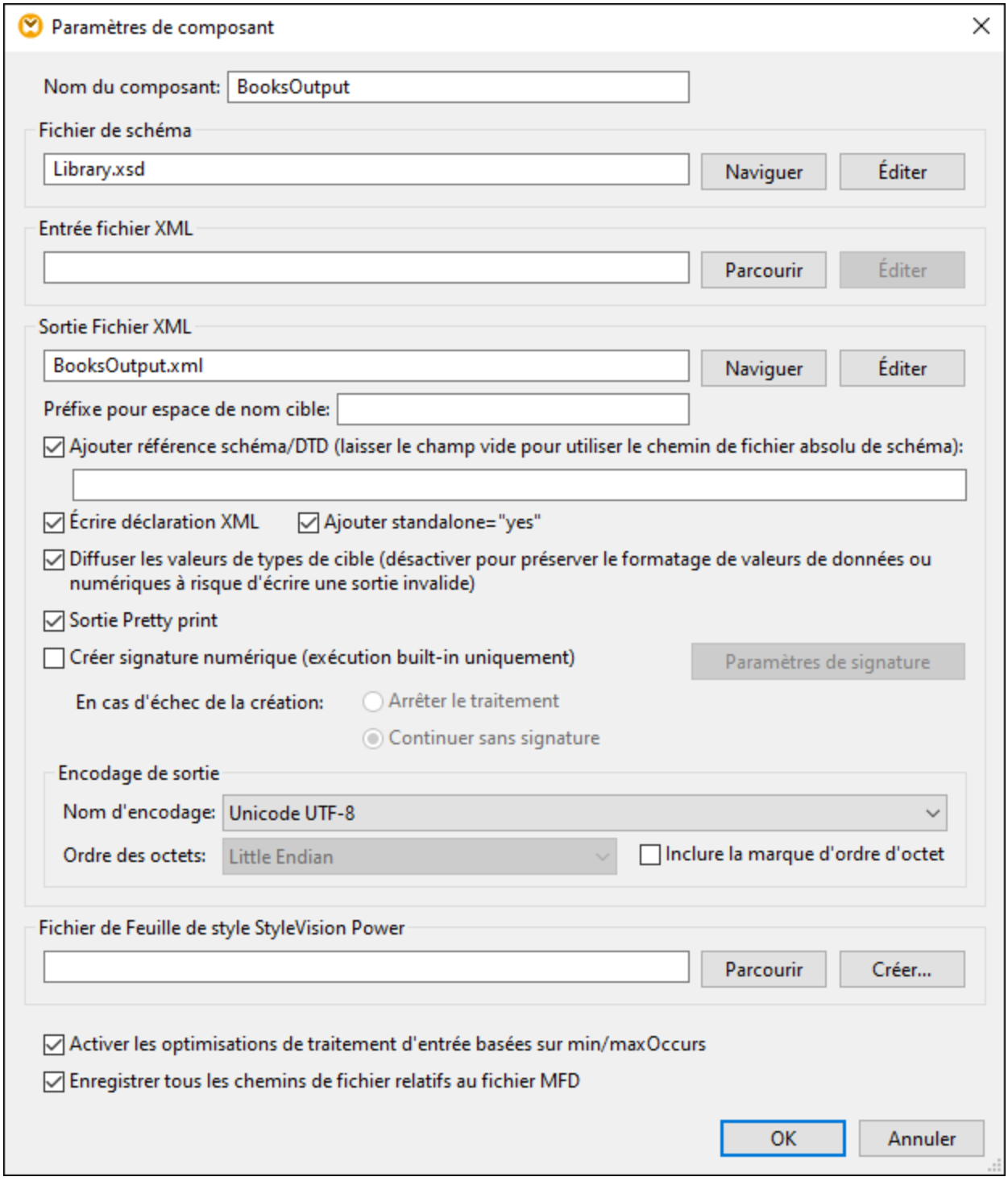
La section est organisée en rubriques suivantes :



## 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 22 et le code généré sont disponibles dans les éditions Professional et Enterprise. Pour plus d'information sur la génération de code, voir le Générateur de Code 935.
- 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 47.

## 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 47.

## 4.1.2 Types dérivés

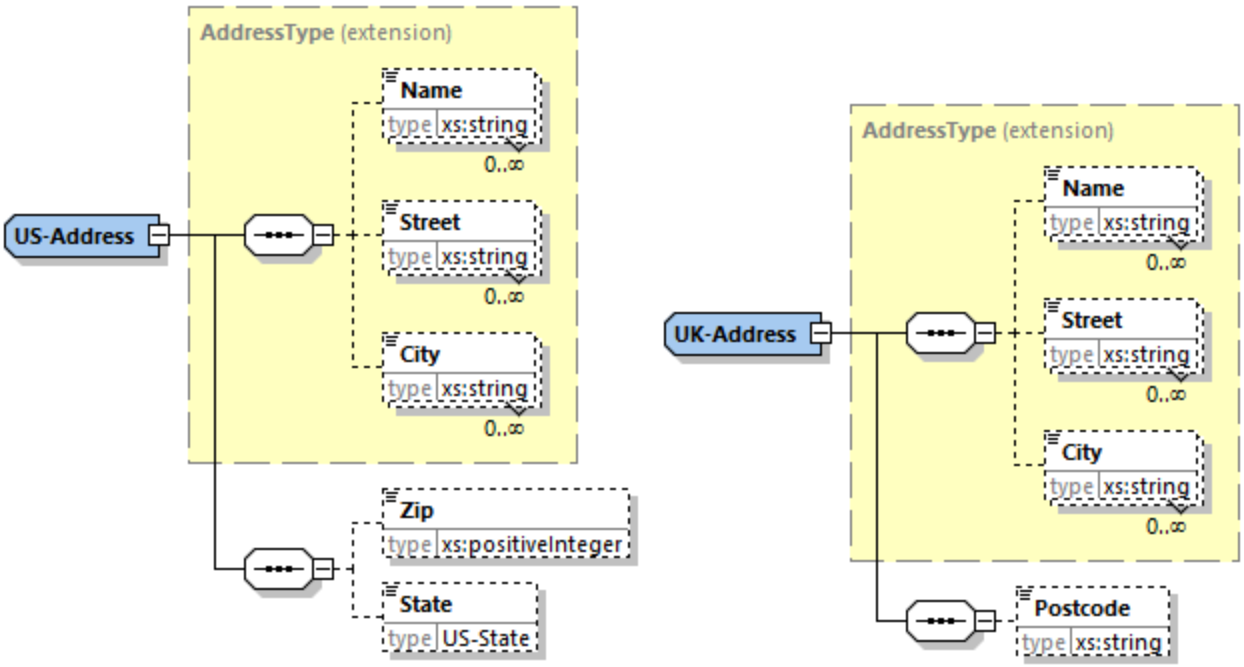
Cette rubrique explique comment utiliser les types dérivés dans les mappages. Les types dérivés sont définis dans la Spécification de schéma W3C XML (la section 2.5.2). Pour un bref aperçu des types dérivés et primitifs, voir la documentation Microsoft. Pour utiliser les types dérivés dans un mappage, 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.

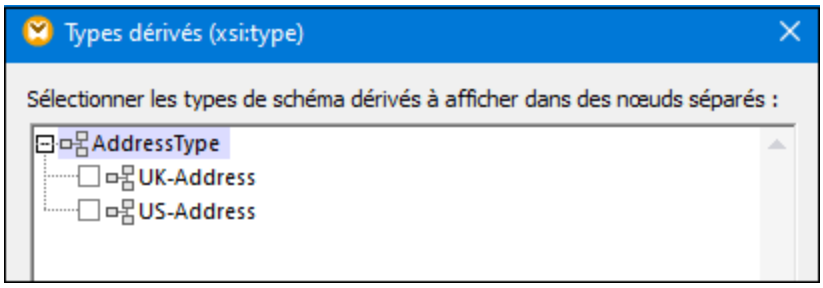



*Type dérivé dans un mappage*

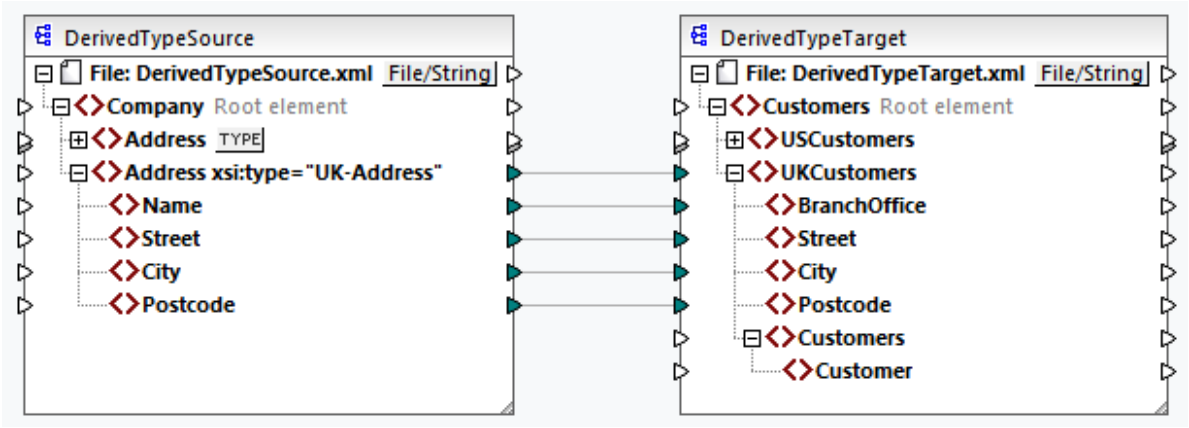
Les instructions ci-dessous montrent comment mapper les données depuis le type dérivé. Notre objectif est de mapper l'information sur les bureaux au UK vers l'élément `UKCustomers`. Les exemples de fichiers sont disponibles dans le dossier **`Tutoriel`**.

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[55].
- 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[55].
- 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[55].
- 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[60].

```
<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**[591] : Aide à vérifier explicitement si un élément source a un attribut `xsi:nil` définit à `true`.
- **substitute-missing**[625] : Substitue une valeur NULL dans l'élément source avec quelque chose de spécifique.
- **set-xsi-nil**[593] : 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**[594] : 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**[599] 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**).

## Valeurs NULL dans les composants de base de données

Cette sous-section affiche comment les valeurs NULL sont traitées dans les composants de base de données.

*Mapper les champs de base de données NULL dans les éléments NULL*
Les éléments cibles qui obtiennent les valeurs NULL depuis les champs de base de données ne sont pas créés dans la sortie automatiquement. Pour voir de tels éléments dans la sortie, vous devez (i) ajouter les attributs `nillable="true"` aux éléments cibles pertinents dans le fichier de schéma et (ii) utilisez la fonction **`substitute-missing-with-xsi-nil`** [594] dans le mappage. L'exemple ci-dessous montre comment gérer les valeurs NULL dans les mappages avec un composant de base de données source.

*Table d'application dans le volet DB Query.*
L'exemple de mappage est situé dans le chemin suivant : **`Tutorial\DBNullToXML.mfd`**. Pour notre exemple, nous avons choisi uniquement une table (`Application`) depuis la base de données `Accounts` (*voir ci-dessous*).

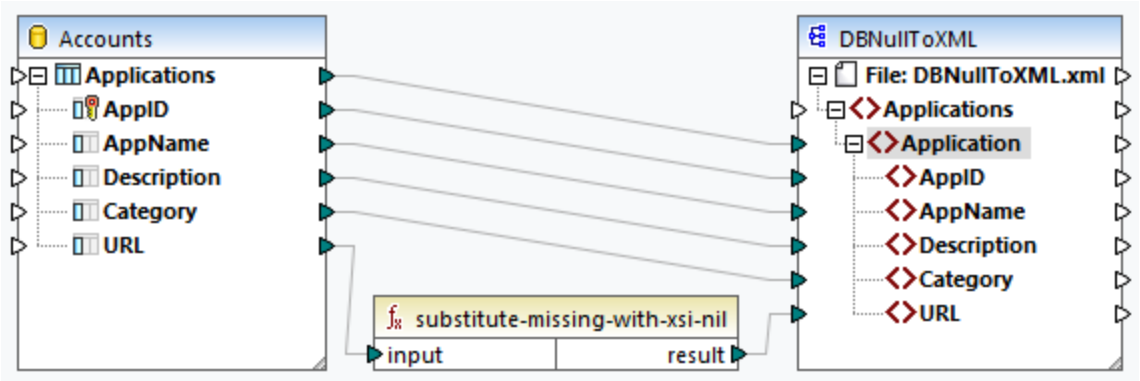
| | AppID | AppName | Description | Category | URL |
|---|---|---|---|---|---|
| 1 | 1 | Altova MapForce | Best data mapping tool! | IDE | https://www.altova.com/mapforce |
| 2 | 2 | Notepad | *[NULL]* | *[NULL]* | *[NULL]* |

Pour voir la table `Application`, suivez les étapes suivantes :

- Ouvrez le volet **DB Query**.
- Sélectionnez la base de données `Accounts` pour voir sa structure dans le Navigateur de la base de données.
- Cliquez avec la touche de droite dans la table `Application` et cliquez sur **Afficher dans l'Éditeur SQL | SELECT**.
- Cliquez sur le bouton  (**Exécuter Requête**). La table `Application` apparaîtra dans l'onglet **Résultats**.

Pour en savoir plus sur les requêtes de bases de données, voir Volet requête BD [300].

*Mappage*
La table `Application` ci-dessus montre que le deuxième enregistrement a des valeurs NULL dans les champs `Description`, `Catégorie` et `URL`. À des fins d'illustration, nous mapperons presque la totalité des colonnes directement vers les éléments cible correspondants. Pour la colonne `URL`, nous utiliserons la fonction **`substitute-missing-with-xsi-nil`** pour que les valeurs NULL dans l'élément cible ait un attribut `xsi:nil="true"` (*voir le mappage ci-dessous*).



*Sortie*
Le fichier de sortie ci-dessous affiche que le premier enregistrement de la table a été entièrement écrit à la sortie, tandis que le deuxième enregistrement a été écrit uniquement partiellement. Les valeurs de base de données NULL sont absentes de la sortie, à l'exception de l'élément `URL`. Depuis que l'élément `URL` est doté de l'attribut `nillable="true"` dans le fichier de schéma et nous avons utilisé la fonction **`substitute-missing-with-xsi-nil`**, l'élément `URL` est désormais doté de l'attribut `xsi:nil="true"` dans la sortie (*en surbrillance jaune*).

```
<Application>
  <AppID>1</AppID>
  <AppName>Altova MapForce</AppName>
  <Description>Best data mapping tool!</Description>
  <Category>IDE</Category>
  <URL>https://www.altova.com/mapforce</URL>
</Application>
<Application>
  <AppID>2</AppID>
  <AppName>Notepad</AppName>
  <URL xsi:nil="true"/>
</Application>
```

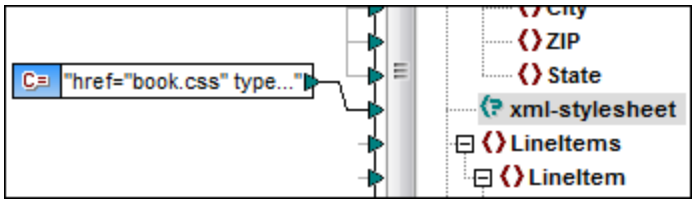
*Mapper les éléments NULL aux champs de base de données NULL*
Lorsque vous mappez un élément NULL XML à la colonne de la base de données, MapForce écrit la valeur NULL à la colonne de base de données correspondante. Vous pouvez aussi utiliser la fonction **`set-null`**[644] si vous voulez définir un champ de base de données à NULL. Pour en savoir plus sur les fonctions liées à la base de données, voir Voir la bibliothèque BD[643].

## 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[60]. 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 45.

*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

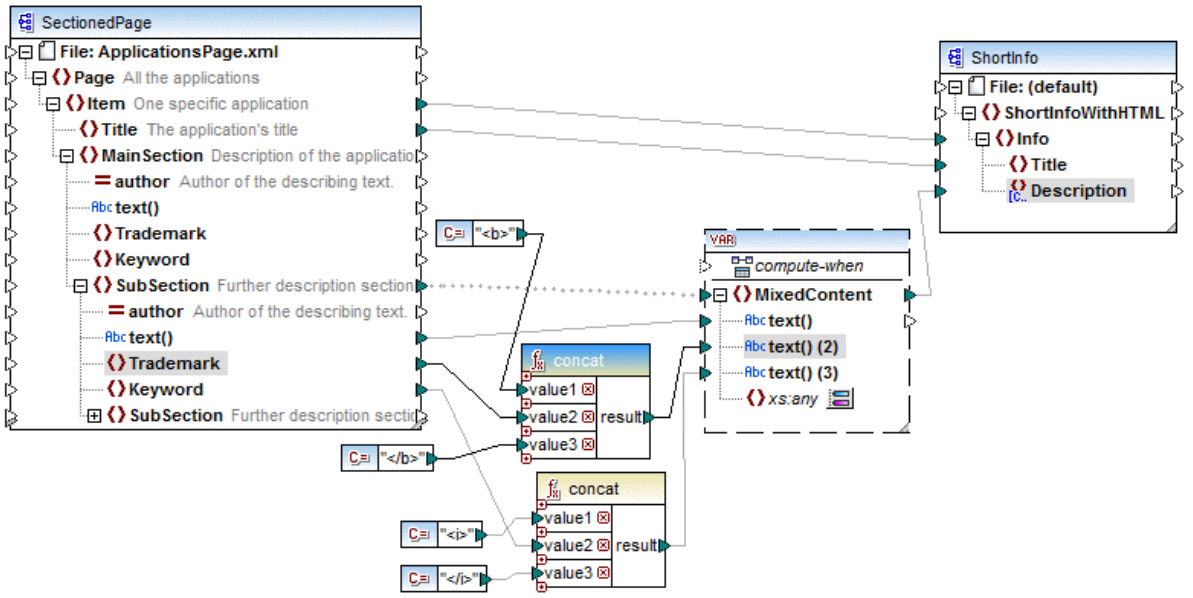
Les sections CDATA sont utilisées pour échapper des blocs de texte contenant des caractères qui devraient normalement être interprétés en tant que balise. Pour plus d'informations sur les sections de CDATA, voir la Spécification W3C. Les nœuds cible recevant des données comme sections CDATA peuvent être 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 55.
- 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.



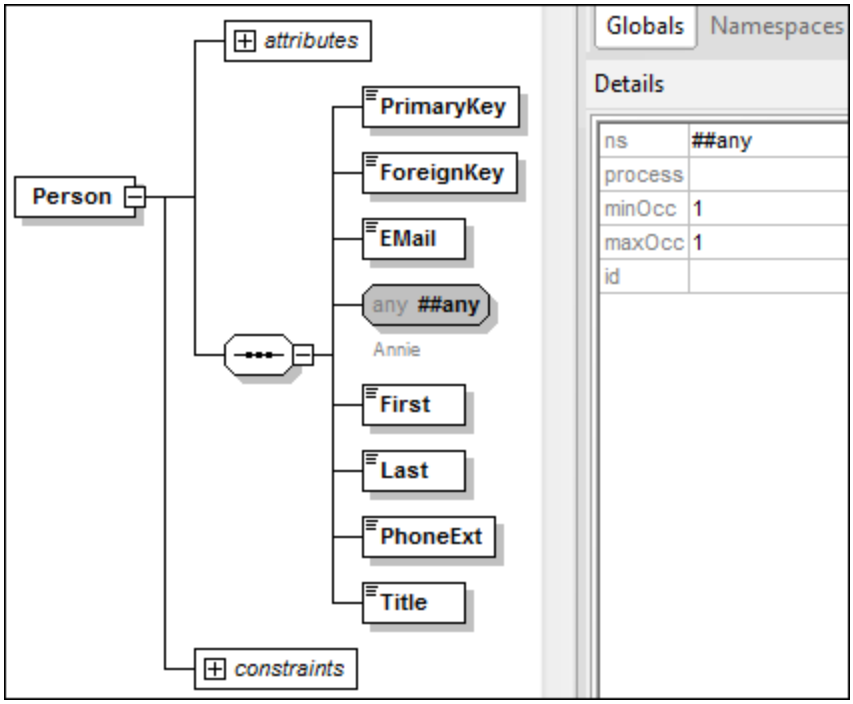

*Sortie*
Cliquez dans le volet **Sortie** pour consulter la section CDATA dans le nœud `Description` (*voir la capture d'écran ci-dessous*).

```
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>
```

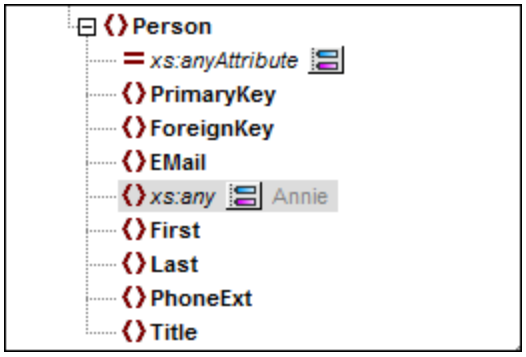
## 4.1.6 Caractères génériques - xs:any / xs:anyAttribute

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.

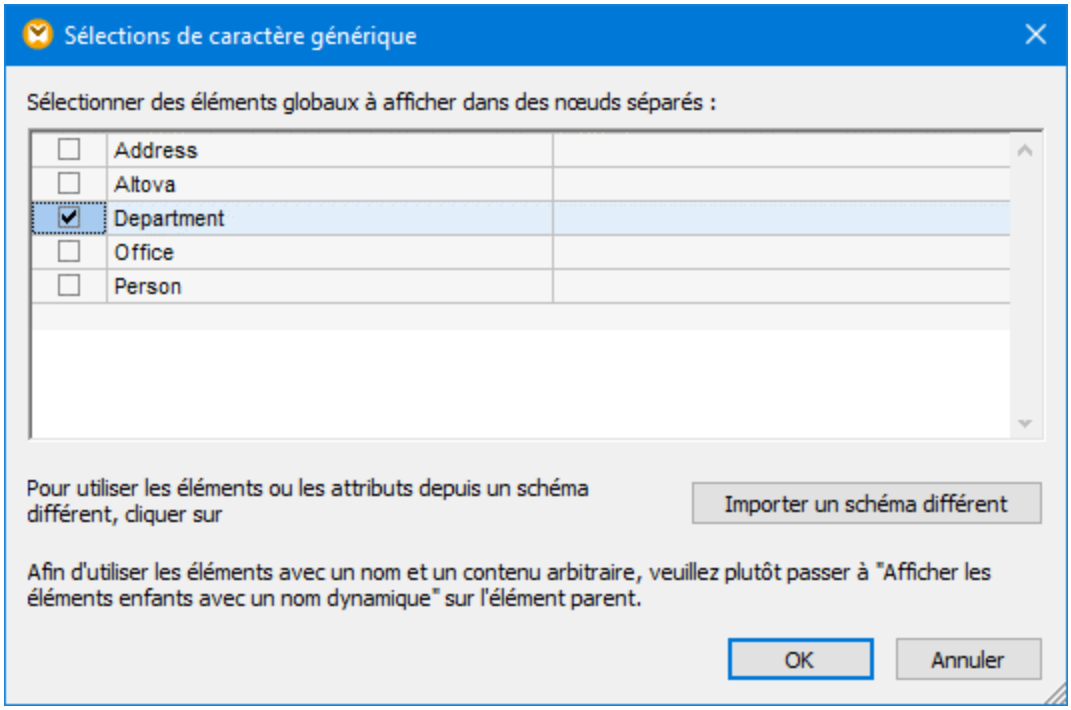
*Caractères génériques dans la définition de schéma*
La capture d'écran ci-dessous affiche qu'un élément `xs:any` a été défini comme élément enfant de l'élément `Person` (*mode Schéma dans Altova XMLSpy*).



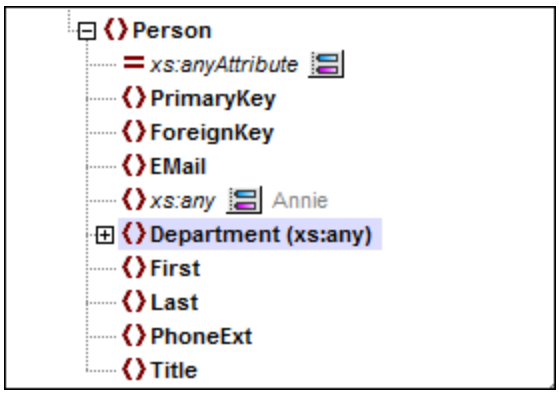

*Caractères génériques dans MapForce.*
Lorsqu'un caractère générique est défini pour un élément et/ou attribut, ce nœud aura un bouton (**Changer Sélection**) à côté de celui-lui (*voir la capture d'écran ci-dessous*).




*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 41) 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 766.

## 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 .

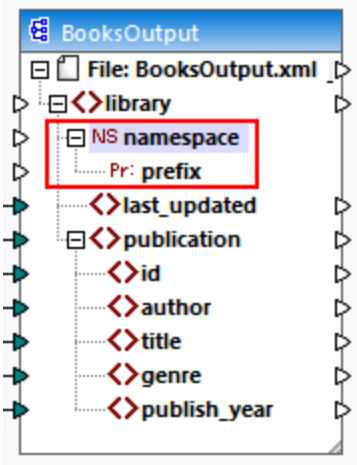
Pour comprendre comment les espaces de noms personnalisés fonctionnent, suivez les instructions dans la sous-section ci-dessous.

## Déclarer l'espace de nom manuellement

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

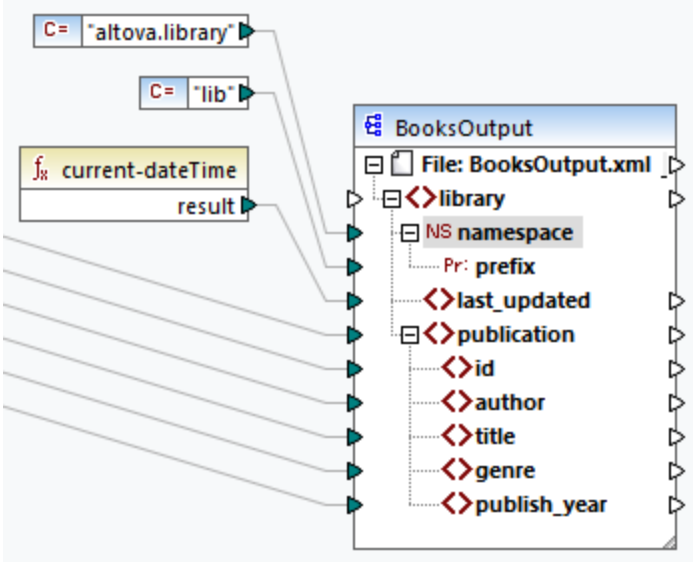
*Ajouter un espace de nom*

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">
...
```

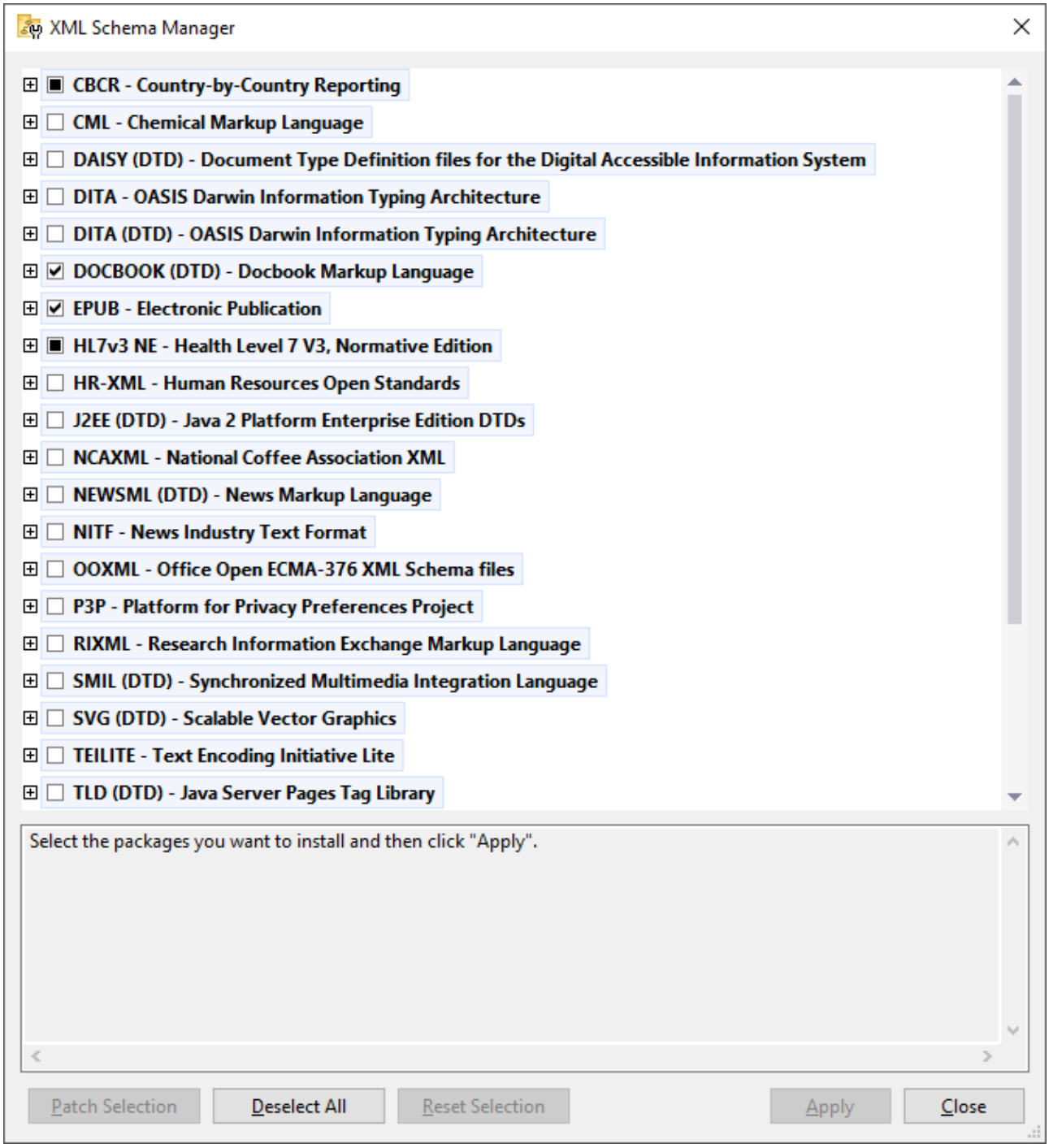
Si vous souhaitez créer des préfixes pour les noms d'attribut, par exemple `<number prod:id="prod557">557</number>`, vous pouvez le faire soit en utilisant l'accès dynamique aux attributs d'un nœud (voir Mapper des noms de nœud 766), soit en éditant le schéma de manière à ce qu'il ait un attribut `prod:id` 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` [153], 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`[156] 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`[155], et (ii) exécutez la commande `initialize`[152]. (En alternative, exécutez la commande **`reset`** avec l'option **`--i`**.)

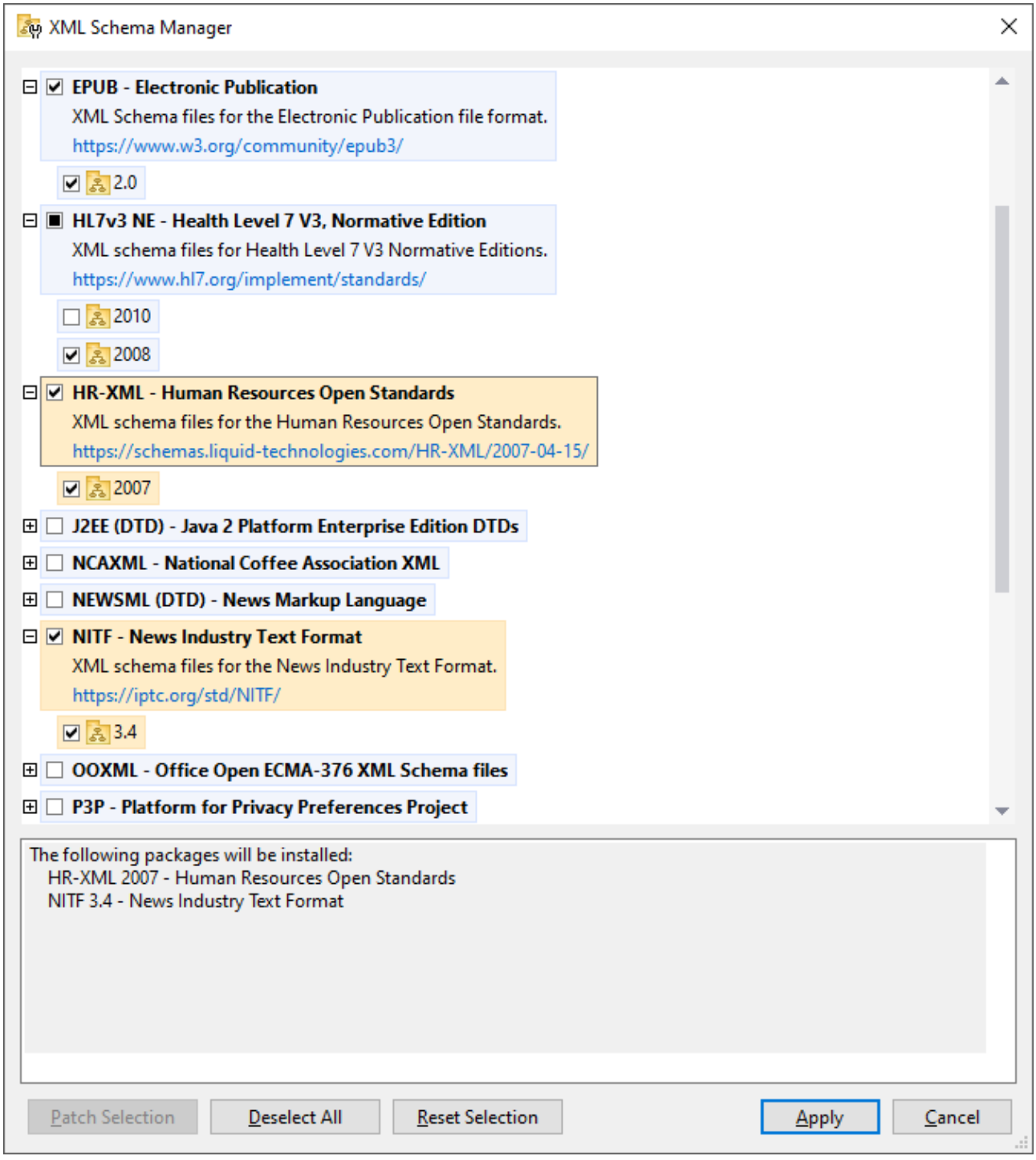
## 4.1.8.1 Exécuter Schema Manager

### 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[151].

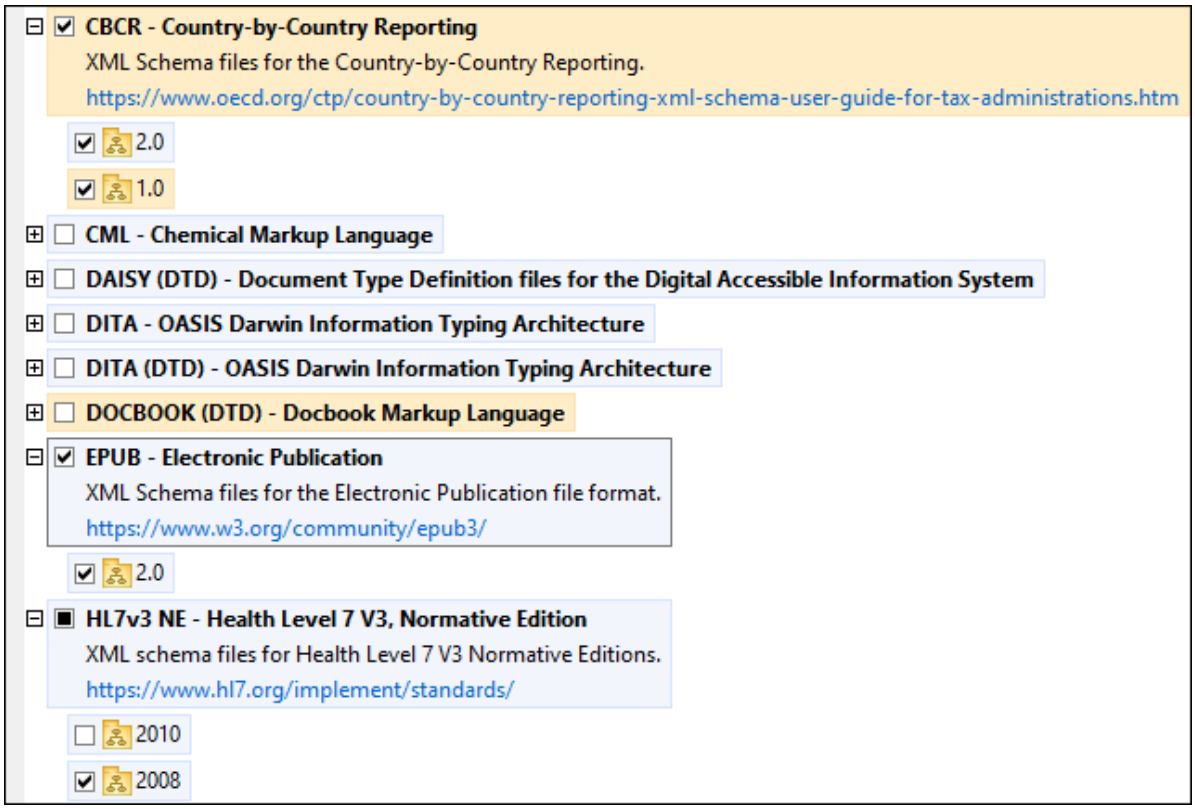
Pour afficher l'aide pour la commande, exécutez l'étape suivante :

- *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` 154 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*[147].) 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`[154] . Cela liste tout schéma lorsque des mises à niveau sont disponibles.
2. Exécutez la commande `upgrade`[157] 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`[153] :

```
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[154].) Vous pouvez saisir autant de schémas que vous le souhaitez. Pour plus de détails, voir la description de la commande install[153].

**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[155] :

```
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[154].) Vous pouvez saisir autant de schémas que vous le souhaitez. Pour plus de détails, voir la description de la commande désinstaller[155].

**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`[155].

Une fois avoir exécuté cette commande, vous devrez exécuter la commande `initialize`[152], pour pouvoir recréer le répertoire de mise sous cache. En alternative, exécutez la commande `reset`[155] avec l'option `-i`.

Notez que `reset -i`[155] restaure l'installation originale du produit, il est recommandé d'exécuter la commande `update`[156] après avoir réalisé la réinitialisation. En alternative, exécutez la commande `reset`[155] 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`154.)
- 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` 154. 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 ?* 141.

*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`[152], 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`[156] 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` 154 . 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 ?* 141.

*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` 155 et `initialize` 152.

## 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`[154].

**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. |

# 4.2 Bases de données

**Site web d'Altova :** Mappage de base de données

MapForce vous permet d'utiliser les bases de données comme sources et cibles de données.

La table ci-dessous énumère toutes les bases de données prises en charge. Si votre application Altova est une version 64-bit, assurez-vous que vous avez accès aux pilotes de base de données 64-bit nécessaires pour la base de données spécifique à laquelle vous tentez de vous connecter.

| Base de données | Notes |
|---|---|
| Firebird 2.x, 3.x, 4.x | |
| IBM DB2 DB2 8.x, 9.x, 10.x, 11.x | |
| IBM Db2 pour i6.x, 7.4, 7.5 | Les fichiers logiques sont pris en charge et présentés en tant qu'affichage. |
| IBM Informix 11.70 et ultérieur | Informix supporte les connexions via ADO, JDBC et ODBC. L'implémentation ne supporte pas les types de données d'objet importants dans un des langages de génération de code. MapForce générera un message d'erreur (pendant la génération de code) si un de ces type de données est utilisé. |
| MariaDB 10 et ultérieur | |
| Microsoft Access 2003 et ultérieur | Au moment de la rédaction de ce manuel (début septembre 2019), il n'a y pas de Microsoft Access Runtime disponible pour Access 2019. Vous pouvez vous connecter à une base de données Access 2019 depuis les produits Altova uniquement si Microsoft Access 2016 Runtime est installé et uniquement si la base de données n'utilise pas le type de données « Large Number ». |
| Microsoft Azure SQL Database | SQL Server 2016 codebase |
| Microsoft SQL Server 2005 et ultérieur<br>Microsoft SQL Server on Linux | |
| MySQL 5 et ultérieur | |
| Oracle 9i et ultérieur | |
| PostgreSQL 8 et ultérieur | Les connexions PostgreSQL sont prises en charge aussi bien en tant que connexions natives qu'en tant que connexions sur base pilote par le biais d'interfaces (pilotes) comme ODBC ou JDBC. Les connexions natives ne nécessitent pas de pilotes. |
| Progress OpenEdge 11.6 | |
| SQLite 3.x | Les connexions SQLite sont prises en charge en tant que connexions directes natives du fichier de base de données |

| Base de données | Notes |
|---|---|
| | SQLite. Aucun pilote séparé n'est nécessaire. |
| Sybase ASE 15, 16 | |
| Teradata 16 | Les connexions sont prises en charge avec ADO.NET, JDBC et ODBC.<br><br>Lorsqu'un mappage insère des données dans une table de base de données, les champs d'identité générés sur base de données ne sont pas pris en charge. |

## Mappages de base de données dans des environnements d'exécution variés

Lorsque vous générez un code de programme[70] depuis un mappage compilez un mappage vers les fichiers d'exécution MapForce Server[864], ou déployez un mappage vers FlowForce Server[867], les détails de connexion de la base de données enregistrés avec les fichiers générés sont adaptés aux pilotes pris en charge pour l'environnement cible choisi, (*comme indiqué dans la table ci-dessous*). Par exemple, si le langage de transformation de mappage est défini sur Java, les connexions ADO sont converties en JDBC lorsque le code Java est généré depuis le mappage.

Lorsqu'un mappage est exécuté dans un environnement autre que celui de MapForce, vous aurez besoin de vous assurer que les détails de connexion de la base de données sont utiles sur l'appareil qui exécute le mappage (par ex., vous allez avoir besoin de vérifier sir le pilote de base de données est installé, le chemin de base de données correct, le serveur de base de données accessible, etc.).

Certains types de connexion de base de données ne sont pas pris en charge dans certains environnements cible, comme indiqué dans la table ci-dessous.

| Type de connexion /Environnement d'exécution | C# | C++ | Java | MapForce Server sur Windows | MapForce Server sur Linux/Mac |
|---|---|---|---|---|---|
| *ADO* | Pont ADO | Tel quel | Converti en JDBC | Tel quel | Converti en JDBC |
| *ADO.NET* | Tel quel | Défini par l'utilisateur | Converti en JDBC | Tel quel | Converti en JDBC |
| *JDBC* | Défini par l'utilisateur | Défini par l'utilisateur | Tel quel | Tel quel | Tel quel |
| *ODBC* | Pont ODBC | Pont ODBC | Converti en JDBC | Tel quel | Converti en JDBC |
| *Native PostgreSQL* | Non pris en charge | Non pris en charge | Non pris en charge | Tel quel | Tel quel |

| Type de connexion /Environnement d'exécution | C# | C++ | Java | MapForce Server sur Windows | MapForce Server sur Linux/Mac |
|---|---|---|---|---|---|
| *Native SQLite* | Non pris en charge | Non pris en charge | Non pris en charge | Tel quel | Tel quel |

Légende de la table :

- *Tel quel* signifie que le type de connexion de base de données (Par ex., JDBC) reste tel que défini dans MapForce.
- *Converti en JDBC* signifie que la connexion de base de données sera convertie en une URL de connexion de base de données semblable à JDBC.
- *Pont ADO* ou *Pont ODBC* signifie que le string de connexion reste tel que défini dans MapForce, mais que le code généré utilisera une classe appropriée qui agit comme un pont ADO (ou un pont ODBC respectivement), par exemple, `System.Data.OleDb.OleDbConnection` ou `System.Data.Odbc.OdbcConnection`.
- *Défini par l'utilisateur* signifie que, afin que la connexion puisse fonctionner dans le code généré, vous devrez saisir manuellement les détails de connexion dans le dialogue Paramètres de composant de base de données 256.

*Empêcher des problèmes possibles avec les connexions JDBC dans l'environnement Java*
Si le mappage se connecte à une base de données par le biais de JDBC, s'assurer que les pilotes JDBC utilisés par le mappage sont installés sur votre système. Pour consulter les paramètres JDBC actuels, les paramètres de tout composant de base de données dans MapForce, double-cliquez dans l'en-tête du composant de la base de données. Ceci ouvrira le dialogue **Paramètres de composant**. Pour plus d'informations, voir Paramètres de composant de base de données 256 et Créer une connexion JDBC 185.

Si le mappage utilise une connexion de base de données non-JDBC, la connexion peut être convertie en JDBC pendant la génération de code Java, pour proposer une compatibilité dans un environnement Java. Pour les détails, voir l' table ci-dessus.

Si vous effectuez un test de l'application Java générée, le pilote JDBC peut devoir être ajouté en tant qu'entrée classpath dans l'exécution actuelle de la configuration. Autrement, exécuter l'application pourrait résulter en une erreur semblable comme suit : `java.lang.ClassNotFoundException: com.mysql.jdbc.Driver`.

Cette sous-section discute certaines des approches possibles pour corriger les problèmes liés à JDBC.

***Approche 1 : Ajouter le pilote JDBC comme dépendance dans Eclipse***
Si vous travaillez avec Eclipse, vous allez devoir ajouter un pilote JDBC comme dépendance comme suit :

1. Générez un code Java dans MapForce et importez le projet dans Eclipse, tel que décrit dans Générer, créer et exécuter le code 939.
2. Cliquez sur la configuration souhaitée dans Eclipse (par ex., `MappingApplication`).
3. Dans l'onglet *Dépendances*, cliquez sur **entrées Classpath**, puis cliquez sur **Ajouter JAR externes**.
4. Recherchez le fichier **`.jar`** de votre pilote JDBC (par ex., **`C:\jdbc\mysql\mysql-connector-java-5.1.16-bin.jar`**).
5. Cliquer sur **Run** pour exécuter le programme avec le pilote JDBC de base de données en tant que dépendance.

***Approche 2 : Ajouter le pilote JDBC à classpath de la tâche de test dans build.xml***
Si vous obtenez une erreur liée au pilote JDBC pendant la marche de test du fichier Ant **`build.xml`**, ajoutez le pilote JDBC dans le classpath de la tâche `test` dans **`build.xml`**. La liste de codes ci-dessous est un exemple d'une tâche Ant `test` qui inclut une référence au fichier **`.jar`** du pilote JDBC (*en surbrillance jaune ci-dessous*).

```
<target name="test" depends="compile">
   <java classpath="C:\codegen\java\mysql_mapping"
classname="com.mapforce.MappingConsole" fork="true" failonerror="true">
      <classpath>
         <pathelement path="${classpath}"/>
         <pathelement location="C:\jdbc\mysql\mysql-connector-java-5.1.16-bin.jar"/>
      </classpath>
      <arg line="${cmdline}"/>
   </java>
</target>
```

***Approche 3 : Inclure le pilote JDBC dans manifest de l'application***
Si vous construisez des fichiers JAR depuis l'application Java générée, ajoutez une référence au pilote de base de données dans la section `manifest` du fichier **`build.xml`**. Cela garantit que la référence au pilote de base de données est disponible dans le fichier manifest (**`MANIFEST.MF`**) une fois que vous avez construit le projet.

Pour ajouter la référence de la base de données au fichier manifest, suivez les étapes ci-dessous :

1. Situer l'élément `manifest` dans le fichier **`build.xml`**.
2. Ajouter un nouvel élément appelé `attribute`, sachant que l'attribut `name` est `Class-Path` et que l'attribut `value` est le nom du fichier **`.jar`**. Par exemple, pour MySQL 5.1.16, le fichier `manifest` mis à jour pourrait avoir l'air de ceci comme suit (*notez la ligne en surbrillance en jaune*):

   ```
   <manifest file="C:\codegen\java\mysql_mapping/META-INF/MANIFEST.MF" mode="replace">
      <attribute name="Created-By" value="MapForce 2024"/>
      <attribute name="Main-Class" value="com.mapforce.MappingConsole"/>
      <attribute name="Class-Path" value="mysql-connector-java-5.1.16-bin.jar"/>
   </manifest>
   ```

3. Copier le fichier JAR du pilote JDBC dans le dossier qui contient le fichier JAR de l'application générée.

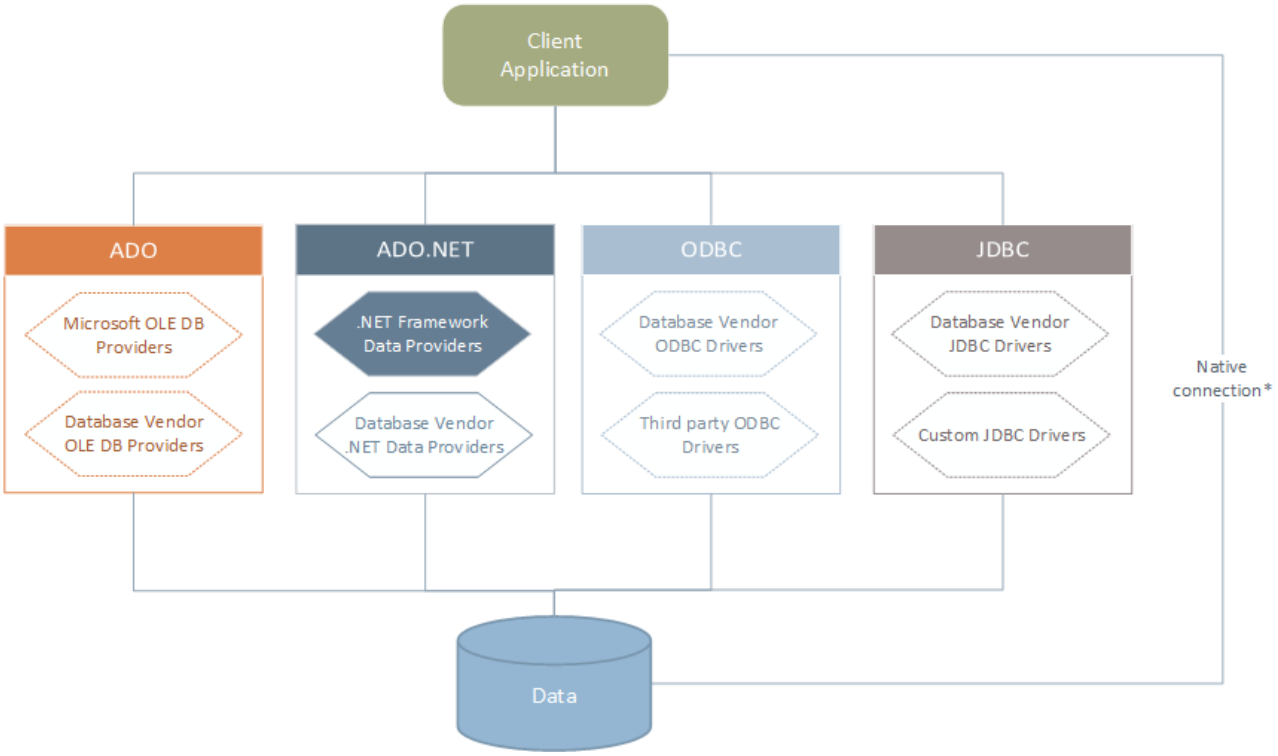
## 4.2.1 Connexion à une source de données

Dans les cas les plus simples, une base de données peut être un fichier local tel qu'un fichier de base de données Microsoft Access ou SQLite. Dans un scénario plus avancé, une base de données peut résider sur un serveur à distance ou un serveur de base données de réseau qui n'utilise pas nécessairement le même système d'exploitation que celui de l'application qui s'y connecte et qui consomme les données. Par exemple, alors que MapForce fonctionne sur un système d'exploitation Windows, la base de données à partir de laquelle vous souhaitez accéder aux données (par exemple MySQL) pourrait, elle, fonctionner sur une machine Linux.

Pour interagir avec des types de bases de données aussi bien locaux qu'à distance, MapForce repose sur les interfaces de connexion de données et les pilotes de base de données d'ores et déjà disponibles sur votre système d'exploitation ou publiés régulièrement par les distributeurs principaux de bases de données. Dans un

environnement technologique de bases de données en constante évolution, cette approche permet une plus grande flexibilité et interopérabilité multiplateforme.

Le diagramme suivant illustre, de manière simplifiée, les options de connectivité des données disponibles entre MapForce (illustré en tant qu'application de client générique) et un stockage de données (qui peut être un serveur de base de données ou un fichier de base de données.




** Les connexions natives directes sont prises en charge pour les bases de données SQLite, MySQL, MariaDB, PostgreSQL. Pour se connecter à ce type de bases de données, il n'est pas nécessaire d'installer un pilote supplémentaire sur votre système.*

Tel qu'indiqué dans le diagramme ci-dessus, MapForce peut accéder à tout type de base de données principale par le biais des technologies d'accès aux données suivantes :

- ADO (Microsoft® ActiveX® Data Objects), qui, à son tour, utilise un fournisseur OLE DB sous-jacent (Object Linking and Embedding, Database)
- ADO.NET (un ensemble de bibliothèques disponibles dans le Microsoft .NET Framework qui permet une interaction avec des données)
- JDBC (Java Database Connectivity)
- ODBC (Open Database Connectivity)

**Note :** certains fournisseurs ADO.NET ne sont pas pris en charge ou ne le sont que de manière limitée. See Notes de prise en charge d'ADO.NET 181.

## À propos des technologies d'accès aux données

L'interface de connexion aux données que vous choisissez dépend largement de votre infrastructure logicielle existante. En règle générale, vous choisirez la technologie d'accès aux données et le pilote de base de

données qui s'intègre le plus étroitement avec le système de base de données auquel vous souhaitez vous connecter. Par exemple, pour se connecter à une base de données Microsoft Access 2013, vous construiriez en principe une chaîne de connexion ADO qui utilise un fournisseur natif comme le **Microsoft Office Access Database Engine OLE DB Provider**. En revanche, pour vous connecter à Oracle, vous choisirez peut-être de télécharger et d'installer les dernières interfaces JDBC, ODBC ou ADO .NET depuis le site Internet Oracle.

Alors que les pilotes pour les produits Windows (comme Microsoft Access ou SQL Server) sont éventuellement déjà disponibles sur votre système d'exploitation Windows, ils peuvent ne pas être disponibles pour d'autres types de base de données. Les distributeurs des principales bases de données publient régulièrement des logiciels de client de base de données et des pilotes accessibles au public qui fournissent un accès multiplateforme aux bases de données respectives par le biais de combinaisons de ADO, ADO.NET, ODBC ou JDBC. De plus, plusieurs pilotes tiers peuvent être disponibles pour une ou plusieurs des technologies mentionnées ci-dessus. Dans la plupart des cas, il existe plusieurs manières de se connecter à la base de données requise depuis votre système d'exploitation, et, par conséquent, depuis MapForce. Les fonctions disponibles, paramètres de performance et les problèmes connus varieront généralement selon la technologie d'accès des données ou pilotes utilisés.

#### 4.2.1.1 Lancer l'Assistant de la connexion

MapForce fournit un assistant de connexion à la base de données qui vous guide dans les étapes requises afin de configurer une connexion vers une source de données. Avant de suivre les étapes de l'assistant, veuillez noter que pour certains types de bases de données il est nécessaire d'installer et de configurer séparément plusieurs prérequis de base de données, tels qu'un pilote de base de données ou un logiciel client de base de données. Ceux-ci sont généralement fournis par les distributeurs de base de données respectifs, et comprennent la documentation taillée sur votre version Windows spécifique. Pour une liste de pilotes regroupés par base de données, voir Aperçu des pilotes de base de données 165.

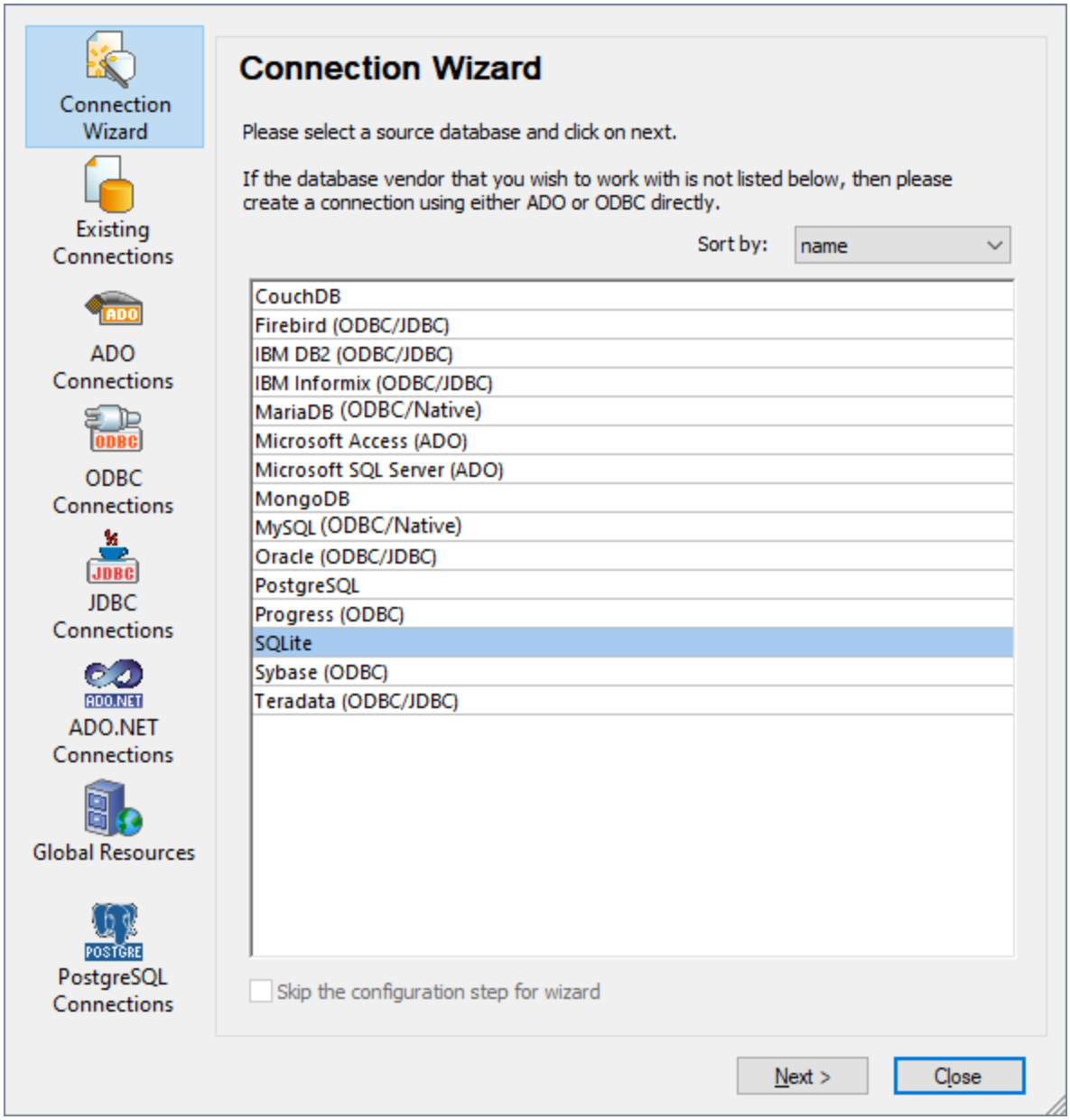
Pour démarrer l'assistant de connexion à la base de données (*voir la capture d'écran ci-dessous*), suivez les étapes suivantes :

- Dans le menu **Insérer**, cliquez sur **Base de données**.

L'assistant de connexion de connexion à la base de données (*voir la capture d'écran ci-dessous*) est lancé. À gauche de la fenêtre, vous pouvez sélectionner la manière la plus appropriée de connexion à la base de données parmi les possibilités suivantes :

- L'assistant de connexion qui vous invite à choisir votre type de base de données, puis vous guide à travers les étapes pour se connecter à une base de données de ce type
- Sélectionnez une connexion existante
- Sélectionnez une technologie d'accès aux données : ADO, ADO.NET, ODBC ou JDBC
- Utilisez une ressource globale Altova dans laquelle la connexion à la base de données est stockée
- Une connexion PostgreSQL native

Dans le volet de l'assistant de connexion (*voir la capture d'écran ci-dessous*), les bases de données peuvent être stockées de manière alphabétique par le nom du type de base de données ou par l'utilisation récente. Sélectionnez l'option que vous souhaitez dans la liste de choix *Trier par*. Après avoir sélectionné le type de base de données auquel vous voulez vous connecter, cliquez sur **Suivant**.




L'assistant vous guidera à travers les prochaines étapes conformément au type de base de données, à la technologie de connexion (ADO, ADO.NET, ODBC, JDBC) et au pilote qui sera utilisé. Pour consulter des exemples applicables aux différents types de base de données, voir Exemples de connexion à des bases de données 193.

En alternative à l'utilisation de l'assistant, vous pouvez utiliser une des technologies d'accès à la base de données suivantes :

- Configurer une connexion ADO 168
- Configurer une connexion ADO.NET 174
- Configurer une connexion ODBC 181
- Configurer une connexion JDBC 185

### 4.2.1.2 Aperçu des pilotes de base de données

La table suivante recense des pilotes communs de base de données que vous pouvez utiliser pour se connecter à une base de données particulière à travers une technologie d'accès des données particulière. Notez que cette liste ne se veut ni exhaustive ni normative ; vous pouvez utiliser d'autres alternatives tierces en plus des pilotes affichés ci-dessous.

Bien qu'un certain nombre de pilotes de base de données est peut-être déjà disponible sur votre système d'exploitation Windows, il se peut que vous nécessitiez le téléchargement d'un pilote alternatif. En ce qui concerne certaines bases de données, le dernier pilote fourni par le distributeur de bases de données est susceptible de fonctionner mieux que le pilote fourni avec le système d'exploitation.

Les distributeurs de bases de données peuvent fournir des pilotes soit en tant que pack séparément téléchargeables soit fournis avec le logiciel de client de la base de données. Dans ce cas, le logiciel de client de base de données inclut normalement tous les pilotes de base de données requis ou vous proposent une option au cours de l'installation pour sélectionner les pilotes et les composants que vous souhaitez installer. En règle générale, le logiciel de client de base de données consiste en des services d'administration et de configuration utilisés pour simplifier l'administration de base de données et la connectivité, et comporte une documentation expliquant comment installer et configurer le client de base de données et tous ses composants.

La configuration correcte du client de base de données est une étape cruciale pour la mise en place d'une connexion réussie à la base de données. Avant d'installer et d'utiliser votre logiciel de client de base de données, nous vous recommandons vivement de lire attentivement les instructions d'installation et de configuration du client de base de données, ceux-ci peuvent varier pour chaque version de base de données et pour chaque version Windows.

Afin de comprendre les possibilités et les limites de chaque technologie d'accès de données, pour chaque type de base de données, veuillez vous référer à la documentation de ce produit de base de données spécifique et testez également la connexion en comparaison avec votre environnement spécifique. Pour éviter les problèmes de connectivité les plus courants, veuillez consulter les notes et recommandations générales suivantes :

- Certains fournisseurs ADO.NET ne sont pas pris en charge ou ont une prise en charge limitée. Voir Notes de prise en charge d'ADO.NET 181.
- Lors de l'installation d'un pilote de base de données, nous vous recommandons d'en utiliser un qui ait la même plate-forme que l'application Altova (32-bit ou 64-bit). Par exemple, si vous utilisez une application Altova 32-bit sur un système d'exploitation 64-bit, installez le pilote 32-bit et configurez votre connexion de base de données à l'aide du pilote 32-bit, voir aussi Consulter les pilotes ODBC disponibles 184.
- Lorsque vous configurez une source de données ODBC, il est recommandé de créer le nom de source de données (DSN) comme System DSN à la place de User DSN. Pour plus d'informations, voir Configurer une connexion ODBC 181.
- Si la base de données cible est MySQL ou MariaDB through ODBC, l'option *Retourner des lignes correspondantes au lieu des lignes affectées* doit être activée dans l'onglet Curseur/Résultats de MySQL ODBC Connector. En alternative, si vous saisissez le string de connexion manuellement dans l'assistant Connexion de base de données, ajoutez `Option=2` au string de connexion (par ex., `Dsn=mydsn;Option=2;`).
- Lors de la configuration d'une source de données JDBC, veuillez vous assurer que le JRE (Java Runtime Environment) ou le Java Development Kit (JDK) soit installé et que la variable d'environnement

CLASSPATH du système d'exploitation soit configurée. Pour plus d'informations, voir Configurer une connexion JDBC 185.

- En ce qui concerne les instructions et les détails de prise en charge de tout pilote ou logiciel de client de base de données que vous installez à partir d'un fournisseur de base de données, contrôlez la documentation fournie avec le paquet d'installation.

| Base de données | Interface | Pilotes |
|---|---|---|
| Firebird | ADO.NET | Fournisseur de données Firebird ADO.NET (https://www.firebirdsql.org/en/additional-downloads/) |
| | JDBC | Pilote Firebird JDBC ( https://www.firebirdsql.org/en/jdbc-driver/ ) |
| | ODBC | Pilote Firebird ODBC ( https://www.firebirdsql.org/en/odbc-driver/ ) |
| IBM DB2 | ADO | Fournisseur IBM OLE DB pour DB2 |
| | ADO.NET | Fournisseur IBM Data Server pour .NET |
| | JDBC | Pilote IBM Data Server pour JDBC et SQLJ |
| | ODBC | Pilote IBM DB2 ODBC |
| IBM DB2 pour i | ADO | • Fournisseur IBM DB2 pour i5/OS IBMDA400 OLE DB<br>• Fournisseur IBM DB2 pour i5/OS IBMDARLA OLE DB<br>• Fournisseur IBM DB2 pour i5/OS IBMDASQL OLE DB |
| | ADO.NET | Fournisseur .NET Framework Data pour IBM i |
| | JDBC | Pilote IBM Toolbox pour Java JDBC |
| | ODBC | Pilote iSeries Access ODBC |
| IBM Informix | ADO | Fournisseur IBM Informix OLE DB |
| | JDBC | Pilote IBM Informix JDBC |
| | ODBC | Pilote IBM Informix ODBC |
| Microsoft Access | ADO | • Microsoft Jet OLE DB Provider<br>• Fournisseur Microsoft Access Database Engine OLE DB |
| | ADO.NET | Fournisseur de données .NET Framework pour OLE DB |
| | ODBC | • Pilote Microsoft Access |
| MariaDB | ADO.NET | En défaut d'un connecteur .NET dédié pour MariaDB, utiliser **Connector/NET** pour MySQL (https://dev.mysql.com/downloads/connector/net/). |
| | JDBC | Connecteur MariaDB /J (https://downloads.mariadb.org/) |
| | ODBC | Connecteur MariaDB /ODBC (https://downloads.mariadb.org/) |

| Base de données | Interface | Pilotes |
|---|---|---|
| | Connexion native | Disponible. Aucun pilote n'est nécessaire. |
| Microsoft SQL Server | ADO | • Microsoft OLE DB Driver for SQL Server (MSOLEDBSQL)<br>• Fournisseur Microsoft OLE DB pour SQL Server (SQLOLEDB)<br>• SQL Server Native Client (SQLNCLI) |
| | ADO.NET | • .NET Framework Fournisseur de données pour SQL Server<br>• Fournisseur de données .NET Framework pour OLE DB |
| | JDBC | • Pilote Microsoft JDBC pour SQL Server ( https://docs.microsoft.com/en-us/sql/connect/jdbc/microsoft-jdbc-driver-for-sql-server ) |
| | ODBC | • Pilote ODBC pour Microsoft SQL Server ( https://docs.microsoft.com/en-us/SQL/connect/odbc/download-odbc-driver-for-sql-server ) |
| MySQL | ADO.NET | • Connecteur/NET (https://dev.mysql.com/downloads/connector/net/) |
| | JDBC | Connecteur/J ( https://dev.mysql.com/downloads/connector/j |
| | ODBC | Connecteur/ODBC ( https://dev.mysql.com/downloads/connector/odbc) |
| | Connexion native | Disponible pour MySQL 5.7 et ultérieur. Aucun pilote n'est nécessaire. |
| Oracle | ADO | • Fournisseur Oracle pour OLE DB<br>• Fournisseur Microsoft OLE DB pour Oracle |
| | ADO.NET | Fournisseur de données Oracle pour .NET (http://www.oracle.com/technetwork/topics/dotnet/index-085163.html) |
| | JDBC | • Pilote JDBC Thin<br>• Pilote JDBC Oracle Call Interface (OCI)<br>Ces pilotes sont généralement installés pendant l'installation de votre client de base de données Oracle. Connectez-vous par le biais du pilote OCI Driver (par le Thin Driver) si vous utilisez le composant Oracle XML DB. |
| | ODBC | • Microsoft ODBC pour Oracle<br>• Pilote Oracle ODBC (généralement installé pendant l'installation de votre client de base de données Oracle) |
| PostgreSQL | JDBC | Pilote PostgreSQL JDBC ( https://jdbc.postgresql.org/download.html ) |
| | ODBC | psqlODBC ( https://odbc.postgresql.org/ ) |
| | Connexion native | Disponible. Aucun pilote n'est nécessaire. |
| Progress OpenEdge | JDBC | Connecteur JDBC ( https://www.progress.com/jdbc/openedge ) |
| | ODBC | Connecteur ODBC ( https://www.progress.com/odbc/openedge ) |

| Base de données | Interface | Pilotes |
|---|---|---|
| SQLite | Connexion native | Disponible. Aucun pilote n'est nécessaire. |
| Sybase | ADO | Fournisseur Sybase ASE OLE DB |
| | JDBC | jConnect™ pour JDBC |
| | ODBC | Pilote Sybase ASE ODBC |
| Teradata | ADO.NET | Fournisseur de données .NET pour Teradata (https://downloads.teradata.com/download/connectivity/net-data-provider-for-teradata) |
| | JDBC | Pilote JDBC Teradata (https://downloads.teradata.com/download/connectivity/jdbc-driver) |
| | ODBC | Pilote ODBC Teradata pour Windows (https://downloads.teradata.com/download/connectivity/odbc-driver/windows) |

### 4.2.1.3 Configurer une connexion ADO

Microsoft ActiveX Data Objects (ADO) est une technologie d'accès aux données qui vous permet de vous connecter à une variété de sources de données par le biais de OLE DB. OLE DB est une interface alternative à ODBC ou JDBC; elle fournit un accès uniforme aux données dans un environnement COM (Component Object Model). ADO est un précurseur du plus nouvel ADO.NET[174] et l'une des manières possibles de se connecter aux bases de données natives de Microsoft comme Microsoft Access ou SQL Server, bien que vous puissiez aussi l'utiliser pour d'autres sources de données.

Ce qui est important, c'est que vous pouvez choisir entre de multiples fournisseurs ADO, et certains doivent être téléchargés et installés sur votre poste de travail avant de pouvoir les utiliser. Par exemple, pour se connecter au serveur SQL, les fournisseurs ADO suivants sont disponibles :

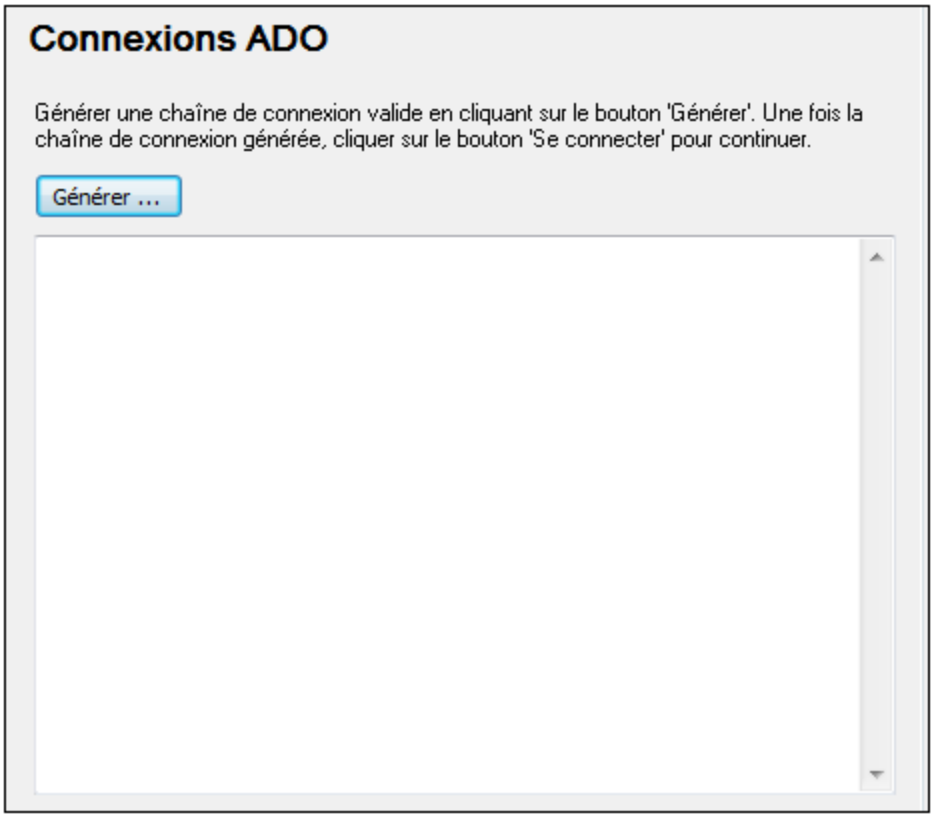
- Microsoft OLE DB *Driver* pour SQL Server (MSOLEDBSQL)
- Microsoft OLE DB *Provider* pour SQL Server (SQLOLEDB)
- SQL Server Native Client (SQLNCLI)

De la liste des fournisseurs ci-dessus, nous recommandons le MSOLEDBSQL; vous pouvez le télécharger depuis https://docs.microsoft.com/en-us/sql/connect/oledb/download-oledb-driver-for-sql-server?view=sql-server-ver15. Veuillez noter qu'il doit correspondre avec la plate-forme de MapForce (32-bit ou 64-bit). Les fournisseurs SQLOLEDB et SQLNCLI sont considérés être dépréciés et ne sont donc pas recommandés.

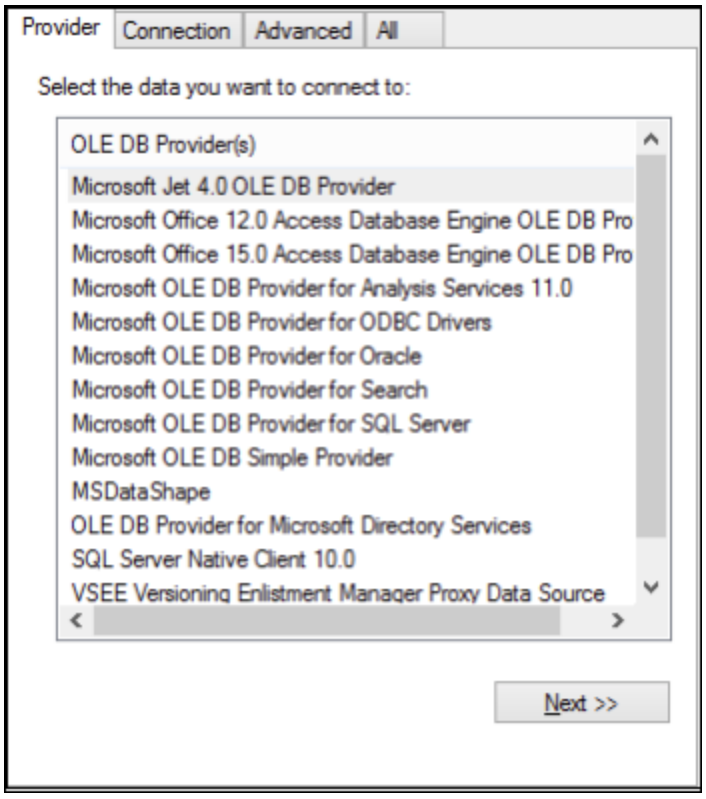
**Microsoft OLE DB Provider for SQL Server (SQLOLEDB)** est connu pour avoir des soucis avec la liaison de paramètre de requêtes complexes comme Common Table Expressions (CTE) et des déclarations imbriquées SELECT.

**Pour configurer une connexion ADO :**

1. Lancer l'assistant de connexion de base de données[163].
2. Cliquez sur **ADO Connections**.




3. Cliquez sur **Générer**.




4. Sélectionnez le fournisseur de données par le biais duquel vous souhaitez vous connecter. La table ci-dessous montre quelques scénarios communs.

| **Pour se connecter à cette base de données...** | **Utiliser ce fournisseur...** |
|---|---|
| Microsoft Access | • **Fournisseur Microsoft Office Access Database Engine OLE DB** (recommandé)<br>• **Microsoft Jet OLE DB Provider**<br><br>Si le **Microsoft Office Access Database Engine OLE DB Provider** n'est pas disponible dans la liste, assurez-vous d'avoir installé soit Microsoft Access ou Microsoft Access Database Engine Redistributable (https://www.microsoft.com/en-us/download/details.aspx?id=54920) sur votre ordinateur. |
| Serveur SQL | • **Microsoft OLE DB Driver pour SQL Server (MSOLEDBSQL)** - c'est le fournisseur OLE DB recommandé. Si vous voulez que ce fournisseur apparaisse dans la liste, vous devez le télécharger depuis https://docs.microsoft.com/en-us/sql/connect/oledb/download-oledb-driver-for-sql-server?view=sql-server-ver15 et l'installer.<br>• **Microsoft OLE DB Provider pour SQL Server (OLEDBSQL)** |

| Pour se connecter à cette base de données... | Utiliser ce fournisseur... |
|---|---|
| | • **SQL Server Native Client (SQLNCLI)** |
| Autre base de données | Sélectionnez le fournisseur applicable à votre base de données.<br><br>Si un fournisseur OLE DB de votre base de données n'est pas disponible, veuillez installer le pilote requis depuis le distributeur de base de données (voir Aperçu des pilotes de base de données 165 ). En alternative, vous pouvez établir une connexion ADO.NET, ODBC ou JDBC.<br><br>Si le système d'exploitation comporte un pilote ODBC pour la base de données requise, vous pouvez également utiliser le **Fournisseur Microsoft OLE DB pour les Pilotes ODBC** ou préférablement opter pour une connexion ODBC 181. |

5. Après avoir sélectionné le fournisseur de votre choix, cliquez sur **Suivant** et achever l'assistant.

Les étapes suivantes de l'assistant sont spécifiques au fournisseur que vous avez choisi. En ce qui concerne SQL Server, vous devrez fournir ou sélectionner le nom d'hôte du serveur de base de données, la méthode d'authentification, le nom de la base de données ainsi que le nom d'utilisateur et le mot de passe de la base de données. Pour un exemple, consultez Se connecter à Microsoft SQL Server (ADO) 217. En ce qui concerne Microsoft Access, vous serez invité à parcourir ou fournir le chemin du fichier de base de données. Pour un exemple, consultez Se connecter à Microsoft Access (ADO) 214.

La liste complète des propriétés d'initialisation (paramètres de connexion) est disponible dans l'onglet **All** de la boîte de dialogue de connexion - ces propriétés varient dépendant du fournisseur choisi et doivent éventuellement être explicitement définies de telle manière que la connexion soit possible. Les prochaines sections fournissent des orientations sur la configuration des propriétés d'intitialisation de base pour les bases de données Microsoft Access et SQL Server :

- Configurer les propriétés de liaison de données SQL Server 172
- Configurer les propriétés de liaison de données Microsoft Access 173

#### 4.2.1.3.1 Se connecter à une base de données Microsoft Access existante

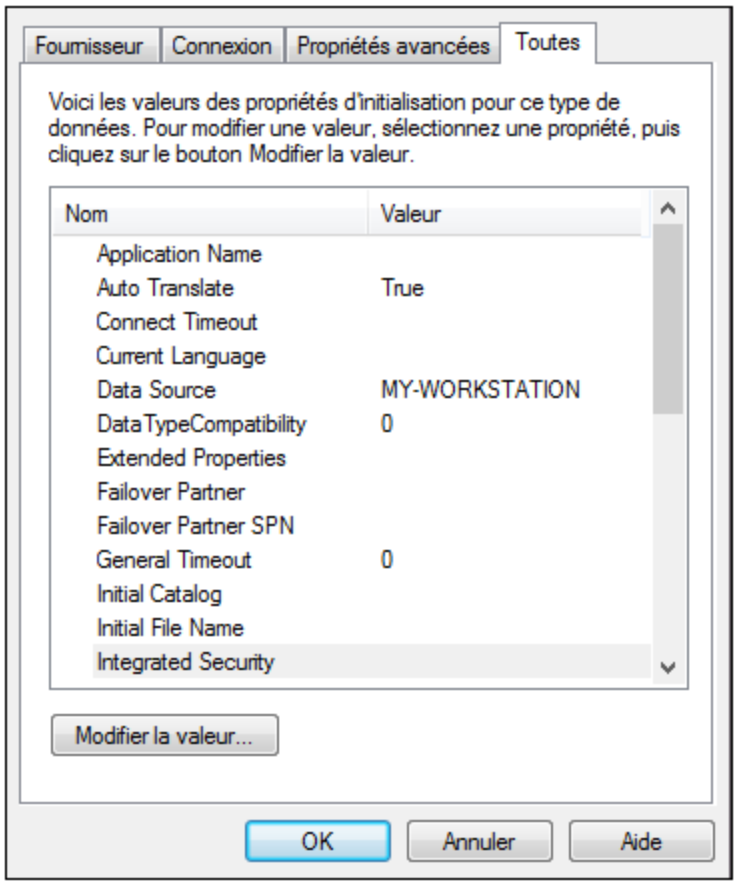
Cette approche est convenable si vous souhaitez vous connecter à une base de données Microsoft Access qui n'est pas protégée par un mot de passe. Si la base de données est protégée par un mot de passe, configurez le mot de passe de la base de données tel qu'indiqué dans Se connecter à Microsoft Access (ADO) 214.

**Pour se connecter à une base de données Microsoft Access existante :**

1. Exécuter l'assistant de connexion de base de données (voir Lancer l'Assistant de la connexion de la base de données 163).
2. Choisir **Microsoft Access (ADO)**, puis cliquez sur **Suivant**.
3. Cherchez le fichier de base de données ou saisissez le chemin y menant (soit relatif, soit absolu).
4. Cliquez sur **Se connecter**.

#### 4.2.1.3.2 Configurer les propriétés de liaison de données SQL Server

Lorsque vous vous connectez à une base de données Microsoft SQL Server par le biais d'ADO[168], veuillez vous assurer que les liaisons de données suivantes sont configurées dans l'onglet **Toutes** de la fenêtre de dialogue Propriétés des liaisons de données.

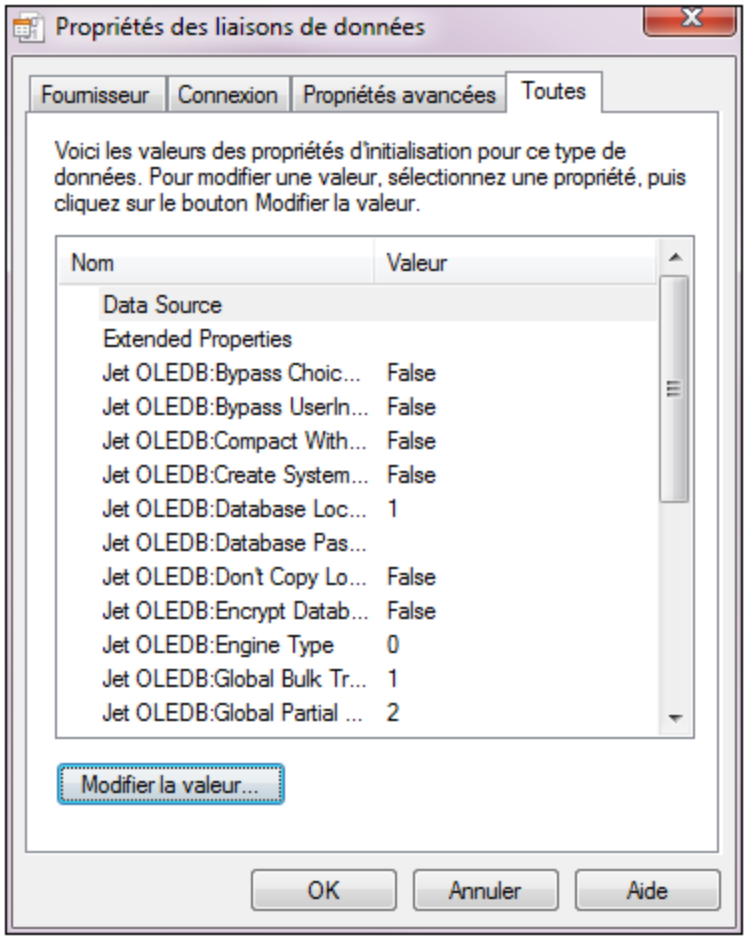



*Fenêtre de dialogue Propriétés des liaisons de données*

| Propriété | Notes |
|---|---|
| **Sécurité intégrée** | Si vous avez sélectionné le fournisseur de données **SQL Server Native Client** dans l'onglet **Fournisseur**, configurez cette propriété sur un caractère d'espace. |
| **Persister les infos de sécurité** | Configurer cette propriété sur **True**. |

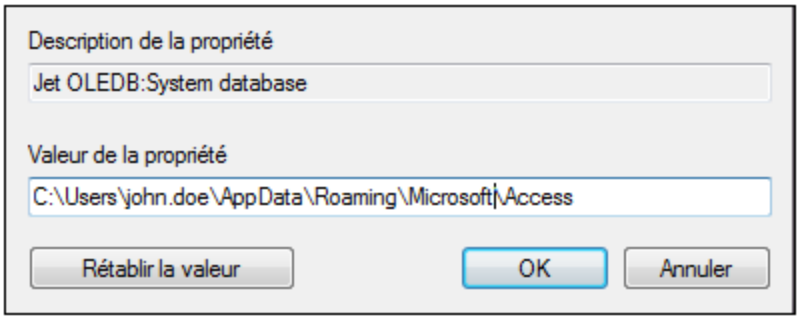
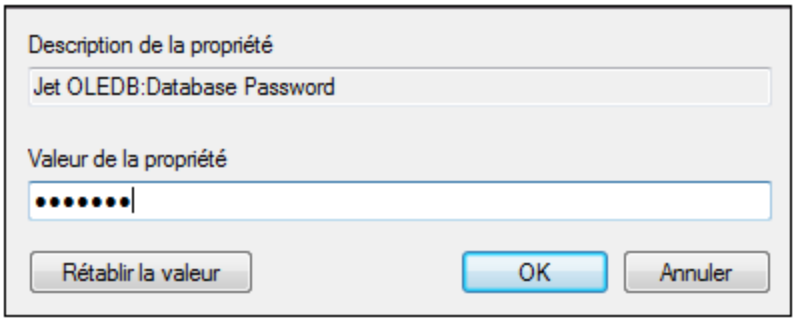
#### 4.2.1.3.3 Configurer les propriétés de liaison de données Microsoft Access

Lorsque vous vous connectez à une base de données Microsoft ADO[168], veuillez vous assurer que les propriétés suivantes sont configurées correctement dans l'onglet **Toutes** de la fenêtre de dialogue Propriétés des liaisons de données.




*Fenêtre de dialogue Propriétés des liaisons de données*

| Propriété | Notes |
|---|---|
| **Source de données** | Cette propriété stocke le chemin vers le fichier de base de données Microsoft Access. Afin d'éviter des problèmes de connectivité, il est recommandé d'utiliser le format de chemin UNC (Universal Naming Convention), par exemple :<br><br>`\\anyserver\share$\filepath` |
| **Jet OLEDB : Base de données du système** | Cette propriété stocke le chemin vers le fichier d'information du groupe de travail. Vous devrez éventuellement régler la valeur de cette propriété explicitement avant de pouvoir vous connecter à une base de données Microsoft Access. |

| Propriété | Notes |
|---|---|
| | Si vous ne pouvez pas vous connecter en raison d'une erreur de "fichier d'information de groupe de travail", localisez le fichier d'information de groupe de travail (**System.MDW**) concernant votre profil d'utilisateur et configurez la valeur de propriété sur le chemin du fichier **System.MDW**.<br>  |
| **Jet OLEDB: Mot de passe de la Base de données** | Si la base de données est protégée par un mot de passe, configurez la valeur de cette propriété sur le mot de passe de la base de données.<br>  |

## 4.2.1.4 Configurer une connexion ADO.NET

ADO.NET est un ensemble de bibliothèques Microsoft .NET Framework conçues pour interagir avec des données, y compris des données provenant de bases de données. Afin de se connecter à une base de données depuis MapForce par le biais de ADO.NET, il est nécessaire d'avoir Microsoft .NET Framework 4 ou plus élevé. Comme affiché ci-dessous, vous vous connectez à une base de données par le biais d'ADO.NET en sélectionnant un fournisseur .NET et en fournissant un string de connexion.

Un fournisseur de données .NET est une collection de classes qui permet de se connecter à une source de données de type particulier (par exemple, un Serveur SQL Server, ou une base de données Oracle), d'exécuter des commandes, et d'en extraire des données. En d'autres termes, avec ADO.NET, une application comme MapForce interagit avec une base de données par le biais d'un fournisseur de données. Chaque fournisseur de données est optimisé pour travailler avec le type spécifique de sources de données pour lequel il a été conçu. Il existe deux types de fournisseurs .NET :

1. offert par défaut avec Microsoft .NET Framework.

2. offert par des revendeurs des principales bases de données, en tant qu'une extension du .NET Framework. Ce genre de fournisseurs ADO.NET doit être installé séparément et peut être généralement téléchargé depuis le site Internet du revendeur de base de données respectif.

**Note :** certains fournisseurs ADO.NET ne sont pas pris en charge ou ne le sont que de manière limitée. Voir Notes de prise en charge d'ADO.NET 181.

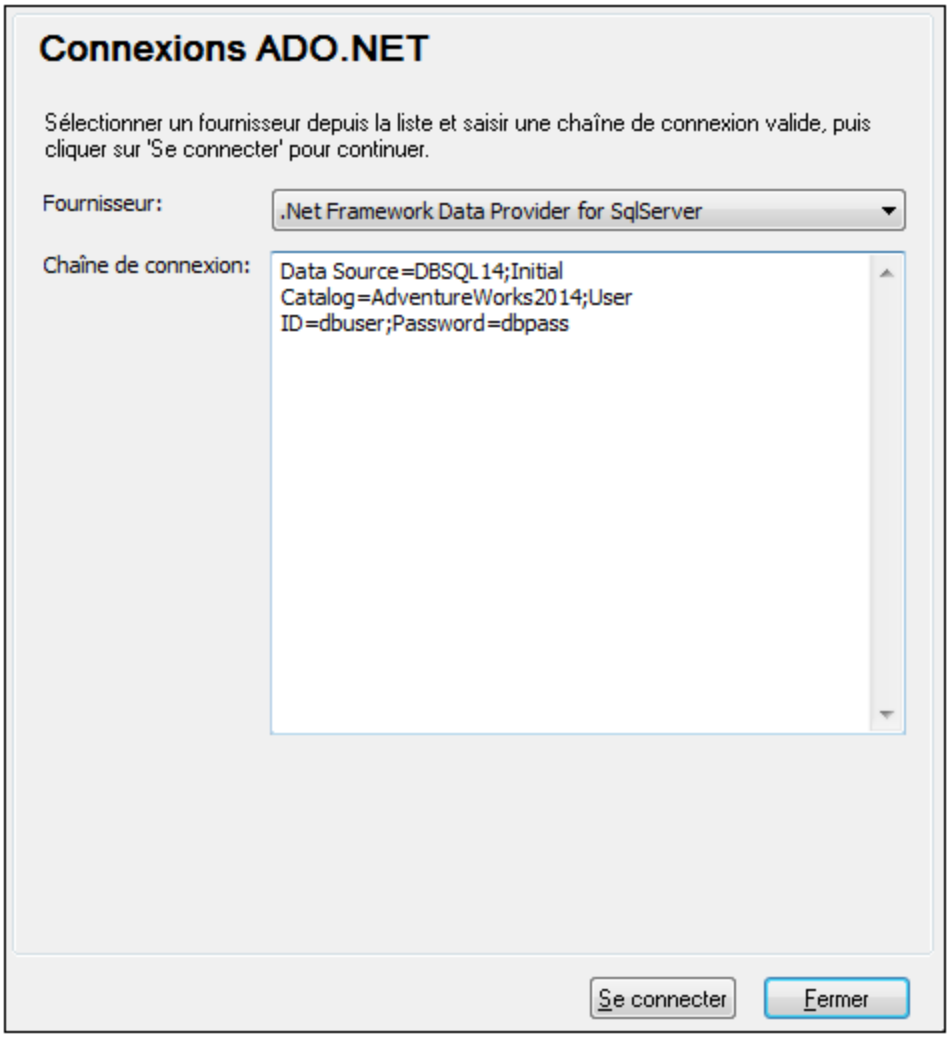
### Pour configurer une connexion ADO.NET :

1. Lancer l'assistant de connexion de base de données 163.
2. Cliquez sur **ADO.NET Connections**.
3. Sélectionnez un fournisseur de données .NET depuis la liste.

> La liste de fournisseurs disponibles par défaut avec .NET Framework apparaît dans la liste des « Fournisseurs ». Les fournisseurs de données .NET spécifiques au revendeur apparaîtront uniquement dans la liste s'ils sont déjà installés sur votre système. Pour les rendre disponibles, les fournisseurs spécifiques au revendeur doivent être installés dans le GAC (Global Assembly Cache), en exécutant le fichier .msi ou .exe fourni par le revendeur de base de données.

4. Saisir un string de connexion à une base de données. Un string de connexion définit les informations de connexion à la base de données sous la forme de paramètres de connexion de paires de clés/valeurs contenues entre des points-virgules. Par exemple, une string de connexion comme **`Data Source=DBSQLSERV;Initial Catalog=ProductsDB;User ID=dbuser;Password=dbpass`** permet de se connecter à la base de données du serveur SQL `ProductsDB` sur le serveur `DBSQLSERV`, avec le nom d'utilisateur `dbuser` et le mot de passe `dbpass`. Vous pouvez créer une string de connexion en saisissant les paires de clé/valeur directement dans la fenêtre de dialogue « Connection String ». Une autre option est de la créer avec Visual Studio (voir Créer une string de connexion dans Visual Studio 176).

> La syntaxe de la string de connexion dépend du fournisseur sélectionné depuis la liste « Fournisseur ». Par exemple voir Strings de connexion ADO.NET d'échantillon 179.



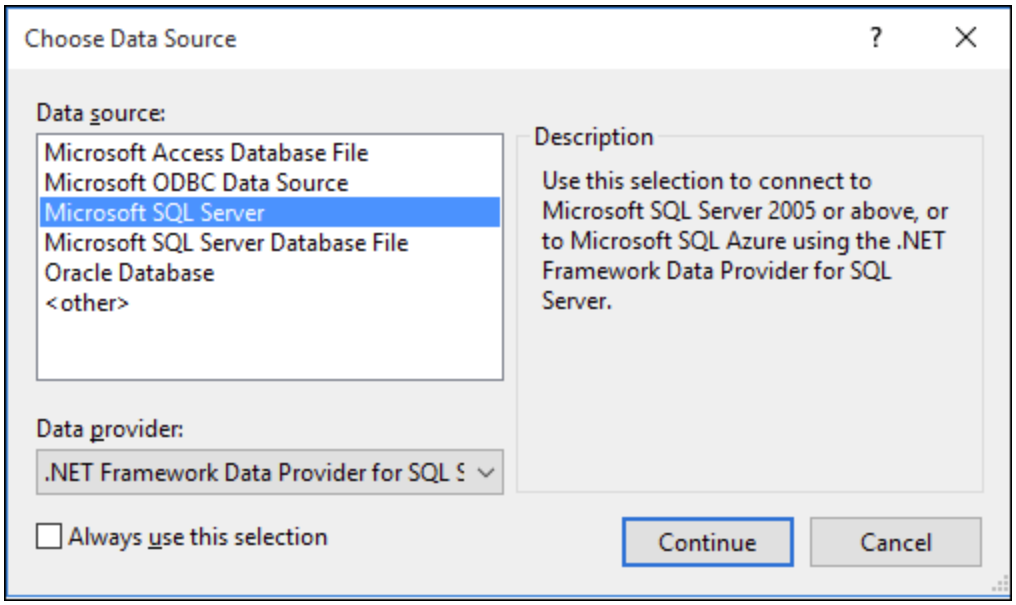

5. Cliquez sur **Se connecter**.

#### 4.2.1.4.1 Créer une chaîne de connexion dans Visual Studio

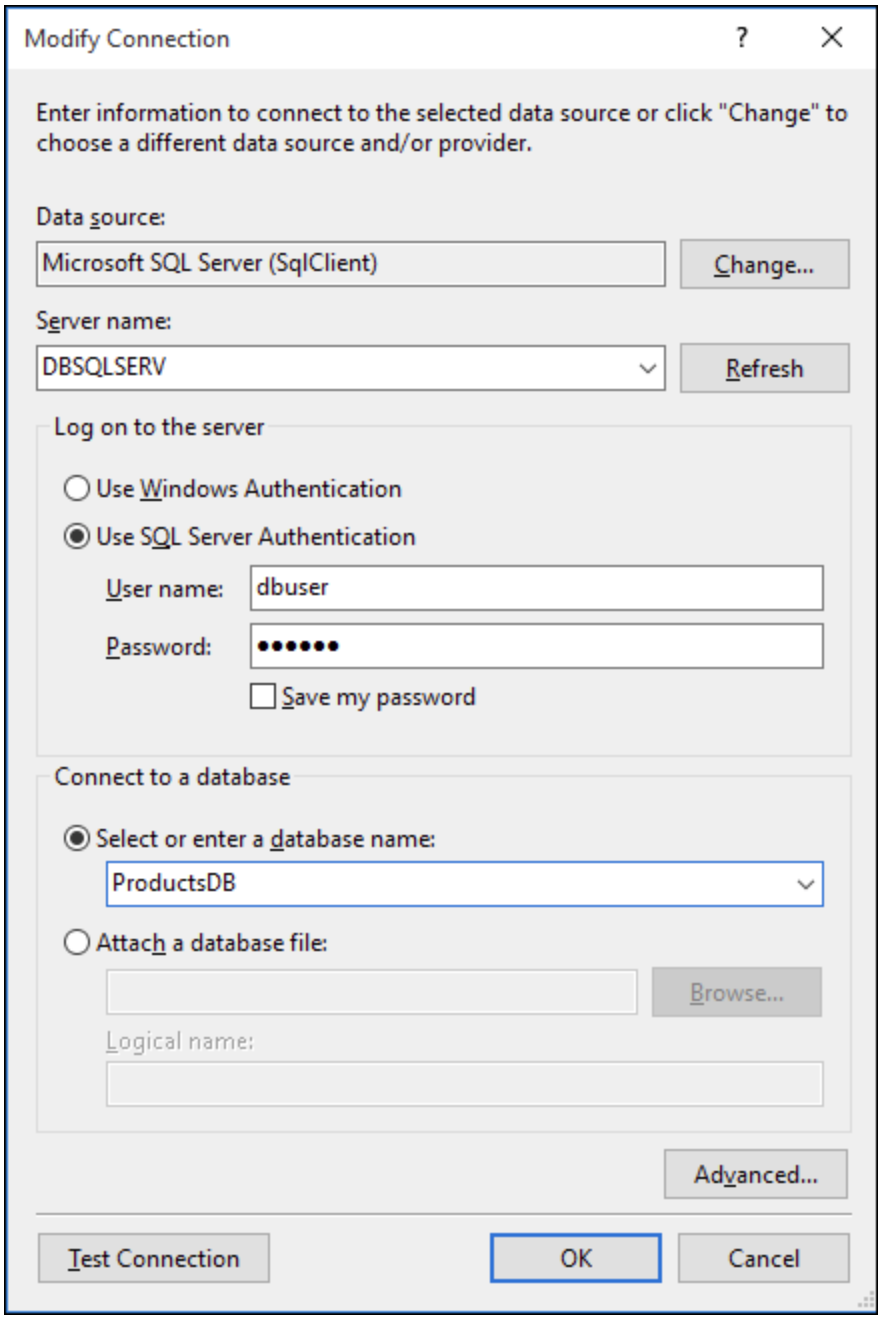
Afin de vous connecter à une source de données en utilisant ADO.NET, il est nécessaire de disposer d'une chaîne de connexion de base de données valide. Les instructions suivantes vous montrent comment créer une chaîne de connexion depuis Visual Studio.

**Pour créer une chaîne de connexion dans Visual Studio :**

1. Dans le menu **Tools**, cliquer sur **Connect to Database**.
2. Choisir une source de données depuis la liste (dans cet exemple, Microsoft SQL Server). Le fournisseur de données est rempli automatiquement sur la base de votre choix.




3. Cliquer sur **Continue**.




4. Saisir le nom de l'hôte du serveur ainsi que le nom d'utilisateur et le mot de passe de la base de données. Dans cet exemple, nous nous connectons à la base de données `ProductsDB` sur le serveur `DBSQLSERV`, à l'aide de l'authentification de serveur SQL.
5. Cliquer sur **OK**.

Si la connexion à la base de données est établie avec succès, elle apparaît dans la fenêtre du Server Explorer. Vous pouvez afficher la fenêtre de Server Explorer à l'aide de la commande de menu **View | Server Explorer**. Pour obtenir la string de connexion à la base de données, cliquer avec la touche de droite sur la fenêtre Server Explorer et sélectionner **Properties**. La string de connexion est maintenant affichée dans la fenêtre Properties

de Visual Studio. Veuillez noter que, avant de coller la string dans la fenêtre « Connection String » de MapForce, vous devrez remplacer les astérisques ( * ) par le mot de passe.

#### 4.2.1.4.2 Chaînes de connexion échantillons ADO.NET

Pour configurer une connexion ADO.NET, veuillez sélectionner un fournisseur ADO.NET depuis le dialogue de connexion à la base de données et saisir un string de connexion (voir aussi Configurer une connexion ADO.NET[174]). Les strings de connexion échantillon ADO.NET pour les bases de données variées sont recensés ci-dessous, sous le fournisseur .NET pertinent.

##### Fournisseur de données .NET pour Teradata

Ce fournisseur peut être téléchargé depuis le site web Teradata (https://downloads.teradata.com/download/connectivity/net-data-provider-for-teradata). Un string de connexion d'échantillon ressemble à l'exemple suivant :

```
Data Source=ServerAddress;User Id=user;Password=password;
```

##### Fournisseur de données .NET Framework pour IBM i

Ce fournisseur est installé dans le *IBM i Access Client Solutions - Windows Application Package*. Un string de connexion d'échantillon ressemble à l'exemple suivant :

```
DataSource=ServerAddress;UserID=user;Password=password;DataCompression=True;
```

Pour plus d'informations, consulter le fichier d'aide ".Référence technique du fournisseur .NET" contenu dans le pack d'installation ci-dessus.

##### Fournisseur de données .NET Framework pour MySQL

Ce fournisseur peut être téléchargé depuis le site Internet MySQL (https://dev.mysql.com/downloads/connector/net/). Un string de connexion d'échantillon ressemble à l'exemple suivant :

```
Server=127.0.0.1;Uid=root;Pwd=12345;Database=test;
```

Voir aussi : https://dev.mysql.com/doc/connector-net/en/connector-net-programming-connecting-connection-string.html

## Fournisseur de données .NET Framework pour Serveur SQL

Un string de connexion d'échantillon ressemble à l'exemple suivant :

```
Data Source=DBSQLSERV;Initial Catalog=ProductsDB;User ID=dbuser;Password=dbpass
```

Voir aussi : https://msdn.microsoft.com/en-us/library/ms254500(v=vs.110).aspx

## Fournisseur de données 10.1.2 IBM DB2 pour .NET Framework 4.0

```
Database=PRODUCTS;UID=user;Password=password;Server=localhost:50000;
```

**Note :** Ce fournisseur est généralement installé avec le paquet IBM DB2 Data Server Client. Si le fournisseur est absent de la liste des fournisseurs ADO.NET après l'installation du paquet IBM DB2 Data Server Client, veuillez vous référer à la note technique suivante : https://www-01.ibm.com/support/docview.wss?uid=swg21429586.

Voir aussi :
https://www.ibm.com/support/knowledgecenter/en/SSEPGG_10.1.0/com.ibm.swg.im.dbclient.adonet.ref.doc/doc/DB2ConnectionClassConnectionStringProperty.html

## Fournisseur de données Oracle pour .NET (ODP.NET)

Le pack d'installation qui comprend le fournisseur ODP.NET peut être téléchargé depuis le site Internet Oracle (voir http://www.oracle.com/technetwork/topics/dotnet/downloads/index.html). Un string de connexion d'échantillon ressemble à l'exemple suivant :

```
Data Source=DSORCL;User Id=user;Password=password;
```

DSORCL est le nom de la source de données qui pointe vers un nom de service Oracle défini dans le fichier **tnsnames.ora**, comme décrit dans la section Connexion à Oracle (ODBC) 229.

Pour vous connecter sans configurer un nom de service dans le fichier **tnsnames.ora**, utiliser un string comme :

```
Data Source=(DESCRIPTION=(ADDRESS_LIST=(ADDRESS=(PROTOCOL=TCP)(HOST=host)(PORT=port)))
(CONNECT_DATA=(SERVER=DEDICATED)(SERVICE_NAME=MyOracleSID)));User
Id=user;Password=password;
```

Voir aussi : https://docs.oracle.com/cd/B28359_01/win.111/b28375/featConnecting.htm

#### 4.2.1.4.3 Notes de prise en charge ADO.NET

La table suivante recense des pilotes de base de données ADO.NET connus qui ne sont pas pris en charge actuellement ou présentent une prise en charge limitée dans MapForce.

| Base de données | Pilote | Notes de prise en charge |
|---|---|---|
| Toutes les bases de données | **.Net Framework Data Provider for ODBC** | Prise en charge limitée. Des problèmes connus existent avec des connexions Microsoft Access. Nous recommandons d'utiliser les connexions directes ODBC à la place. |
| | **.Net Framework Data Provider for OleDb** | Prise en charge limitée. Des problèmes connus existent avec des connexions Microsoft Access. Nous recommandons d'utiliser les connexions directes ADO à la place. |
| Firebird | **Firebird ADO.NET Data Provider** | Prise en charge limitée. Nous recommandons d'utiliser ODBC ou JDBC à la place. |
| Informix | **IBM Informix Data Provider for .NET Framework 4.0** | N'est pas pris en charge. Utiliser **DB2 Data Server Provider** à la place. |
| IBM DB2 for i (iSeries) | **.Net Framework Data Provider for i5/OS** | N'est pas pris en charge. Utiliser **.Net Framework Data Provider for IBM i** à la place, installé en tant que partie de *IBM i Access Client Solutions - Windows Application Package*. |
| Oracle | **.Net Framework Data Provider for Oracle** | Prise en charge limitée. Bien que ce pilote soit fourni avec le .NET Framework, Microsoft n'encourage pas son utilisation car il est obsolète. |
| PostgreSQL | **-** | Aucun pilote ADO.NET pour ce fournisseur n'est pris en charge. Utiliser une connexion native à la place. |
| Sybase | **-** | Aucun pilote ADO.NET pour ce fournisseur n'est pris en charge. |

### 4.2.1.5 Configurer une connexion ODBC

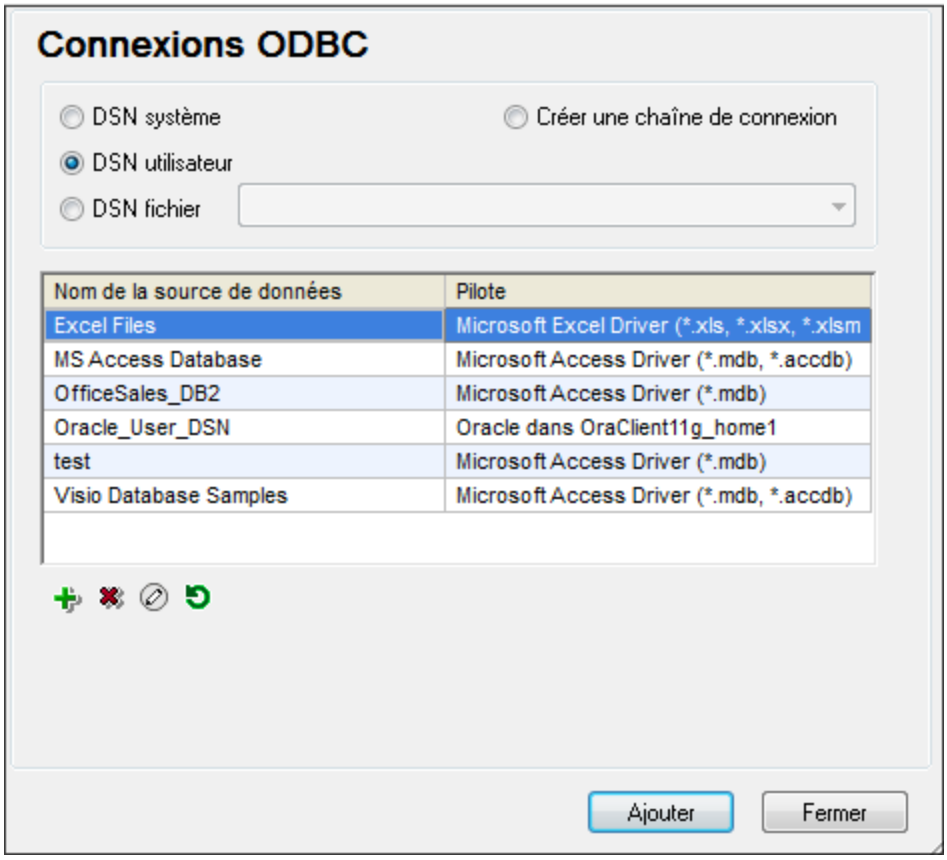
ODBC (Open Database Connectivity) est une technologie d'accès de données largement utilisée qui vous permet de vous connecter à une base de données depuis MapForce. Elle peut être utilisée soit en tant que principal moyen pour se connecter à la base de données, ou comme alternative aux connexions pilotées native, OLE DB ou JDBC.

Pour se connecter à une base de données par le biais d'ODBC, vous devrez tout d'abord créer un nom de source de données (DSN) ODBC sur le système d'exploitation. Cette étape n'est pas nécessaire si le DSN a déjà été créé, par exemple par un autre utilisateur du système d'exploitation. Le DSN représente une manière uniforme de décrire la connexion de base de données à toute application client prise en charge ODBC du système d'exploitation, y compris MapForce. Les variables peuvent être de types suivants :

- DSN Système
- DSN Utilisateur
- DSN Fichier

Une source de données *System* est accessible par tous les utilisateurs avec des privilèges sur le système d'exploitation. Une source de données *Utilisateur* est disponible pour l'utilisateur l'ayant créée. Enfin, si vous créez un *DSN Fichier*, la source de données sera créée en tant que fichier avec une extension .dsn que vous pouvez partager avec d'autres utilisateurs, à condition qu'ils aient installé les pilotes utilisés par la source de données.

Tout DSN déjà disponible sur votre machine est regroupé par la boîte de dialogue de connexion de base de données lorsque vous cliquez sur **connexions ODBC** dans la boîte de dialogue de connexion ODBC.




*Boîte de dialogue de connexions ODBC*

Si un DSN vers la base de données requise n'est pas disponible, l'assistant de connexion de base de données de MapForce vous guidera pendant la création ; néanmoins, vous pouvez aussi le créer directement sur votre système d'exploitation Windows. Dans tous les cas, avant de commencer, assurez-vous que le pilote ODBC

applicable pour votre base de données se trouve dans la liste des pilotes ODBC disponibles pour le système d'exploitation (voir Consulter les pilotes ODBC disponibles 184 ).

**Pour se connecter en utilisant un nouveau DSN :**

1. Lancer l'assistant de connexion de base de données 163.
2. Dans le dialogue de connexion à la base de données, cliquez sur **Connexions ODBC**.
3. Sélectionnez un type de source de données (DSN Utilisateur, DSN Système, DSN Fichier).

   > Pour créer un DSN Système, vous avez besoin de droits d'administration sur le système d'exploitation et MapForce doit être exécuté en tant qu'administrateur.

4. Cliquez sur **Ajouter** .
5. Sélectionnez un pilote, puis cliquez sur **DSN Utilisateur** ou **DSN Système** (dépendant du type de DSN que vous voulez créer). Si le pilote applicable pour votre base de données n'est pas dans la liste, téléchargez-le de la base de données et installez-le (voir Aperçu des pilotes de base de données 165 ).
6. Dans la boîte de dialogue qui apparaît, remplissez toute information de connexion spécifique au pilote pour compléter la configuration.

Pour que la connexion soit établie avec succès, vous devrez fournir le nom d'hôte (ou l'adresse IP) du serveur de base de données, ainsi que le nom d'utilisateur et le mot de passe de la base de données. Il peut y avoir d'autres paramètres de connexion optionnels - ces paramètres varient entre les fournisseurs de bases de données. Pour lire des informations plus détaillées concernant les paramètres spécifiques à chaque méthode de connexion, veuillez consulter la documentation du fournisseur du pilote. Pour se connecter en utilisant un DSN existant : Ceci vous permet de réutiliser les détails liés à la connexion de base de données à chaque fois que vous voulez vous connecter à la base de données. Veuillez noter que les DSN Utilisateur sont ajoutés à la liste des DSN Utilisateur alors que les DSN Système sont ajoutés à la liste des DSN Système.

**Pour se connecter en utilisant un DSN existant :**

1. Lancer l'assistant de connexion de base de données 163.
2. Cliquez sur **ODBC Connections**.
3. Sélectionnez le type de la source de données existante (DSN Utilisateur, DSN Système, DSN Fichier).
4. Cliquez sur l'enregistrement DSN existant, puis cliquez sur **Connexion**.

**Pour construire une chaîne de connexion basée sur un fichier .dsn existant :**

1. Lancer l'assistant de connexion de base de données 163.
2. Cliquez sur **ODBC Connections**.
3. Sélectionnez **Générer un string de connexion**, puis cliquez sur **Générer**.
4. Si vous souhaitez construire la chaîne de connexion en utilisant un DSN Fichier, cliquez sur l'onglet **Source de données de fichier**. Sinon, cliquez sur l'onglet **Source de données de l'appareil**. (Les DSN Système et DSN Utilisateur sont connus comme sources de données « Appareil ».)
5. Sélectionnez le fichier .dsn requis, puis cliquez sur **OK**.

**Pour se connecter en utilisant une chaîne de connexion préparée :**

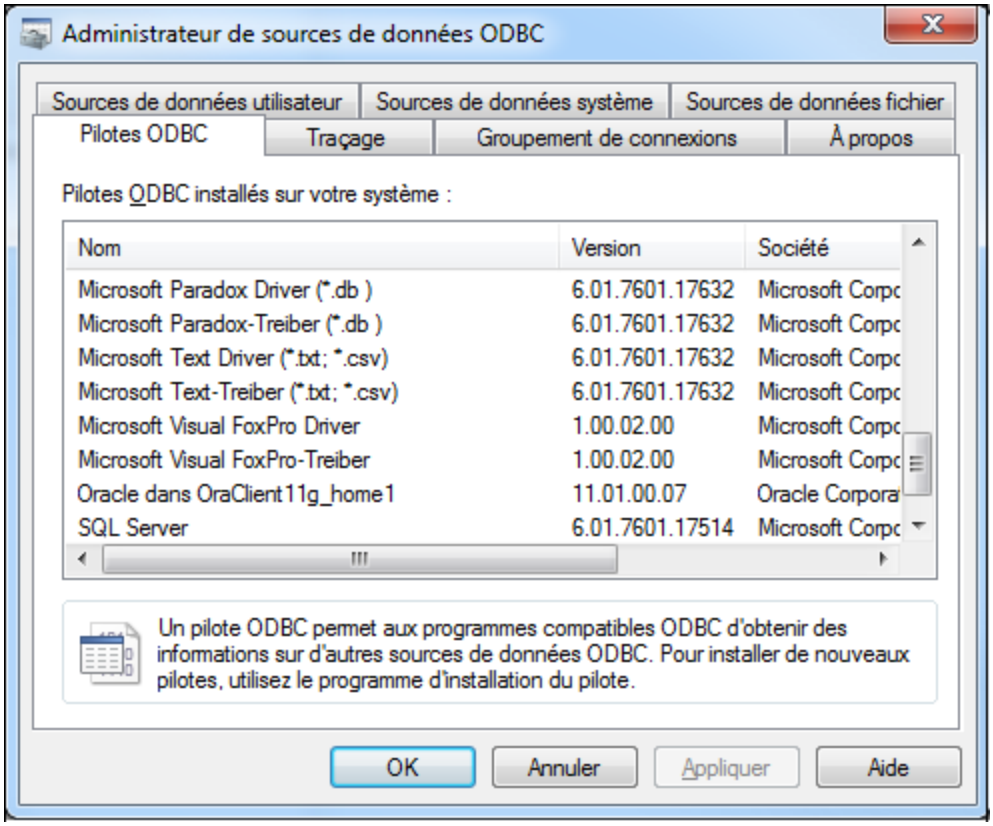
1. Lancer l'assistant de connexion de base de données 163.
2. Cliquez sur **ODBC Connections**.
3. Sélectionnez **Créer une chaîne de connexion**.
4. Collez la chaîne de connexion dans la fenêtre fournie, puis cliquez sur **Se connecter**.

#### 4.2.1.5.1 Consulter les pilotes ODBC disponibles

Vous pouvez consulter les pilotes ODBC disponibles sur votre système d'exploitation dans l'Administrateur de source de données ODBC. Vous pouvez accéder à l'Administrateur de source de données ODBC (**Odbcad32.exe**) à partir du Panneau de configuration Windows, sous **Outils d'administration**. Dans les systèmes d'exploitation 64-bit, vous trouverez deux versions de ce programme d'exécution :

- La version 32-bit du fichier **Odbcad32.exe** est située dans le répertoire **C:\Windows\SysWoW64** (en partant du principe que **C:** est votre disque système).
- La version 64-bit du fichier **Odbcad32.exe** est située dans le répertoire **C:\Windows\System32**.

Tout pilote de base de données 32-bit installé sera visible dans la version 32-bit de l'Administrateur de source de données ODBC, alors que les pilotes 64-bit seront visibles dans la version 64-bit. Veuillez donc vous assurer que vous contrôlez les pilotes de base de données depuis les versions pertinentes de l'Administrateur de source de données ODBC.




*Administrateur de source de données ODBC*

Si le pilote de votre base de données cible n'existe pas dans la liste, ou si vous souhaitez ajouter un autre pilote, vous devrez le télécharger auprès du distributeur de base de données (voir Aperçu des pilotes de base de données 165). Une fois que le pilote ODBC est disponible sur votre système, vous êtes prêt à créer des connexions ODBC (voir Configurer une connexion ODBC 181).

## 4.2.1.6 Configurer une connexion JDBC

JDBC (Java Database Connectivity) est une interface d'accès à des bases de données, elle fait partie de la plateforme logicielle Java d'Oracle. Les connexions JDBC nécessitent généralement plus de ressources que les connexions ODBC, mais elles peuvent fournir des fonctions qui ne sont pas disponibles par le biais d'ODBC.

### Prérequis

- JRE (Java Runtime Environment) ou Java Development Kit (JDK) doit être installé. Il peut s'agir soit d'Oracle JDK soit d'un build open source comme Oracle OpenJDK. MapForce déterminera le chemin vers la Java Virtual Machine (JVM) depuis les emplacements suivants, dans l'ordre suivant : a) le chemin personnalisé JVM que vous avez éventuellement défini dans l'application **Options**, voir Paramètres Java 1091; b) le chemin JVM trouvé dans le registre Windows ; c) la variable d'environnement `JAVA_HOME`.
- Veuillez vous assurer que la plateforme de MapForce (32-bit, 64-bit) corresponde à celle du JRE/JDK.

- Les pilotes JDBC du distributeur de base de données doivent être installés. Il se peut que ce soient des pilotes JDBC installés en tant que partie de l'installation client de la base de données, ou de bibliothèques JDBC (.jar files) téléchargées séparément, si disponible ou pris en charge par la base de données, voir aussi Exemples de connexion à la base de données 193.
- La variable d'environnement CLASSPATH doit inclure le chemin du pilote JDBC (un ou plusieurs fichiers .jar) sur votre système d'exploitation Windows. Lorsque vous installez des clients de base de données, l'installateur peut configurer cette variable automatiquement. Voir aussi Configuration de CLASSPATH 188.

## Se connecter à SQL Server par le biais de JDBC avec des identifiants Windows

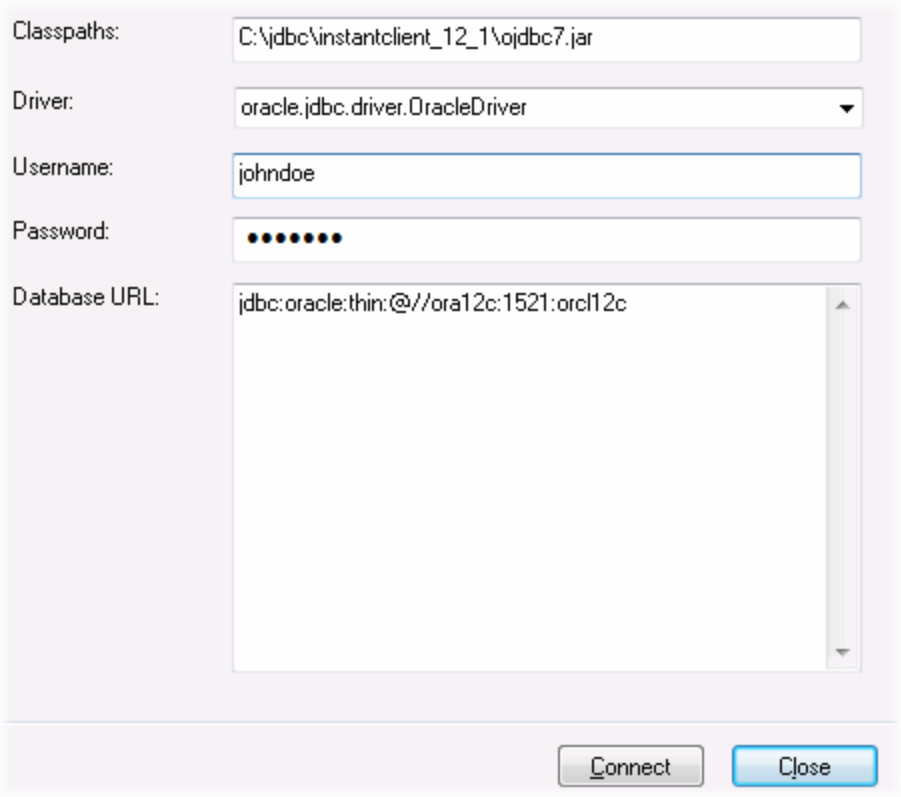
Si vous vous connectez à SQL Server par le biais de JDBC avec des identifiants Windows (sécurité intégrée), veuillez noter :

- Le fichier **sqljdbc_auth.dll** contenu dans le package du pilote JDBC doit être copié dans un répertoire qui se trouve dans la variable d'environnement du système PATH. Il y a deux fichiers de ce type, un pour la plateforme x86 et un pour la plateforme x64. Veuillez vous assurer que vous ajoutez à PATH celui qui correspond à votre plateforme JDK. De même, veuillez vous assurer que vous redémarrez MapForce (ou le programme qui exécute le mappage, le cas échéant) après avoir changé la variable d'environnement.
- Le string de connexion JDBC doit inclure la propriété **`integratedSecurity=true`**. Vous pouvez ajouter cette propriété depuis plusieurs endroits :
  - depuis l'assistant à la connexion de base de données, voir ci-dessous
  - depuis les paramètres de composant 256 de base de données
  - le cas échéant, en éditant le string de connexion à la base de données dans du code Java généré.

Pour plus d'informations, veuillez consulter la documentation *Microsoft JDBC driver for SQL Server*, https://docs.microsoft.com/en-us/sql/connect/jdbc/building-the-connection-url.

## Configurer une connexion JDBC

1. Lancer l'assistant de connexion de base de données 163.
2. Cliquez sur **Connexions JDBC**.
3. En option, saisir une liste de chemins d'accès de fichiers .jar séparées par des points-virgules dans la zone de texte « Chemins de classe ». Les bibliothèques .jar qui sont saisies ici seront chargées dans l'environnement en plus de celles déjà définies dans la variable d'environnement CLASSPATH. Une fois que vous avez terminé d'éditer la fenêtre de texte "Chemins de classe", tout pilote JDBC trouvé dans les librairies de source .jar sont automatiquement ajoutées à la liste « Pilote » (voir l'étape suivante).




4. À côté de « Pilote », choisir un pilote JDBC depuis la liste des pilotes ou saisir un nom de classe Java. Veuillez noter que cette liste contient tous les pilotes JDBC configurés par la variable d'environnement CLASSPATH (voir Configuration de CLASSPATH 188) ainsi que celles trouvées dans la zone de texte « Chemins de classe ».

   Les chemins de pilote JDBC définis dans la variable CLASSPATH, ainsi que dans tout chemin de fichier .jar saisis directement dans la boîte de dialogue de connexion de base de données sont tous fournis sur la Java Virtual Machine (JVM). La JVM décide ensuite quels pilotes utiliser pour établir une connexion. Il est recommandé de garder une trace des classes Java chargées dans la JVM pour ne pas créer de conflits de pilotes JDBC potentiels et éviter des résultats inattendus lors d'une connexion à la base de données.

5. Saisir le nom d'utilisateur et le mot de passe de la base de données dans les fenêtres correspondantes.
6. Dans la zone de texte URL de base de données, saisir le string URL de connexion JDBC dans le format spécifique à votre type de base de données. La table suivante décrit la syntaxe des URL de connexion JDBC (chaînes) pour les types de base de données communs.

| Base de données | URL de connexion JDBC |
|---|---|
| Firebird | `jdbc:firebirdsql://<host>[:<port>]/<database path or alias>` |
| IBM DB2 | `jdbc:db2://hostName:port/databaseName` |

| Base de données | URL de connexion JDBC |
|---|---|
| IBM DB2 for i | `jdbc:as400://[host]` |
| IBM Informix | `jdbc:informix-sqli://hostName:port/databaseName:INFORMIXSERVER=myserver` |
| MariaDB | `jdbc:mariadb://hostName:port/databaseName` |
| Microsoft SQL Server | `jdbc:sqlserver://hostName:port;databaseName=name` |
| MySQL | `jdbc:mysql://hostName:port/databaseName` |
| Oracle | `jdbc:oracle:thin:@hostName:port:SID`<br>`jdbc:oracle:oci:@//hostName:port:service` |
| Oracle XML DB | `jdbc:oracle:oci:@//hostName:port:service` |
| PostgreSQL | `jdbc:postgresql://hostName:port/databaseName` |
| Progress OpenEdge | `jdbc:datadirect:openedge://host:port;databaseName=db_name` |
| Sybase | `jdbc:sybase:Tds:hostName:port/databaseName` |
| Teradata | `jdbc:teradata://databaseServerName` |

**Note :** des variations de syntaxe pour les formats cités ci-dessus sont également possibles (par exemple, l'URL de base de données peut exclure le port ou peut inclure le nom d'utilisateur et le mot de passe de la base de données). Vérifiez la documentation du distributeur de la base de données pour plus de détails.

7. Cliquez sur **Se connecter**.

#### 4.2.1.6.1 Configuration de CLASSPATH

La variable d'environnement CLASSPATH est utilisée par Java Runtime Environment (JRE) ou le Kit de développement Java (JDK) pour situer les classes Java et d'autres fichiers de ressource sur votre système d'exploitation. Lorsque vous vous connectez à une base de données par le biais de JDBC, cette variable doit être configurée pour contenir le chemin vers le pilote JDBC sur votre système d'exploitation, et, dans certains cas, le chemin vers des fichiers de librairie supplémentaires spécifiques au type de base de données que vous utilisez.

La table suivante recense les chemins de fichier échantillon qui sont généralement inclus dans la variable CLASSPATH. Attention ! Vous devrez éventuellement ajuster cette information sur la base de l'emplacement du pilote JDBC sur votre système, du nom de pilote JDBC, et de la version JRE/JDK présente sur votre système d'exploitation. Pour éviter tout problème de connectivité, contrôlez les instructions d'installation et les étapes de configuration de pré-installation ou de post-installation applicables au pilote JDBC installé sur votre système d'exploitation.

| Base de données | Entrées CLASSPATH échantillons |
|---|---|
| Firebird | `C:\Program Files\Firebird\Jaybird-2.2.8-JDK_1.8\jaybird-full-2.2.8.jar` |
| IBM DB2 | `C:\Program Files (x86)\IBM\SQLLIB\java\db2jcc.jar;C:\Program Files (x86)\IBM\SQLLIB\java\db2jcc_license_cu.jar;` |
| IBM DB2 for i | `C:\jt400\jt400.jar;` |
| IBM Informix | `C:\Informix_JDBC_Driver\lib\ifxjdbc.jar;` |
| Microsoft SQL Server | `C:\Program Files\Microsoft JDBC Driver 4.0 for SQL Server\sqljdbc_4.0\enu\sqljdbc.jar` |
| MariaDB | **`<installation directory>`**`\mariadb-java-client-2.2.0.jar` |
| MySQL | `mysql-connector-java-`*`version`*`-bin.jar;` |
| Oracle | **`ORACLE_HOME`**`\jdbc\lib\ojdbc6.jar;` |
| Oracle (avec XML DB) | **`ORACLE_HOME`**`\jdbc\lib\ojdbc6.jar;`**`ORACLE_HOME`**`\LIB\xmlparserv2.jar;` **`ORACLE_HOME`**`\RDBMS\jlib\xdb.jar;` |
| PostgreSQL | **`<installation directory>`**`\postgresql.jar` |
| Progress OpenEdge | `%DLC%\java\openedge.jar;%DLC%\java\pool.jar;`<br><br>Note : Si le Progress OpenEdge SDK est installé sur la machine, `%DLC%` est le répertoire où OpenEdge est installé. |
| Sybase | `C:\sybase\jConnect-7_0\classes\jconn4.jar` |
| Teradata | **`<installation directory>`**`\tdgssconfig.jar;`**`<installation directory>`**`\terajdbc4.jar` |

- Un changement de la variable CLASSPATH peut influer sur le comportement des applications Java sur votre machine. Afin de comprendre les implications possibles avant de commencer, veuillez vous référer à la documentation Java.
- Les variables d'environnement peuvent être utilisateur ou système. Pour changer les variables d'environnement système, vous aurez besoin de droits administratifs sur le système d'exploitation.
- Après avoir changé la variable d'environnement, redémarrez les programmes en cours pour que la configuration puisse prendre effet. En alternative, déconnectez-vous ou redémarrez votre système d'exploitation.

### Pour configurer CLASSPATH sur Windows 7 :

1. Ouvrir le menu **Démarrer** et cliquer avec le bouton de droite sur **Ordinateur**.
2. Cliquer sur **Propriétés**.
3. Cliquer sur **Configurations de système avancées**.
4. Dans l'onglet **Avancé**, cliquer sur **Variables d'environnement**,

5. Localiser la variable CLASSPATH sous les variables d'environnement utilisateur ou système, puis cliquer sur **Éditer**. Si la variable CLASSPATH n'existe pas, cliquer sur **Nouveau** pour la créer.
6. Éditer la valeur de variable pour inclure le chemin sur votre système d'exploitation où se trouve le pilote JDBC. Pour séparer le chemin de pilote JDBC des autres chemins qui pourraient déjà se trouver dans la variable CLASSPATH, utiliser un point-virgule en tant que séparateur ( ; ).

**Pour configurer CLASSPATH sur Windows 10 :**

1. Cliquer sur la touche Windows, puis saisir **"variables d'environment"**.
2. Cliquer sur **Éditer les variables d'environnement de système**.
3. Cliquer sur **Variables d'environnement**.
4. Localiser la variable CLASSPATH sous les variables d'environnement utilisateur ou système, puis cliquer sur **Éditer**. Si la variable CLASSPATH n'existe pas, cliquer sur **Nouveau** pour la créer.
5. Éditer la valeur de variable pour inclure le chemin sur votre système d'exploitation où se trouve le pilote JDBC. Pour séparer le chemin de pilote JDBC des autres chemins qui pourraient déjà se trouver dans la variable CLASSPATH, utiliser un point-virgule en tant que séparateur ( ; ).

## 4.2.1.7 Configurer une connexion SQLite

SQLite est un type autonome de base de données sur base de fichier, idéal pour des scénarios mettant l'accent sur la portabilité et la simplicité de configuration. Puisque les bases de données SQLite sont prises en charge nativement par MapForce, vous n'aurez pas besoin d'installer des pilotes pour les connecter.

### Notes de prise en charge de la base de données SQLite

- Sur Linux, le timeout pour l'exécution d'instruction n'est pas pris en charge pour les bases de données SQLite.
- La recherche plein texte n'est pas prise en charge pour les tables.
- SQLite permet des valeurs de types de données différents dans chaque ligne d'une table donnée. Toutes les valeurs traitées doivent être compatibles avec le type de colonne déclaré ; c'est pourquoi les valeurs non attendues peuvent être extraites et les erreurs d'exécution peuvent se produire si votre base de données SQLite a des valeurs de ligne qui ne sont pas les mêmes que le type de colonne déclaré.
- Si votre mappage doit écrire des donnés sur une base de données SQLite et si vous ne possédez pas déjà le fichier de base de données cible, vous devrez le créer séparément. Dans ce cas, vous soit le créer avec un outil comme DatabaseSpy soit télécharger le shell de ligne de commande SQLite depuis le site Internet officiel, puis créer le fichier de base de données depuis la ligne de commande (voir aussi Exemple : Données de mappage de XML à SQLite). Pour une référence complète à la syntaxe de commande SQLite, veuillez vous référer à la documentation SQLite officielle.
- Les bases de données SQLite sont prises en charge dans le langage de transformation BUILT-IN MapForce (soit lorsque vous prévisualisez le mappage, soit lorsque vous exécutez un fichier d'exécution MapForce Server).
- Les bases de données SQLite ne sont pas prises en charge dans les fonctions définies par les utilisateurs (UDF).

**Important**

Il est recommandé de créer des tables avec le mot-clé `STRICT` afin de garantir un comportement plus prévisible de vos données. Autrement, les données ne sont pas écrites ou lues correctement lorsque les

valeurs de différents types sont mélangées dans une colonne. Pour en savoir plus sur les tables STRICT, voir la documentation SQLite.

#### 4.2.1.7.1 Connexion à une base de données SQLite existante

**Pour se connecter à une base de données SQLite existante :**

1. Exécuter l'assistant de connexion de base de données (voir Lancer l'Assistant de la connexion de la base de données 163).
2. Choisir **SQLite**, puis cliquer sur **Suivant**.
3. Chercher le fichier de base de données SQLite, ou saisir le chemin (soit relatif ou absolu) à la base de données. La touche **Se connecter** s'active une fois que vous avez saisi le chemin vers un fichier de base de données SQLite.
4. Cliquer sur **Se connecter**.

### 4.2.1.8 Connexion native

Les connexions natives sont des connexions directes vers la BD qui n'ont pas besoin de pilotes pour être installés. Aussi, si vous avez l'intention de déployer des fichiers pour l'exécution sur le serveur Linux ou macOS, vous ne devez pas installer les pilotes sur le serveur lui-même.

Vous pouvez configurer des connexions natives pour les BD suivantes :

- MariaDB
- MySQL
- SQLite
- PostgreSQL

Si vous préférez établir une connexion par le biais d'un pilote voir les sections suivantes :

- Configurer une connexion JDBC 185
- Connexion SQLite 190
- Connexion à PostgreSQL (ODBC) 236

#### Configuration de connexion

Pour configurer une connexion native, veuillez suivre les étapes suivantes. Vous aurez besoin des informations suivantes : nom d'hôte, port, nom de base de données, nom d'utilisateur et mot de passe.

1. Lancer l'assistant de connexion de base de données 163.
2. Sélectionnez la BD que vous souhaitez connecter à (MariaDB, MySQL, PostgreSQL, ou SQLite).
3. Dans le dialogue qui apparaît, saisissez l'hôte (par exemple, *localhost*), en option le port (typiquement 5432), Mode SSL dans le cas de MySQL, le nom de la base de données, nom utilisateur et mot de passe dans les boîtes correspondantes.
4. Cliquez sur **Connexion**.

### Connexions SQLite

Pour des informations détaillées sur les connexions SQLite, voir le sujet Connexion SQLite[190].

### Notes pour PostrgreSQL

Si le serveur de base de données PostgreSQL se trouve sur une machine différente, veuillez noter les points suivants :

- Le serveur de base de données PostgreSQL doit être configuré de manière à accepter les connexions provenant des clients. En particulier, le fichier **pg_hba.conf** doit être configuré de manière à permettre des connexions non locales. Ensuite, le fichier **postgresql.conf** doit être configuré de manière à écouter des adresses IP et des ports spécifiés. Pour plus d'information, consultez la documentation PostgreSQL (https://www.postgresql.org/docs/9.5/static/client-authentication-problems.html).
- Le serveur doit être configuré de manière à accepter les connexions sur le port désigné (en général 5432) par le biais du pare-feu. Par exemple, sur un serveur de base de données exécuté sur Windows, une règle peut être créée pour permettre des connexions sur le port 5432 par le biais du pare-feu, depuis **Panneau de configuration > Pare-feu Windows > Fonctions avancées > Règles entrantes**.

## 4.2.1.9 Ressources globales

Après avoir créé une base de données en tant que ressource globale, ses détails de connexion sont stockés et peuvent être utilisés dans tous les produits Altova installés sur votre appareil.

### Créer une base de données en tant que ressource globale

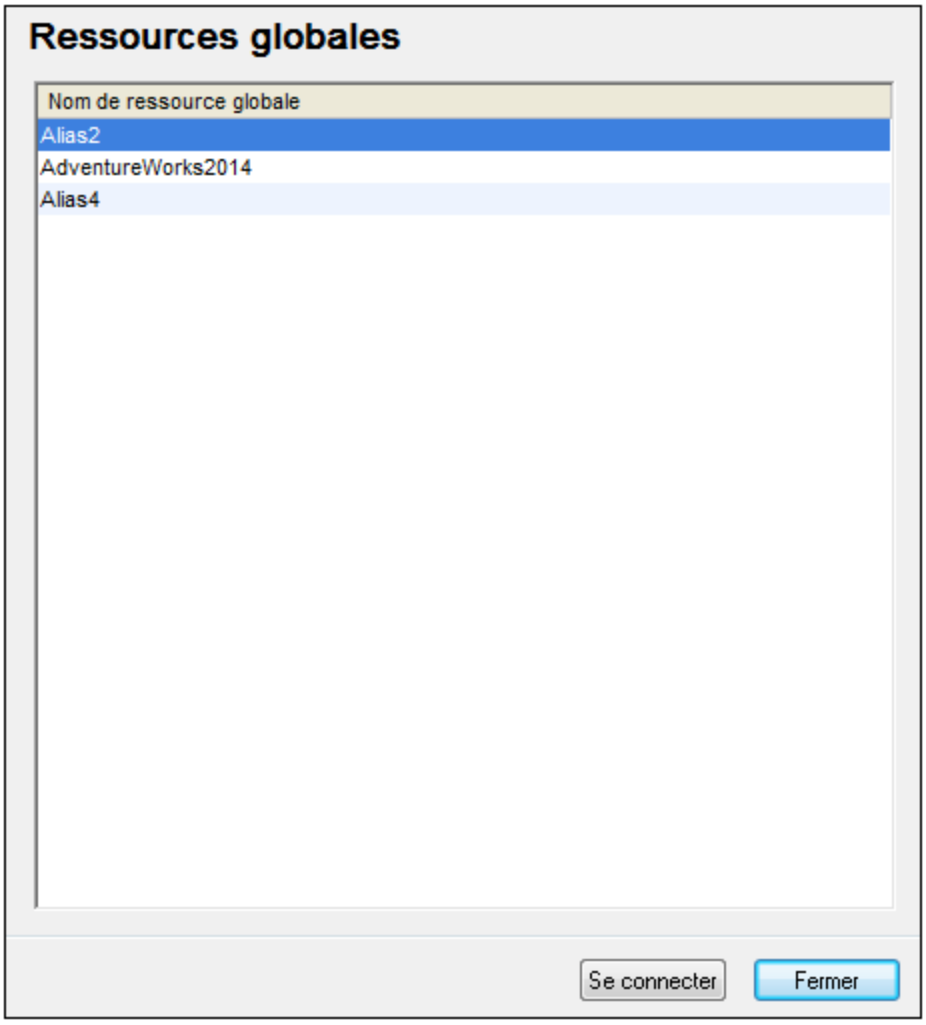
Pour créer une base de données en tant que ressource globale, suivez les étapes suivantes

1. Dans le menu **Outils** de MapForce, cliquez sur les **Ressources globales**.
2. Cliquez sur **Ajouter**, puis cliquez sur Base de données.
3. Saisissez un nom pour la ressource globale dans le champ *Resource Alias*.
4. Cliquer sur **Choose Database**. L'assistant de connexion[163] apparait.
5. Utiliser l'assistant de connexion pour ajouter une connexion de base de données, tel que décrit ci-dessus.

### Utiliser une base de données ressource globale

Pour utiliser une base de données qui a été créée en tant que ressource globale (*voir-ci-dessus*), suivez les étapes suivantes :

1. Lancer l'assistant de connexion de base de données, tel que décrit ci-dessus.
2. Sélectionnez les Ressources globales. Toutes les bases de données qui ont été créées en tant que ressources globales seront recensées par leurs noms dans le volet des Ressources globales (*voir la capture d'écran ci-dessous*).




3. Sélectionnez la ressource globale de votre choix. Astuce : déplacez le curseur sur une ressource globale dans la liste afin de voir l'information sur les bases de données.

## 4.2.1.10 Exemples de connexion à la base de données

Cette section comprend des exemples de connexion à une base de données provenant de MapForce par le biais de ADO, ODBC ou JDBC. Les exemples de connexion ADO.NET sont recensés séparément, voir Chaînes de connexion échantillons ADO.NET[179]. Pour consulter des instructions concernant la mise en place d'une connexion native sur PostgreSQL et SQLite, voir Configurer une connexion PostgreSQL[191] et Configurer une connexion SQLite[190], respectivement.

Veuillez noter les points suivants :

- Les instructions peuvent différer si votre configuration Windows, l'environnement de réseau et le client de base de données ou le logiciel de serveur ne sont pas les mêmes que ceux décrits dans chaque exemple.

- En ce qui concerne la plupart des types de base de données, il est possible de se connecter en utilisant plus d'une technologie d'accès aux données (ADO, ADO.NET, ODBC, JDBC) ou pilote. La performance de la connexion de la base de données, ainsi que ses fonctions et ses limitations dépendront du pilote sélectionné, du logiciel client de base de données (le cas échéant), et de tout autre paramètre de connectivité que vous êtes susceptible d'avoir configuré outre MapForce.

#### 4.2.1.10.1 Firebird (JDBC)

Cet exemple illustre comment se connecter à un serveur de base de données Firebird par le biais de JDBC.

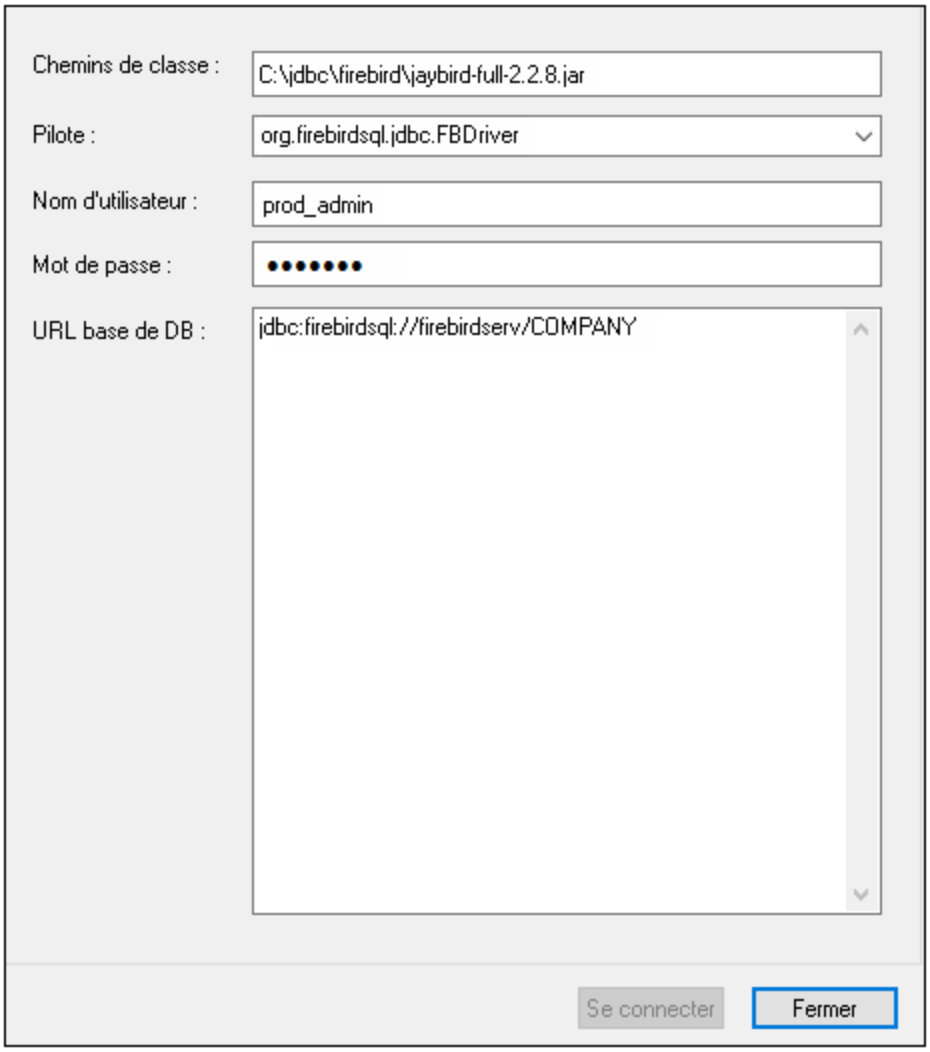
**Conditions préalables :**

- JRE (Java Runtime Environment) ou Java Development Kit (JDK) doit être installé. Il peut s'agir soit d'Oracle JDK soit d'un build open source comme Oracle OpenJDK. MapForce déterminera le chemin vers la Java Virtual Machine (JVM) depuis les emplacements suivants, dans l'ordre suivant : a) le chemin personnalisé JVM que vous avez éventuellement défini dans l'application **Options**, voir Paramètres Java 1091; b) le chemin JVM trouvé dans le registre Windows ; c) la variable d'environnement `JAVA_HOME`.
- Veuillez vous assurer que la plateforme de MapForce (32-bit, 64-bit) corresponde à celle du JRE/JDK.
- Le pilote Firebird JDBC doit être disponible sur votre système d'exploitation (il prend la forme d'un fichier .jar qui offre une connectivité à la base de données). Le pilote peut être téléchargé depuis le site web Firebird ( https://www.firebirdsql.org/ ). Cet exemple utilise *Jaybird 2.2.8.*
- Vous disposez des détails de connexion pour la base de données suivante : l'hôte, le chemin de base de données ou alias, le nom d'utilisateur et le mot de passe.

**Pour se connecter à Firebird par le biais de JDBC :**

1. Lancer l'assistant de connexion de base de données 163.
2. Cliquer sur **JDBC Connections**.
3. À côté de « Chemins de classe », saisir le chemin menant au fichier .jar qui fournit la connexion à la base de données. Si nécessaire, vous pouvez aussi saisir une liste séparée par des points-virgules de chemins d'accès de fichiers .jar. Dans cet exemple, le fichier .jar exigé se trouve sous le chemin suivant : **C:\jdbc\firebird\jaybird-full-2.2.8.jar**. Veuillez noter que vous pouvez laisser en blanc le champ « Chemins de classe » si vous avez ajouté le chemin d'accès du fichier .jar à la variable d'environnement CLASSPATH du système d'exploitation (voir aussi Configuration du CLASSPATH 188 ).
4. Dans le champ « Pilote », choisir **org.firebirdsql.jdbc.FBDriver**. Veuillez noter que cette entrée est disponible si un chemin d'accès de fichier .jar valide est trouvé soit dans le champ « Chemins de classe », soit dans la variable d'environnement CLASSPATH du système d'exploitation (voir l'étape précédente).




5. Saisir le nom d'utilisateur et le mot de passe de la base de données dans les champs correspondants.
6. Saisir la chaîne de connexion du serveur de base de données dans le champ de saisie URL de base de données, en remplaçant les valeurs soulignées avec celles applicables à votre serveur de base de données.

```
jdbc:firebirdsql://<host>[:<port>]/<database path or alias>
```

7. Cliquer sur **Se connecter**.

### 4.2.1.10.2 Firebird (ODBC)

Cet exemple illustre comment se connecter à une base de données Firebird 2.5.4 fonctionnant sur un serveur Linux.

**Conditions préalables :**

- JRE (Java Runtime Environment) ou Java Development Kit (JDK) doit être installé. Il peut s'agir soit d'Oracle JDK soit d'un build open source comme Oracle OpenJDK. MapForce déterminera le chemin vers la Java Virtual Machine (JVM) depuis les emplacements suivants, dans l'ordre suivant : a) le chemin personnalisé JVM que vous avez éventuellement défini dans l'application **Options**, voir Paramètres Java[1091]; b) le chemin JVM trouvé dans le registre Windows ; c) la variable d'environnement `JAVA_HOME`.
- Le pilote Firebird ODBC doit être installé sur votre système d'exploitation. Cet exemple utilise le pilote Firebird ODBC version 2.0.3.154 téléchargé depuis le site web Firebird ( https://www.firebirdsql.org/ ).
- Le client Firebird doit être installé sur votre système d'exploitation. Veuillez noter qu'aucun installateur autonome n'est disponible pour le client Firebird 2.5.4 ; le client fait partie du paquet d'installation du serveur Firebird. Vous pouvez télécharger le paquet d'installation du serveur Firebird depuis le site web Firebird ( https://www.firebirdsql.org/ ), chercher « Windows executable installer for full Superclassic/Classic or Superserver ». Pour n'installer que les fichiers client, choisir « **Minimum client install - no server, no tools** » lorsque vous suivrez les instructions de l'assistant.

  **Important :**

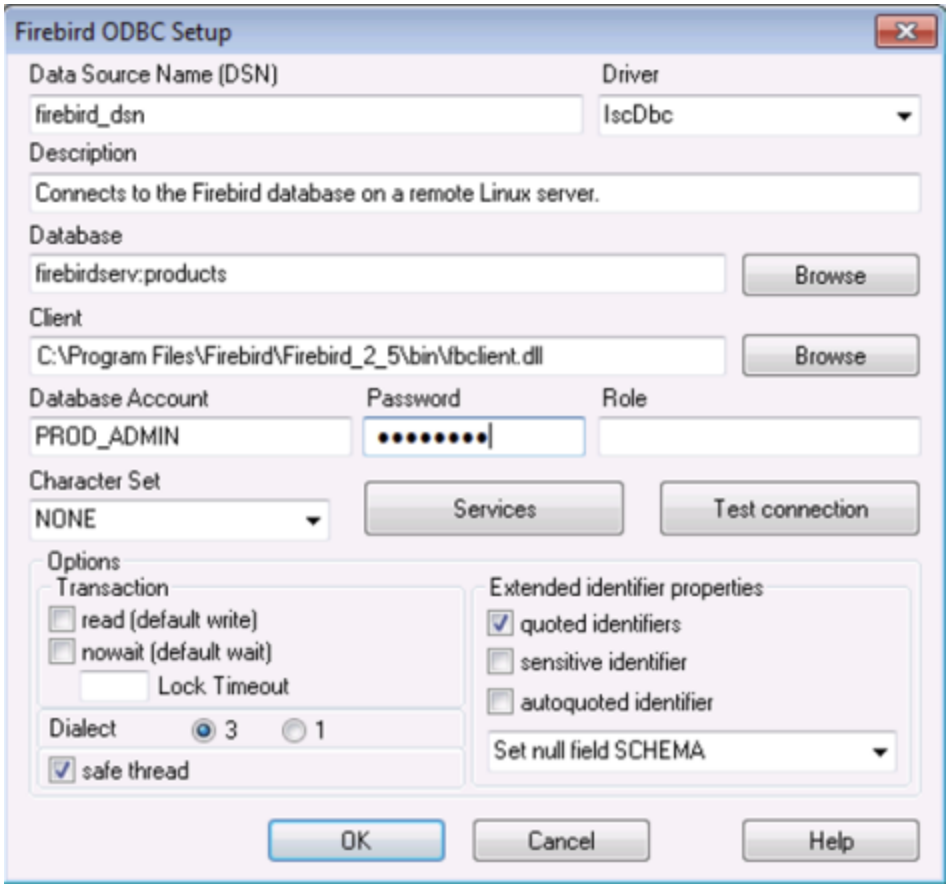
  - La plateforme du pilote Firebird ODBC et du client (32-bit ou 64-bit) doit correspondre à celle de MapForce.
  - La version du client Firebird doit correspondre à la version du serveur Firebird auquel vous vous connectez.
- Vous disposez des détails de connexion à la base de données suivantes : le nom d'hôte du serveur ou l'adresse IP, le chemin de base de données (ou alias) sur le serveur, le nom d'utilisateur et le mot de passe.

**Pour se connecter à Firebird via ODBC :**

1. Lancer l'assistant de connexion de base de données[163].
2. Cliquer sur **Connexions ODBC**.
3. Choisir **DSN utilisateur** (ou **DSN système**, si vous disposez de privilèges d'administrateur), puis cliquez sur **Ajouter** .


4. Choisir le pilote Firebird, puis cliquer sur **DSN utilisateur** (ou **DSN système**, selon ce que vous aurez sélectionné dans l'étape précédente). Si le pilote Firebird n'est pas disponible dans la liste, veuillez

vous assurer qu'il est installé dans votre système d'exploitation (voir aussi Consulter les pilotes ODBC disponibles 184 ).




5. Saisir les détails de connexion de base de données comme suit :

| | |
|---|---|
| *Nom de source de données (DSN)* | Saisir un nom descriptif pour la source de données que vous êtes en train de créer. |
| *Base de données* | Saisir le nom d'hôte du serveur ou l'adresse IP, suivi d'un double point, suivi d'un alias de base de données (ou chemin). Dans cet exemple, le nom d'hôte est `firebirdserv`, et l'alias de base de données est `products`, comme suit :<br><br>`firebirdserv:products`<br><br>L'utilisation d'une base de données assume que, du côté serveur, l'administrateur de la base de données a configuré les *produits* d'alias pour pointer vers le fichier de base de données Firebird (.fdb) sur le serveur (voir la documentation Firebird pour plus de détails).<br><br>Vous pouvez également utiliser l'adresse IP du serveur à la place du nom d'hôte et un chemin à la place d'un alias ; donc n'importe laquelle des chaînes de connexion d'échantillon est valide : |

| | |
|---|---|
| | `firebirdserver:/var/Firebird/databases/butterflies.fdb`<br>`127.0.0.1:D:\Misc\Lenders.fdb`<br><br>Si la base de données se trouve sur la machine Windows locale, cliquer sur **Parcourir** et sélectionner directement le fichier de base de données Firebird (.fdb). |
| *Client* | Saisir le chemin vers le fichier **fbclient.dll**. Par défaut, il s'agit du sous-répertoire `bin` du répertoire d'installation Firebird. |
| *Compte de base de données* | Saisir le nom d'utilisateur de base de données fourni par l'administrateur de la base de données (dans cet exemple, `PROD_ADMIN`). |
| *Mot de passe* | Saisir le mot de passe de la base de données fourni par l'administrateur de la base de données. |

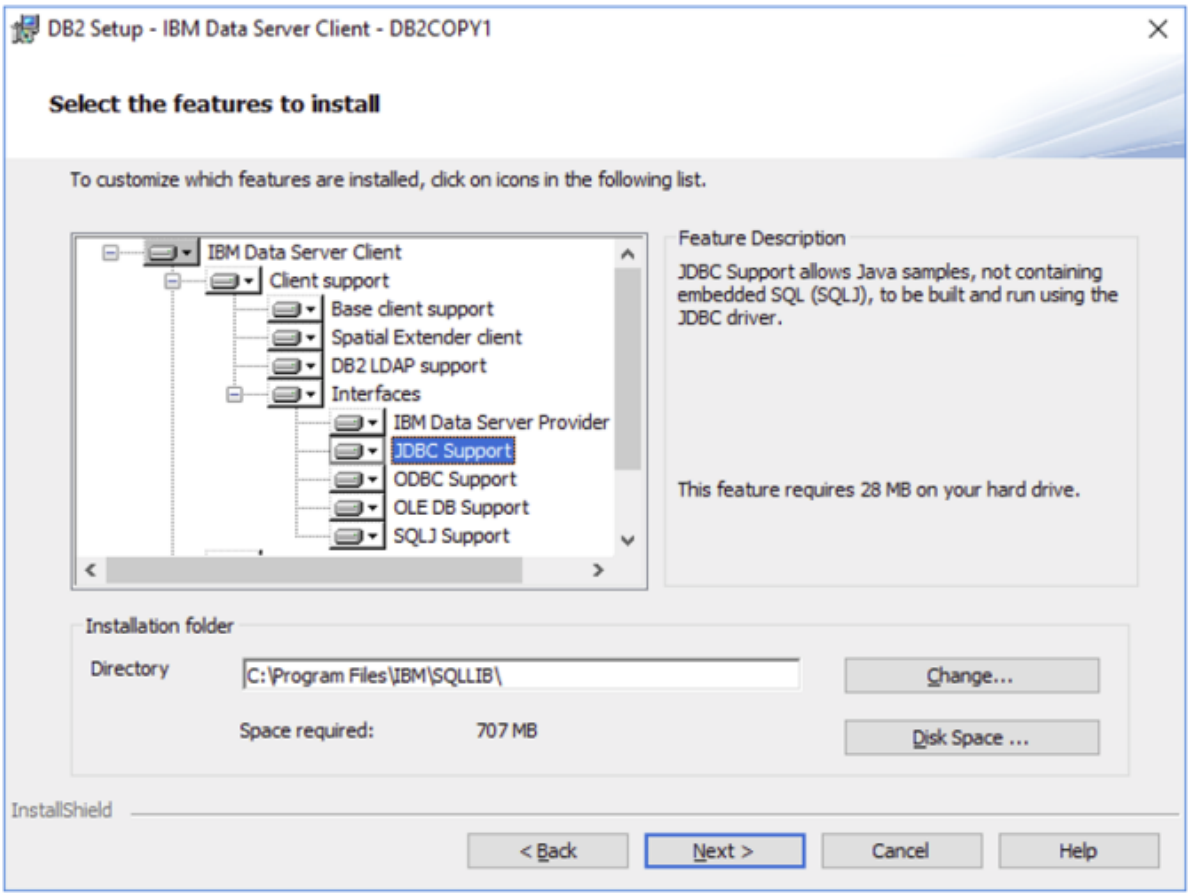
6. Cliquer sur **OK**.

### 4.2.1.10.3 IBM DB2 (JDBC)

Cet exemple illustre comment se connecter à un serveur de base de données **IBM DB2** par le biais de JDBC.

**Conditions préalables :**

- JRE (Java Runtime Environment) ou Java Development Kit (JDK) doit être installé. Il peut s'agir soit d'Oracle JDK soit d'un build open source comme Oracle OpenJDK. MapForce déterminera le chemin vers la Java Virtual Machine (JVM) depuis les emplacements suivants, dans l'ordre suivant : a) le chemin personnalisé JVM que vous avez éventuellement défini dans l'application **Options**, voir Paramètres Java[1091]; b) le chemin JVM trouvé dans le registre Windows ; c) la variable d'environnement `JAVA_HOME`.
- Veuillez vous assurer que la plateforme de MapForce (32-bit, 64-bit) corresponde à celle du JRE/JDK. Cet exemple utilise l'OpenJDK 11.0 64-bit d'Oracle, et, par conséquent, la version 64-bit de MapForce.
- Le pilote JDBC (un ou plusieurs fichiers .jar qui fournissent une connectivité dans la base de données) doit être disponible sur votre système d'exploitation. Cet exemple utilise le pilote JDBC disponible après avoir installé la version 10.1 (64-bit) **IBM Data Server Client**. Pour installer les pilotes JDBC, choisir une installation **Typical**, ou sélectionner cette option explicitement sur l'assistant d'installation.

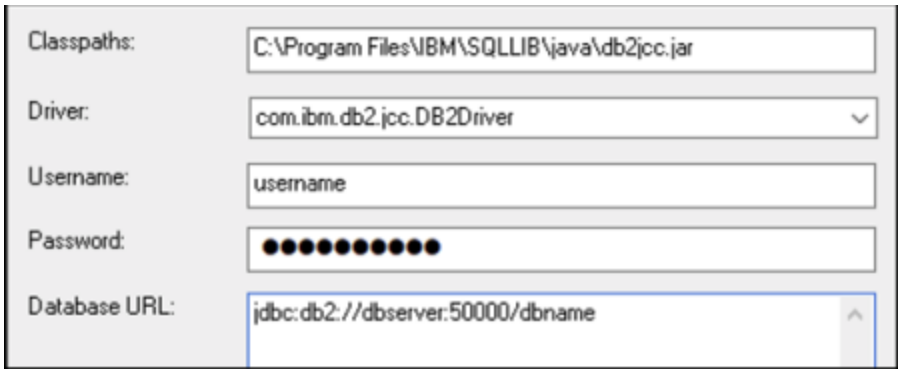



Si vous n'avez pas changé le chemin d'installation par défaut, les fichiers .jar requis se trouvent dans le répertoire **C:\Program Files\IBM\SQLLIB\java** après l'installation.

- Vous nécessitez les détails de connexion de base de données suivants : hôte, port, nom d'utilisateur et mot de passe.

**Pour se connecter à IBM DB2 par le biais de JDBC :**

1. Lancer l'assistant de connexion de base de données[163].
2. Cliquer sur **JDBC Connections**.
3. À côté de « Classpaths », saisir le chemin menant au fichier .jar qui fournit la connexion à la base de données. Cet exemple se réfère à **C:\Program Files\IBM\SQLLIB\java\db2jcc.jar.** Vous devrez éventuellement vous référer au pilote **db2jcc4.jar**, selon la version du serveur de base de données. Pour une compatibilité de pilote, se référer à la documentation IBM (http://www-01.ibm.com/support/docview.wss?uid=swg21363866). Veuillez noter que vous pouvez laisser le champ de texte "Classpaths" vide si vous avez ajouté les chemins de fichier .jar dans la variable d'environnement CLASSPATH du système d'exploitation (voir aussi Configurer le CLASSPATH[188]).
4. Dans la case "Driver", choisir **com.ibm.db2.jcc.DB2Driver**. Cette entrée devient disponible uniquement si un chemin de fichier .jar valide a été trouvé soit dans le champ de texte "Classpaths", soit dans la variable d'environnement CLASSPATH du système d'exploitation (voir l'étape précédente).




5. Saisir le nom d'utilisateur et le mot de passe de l'utilisateur de la base de données dans le champ de texte correspondant.
6. Saisir le string de connexion JDBC dans le champ de texte de l'**URL de base de données**. Veuillez vous assurer de remplacer les détails de connexion avec ceux applicable dans votre serveur de base de données.

```
jdbc:db2://hostName:port/databaseName
```

7. Cliquer sur **Se connecter**.

### 4.2.1.10.4 IBM DB2 (ODBC)

Cet exemple illustre comment se connecter à une base de données IBM DB2 via ODBC.

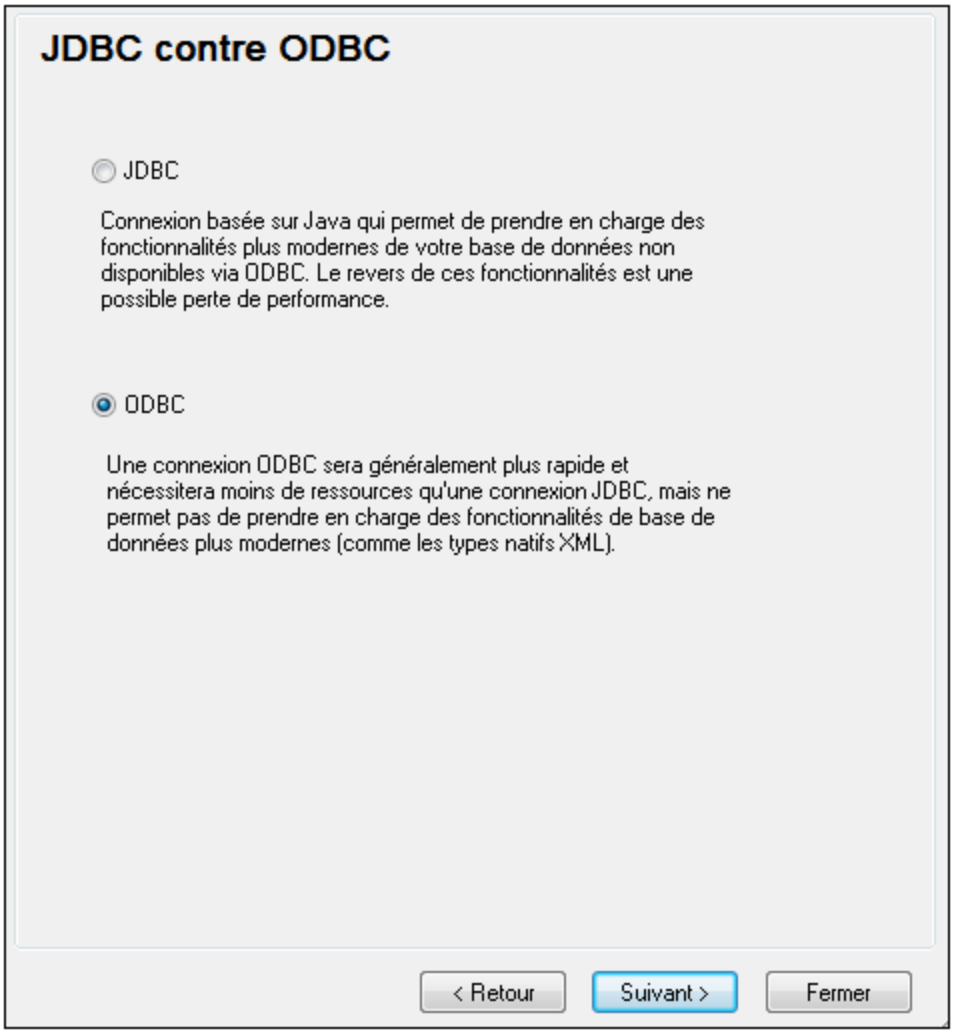
**Conditions préalables :**

- L'IBM Data Server Client doit être installé et configuré sur votre système d'exploitation (cet exemple utilise IBM Data Server Client 9.7). Pour travailler avec des instructions d'installation, veuillez consulter la documentation fournie avec votre logiciel IBM DB2. Après avoir installé IBM Data Server Client, vérifiez si les pilotes ODBC sont disponibles sur votre machine (voir Consulter les pilotes ODBC disponibles 184).
- Créer un alias de base de données. Vous pouvez procéder de plusieurs manières :
  - Depuis l'Assistant de configuration IBM DB2
  - Depuis le IBM DB2 Command Line Processor
  - Depuis l'assistant de source de données ODBC (dans ce cas, les instructions sont affichées ci-dessous)
- Vous disposez des détails de connexion de base de données suivants : hôte, base de données, port, nom d'utilisateur et mot de passe.

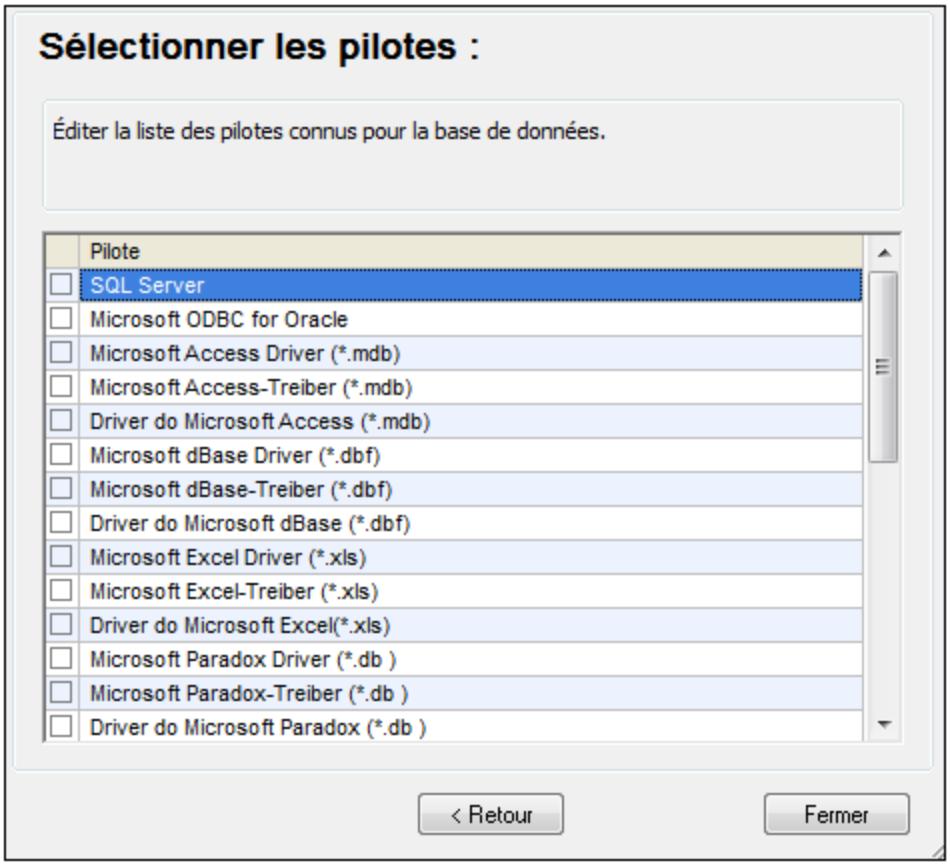
**Pour se connecter à IBM DB2 :**

1. Lancer l'assistant de connexion de base de données 163 et sélectionner **IBM DB2 (ODBC/JDBC)**.
2. Cliquer sur **Suivant**.

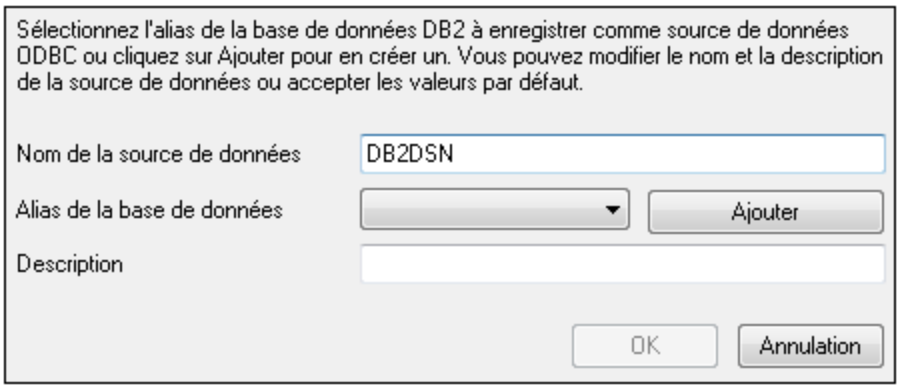



3. Choisir **ODBC** et cliquer sur **Suivant**. Si vous êtes invité à éditer la liste des pilotes connus pour la base de données, sélectionnez les pilotes de base de données applicables pour IBM DB2 (voir les Conditions préalables[200]), et cliquer sur **Suivant**.

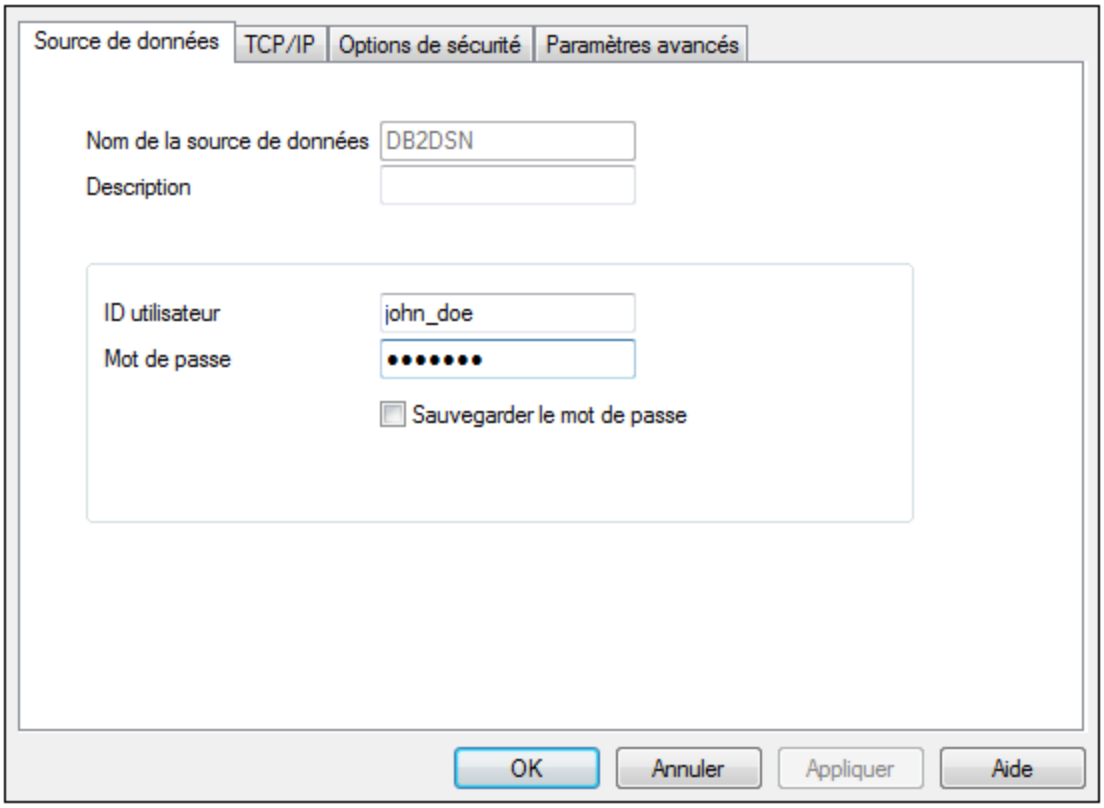



4. Choisir le pilote IBM DB2 depuis la liste et cliquer sur **Se connecter**. (Pour éditer la liste des pilotes disponibles, cliquer sur **Éditer pilotes**, puis cocher ou décocher les pilotes IBM DB2 que vous souhaitez ajouter ou supprimer, respectivement.)

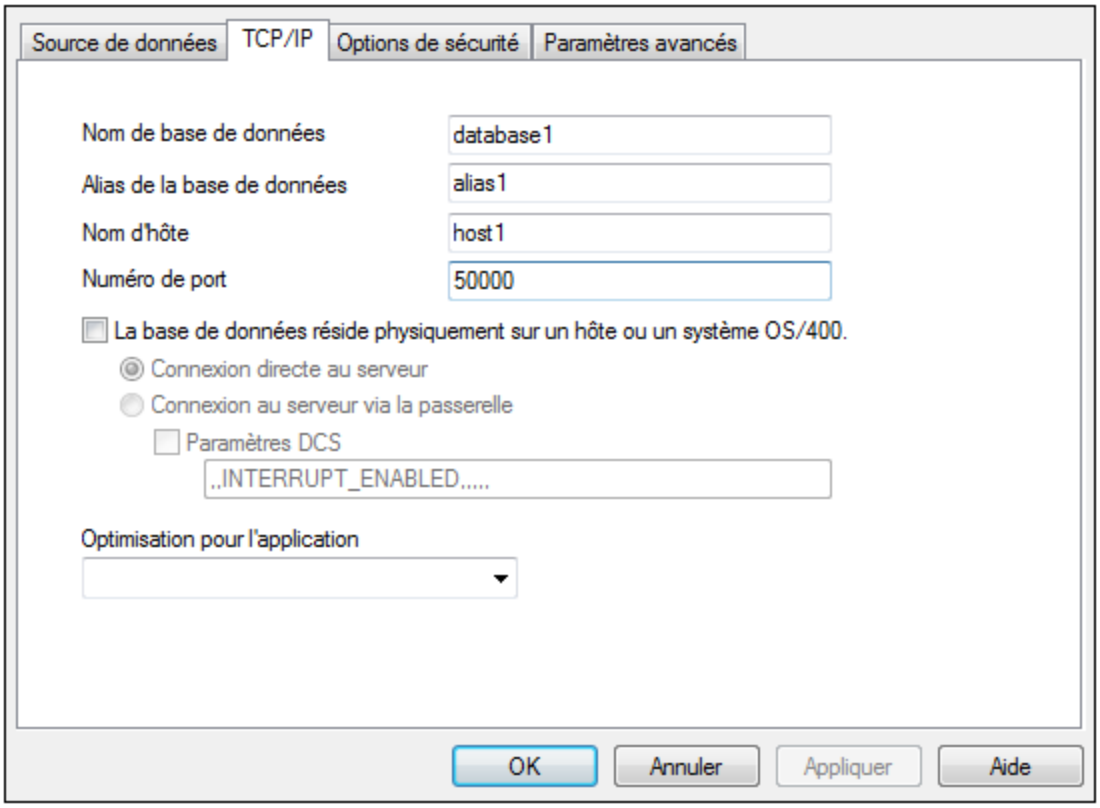

5. Saisir un nom de source de données (dans cet exemple, **DB2DSN**), puis cliquer sur **Ajouter**.



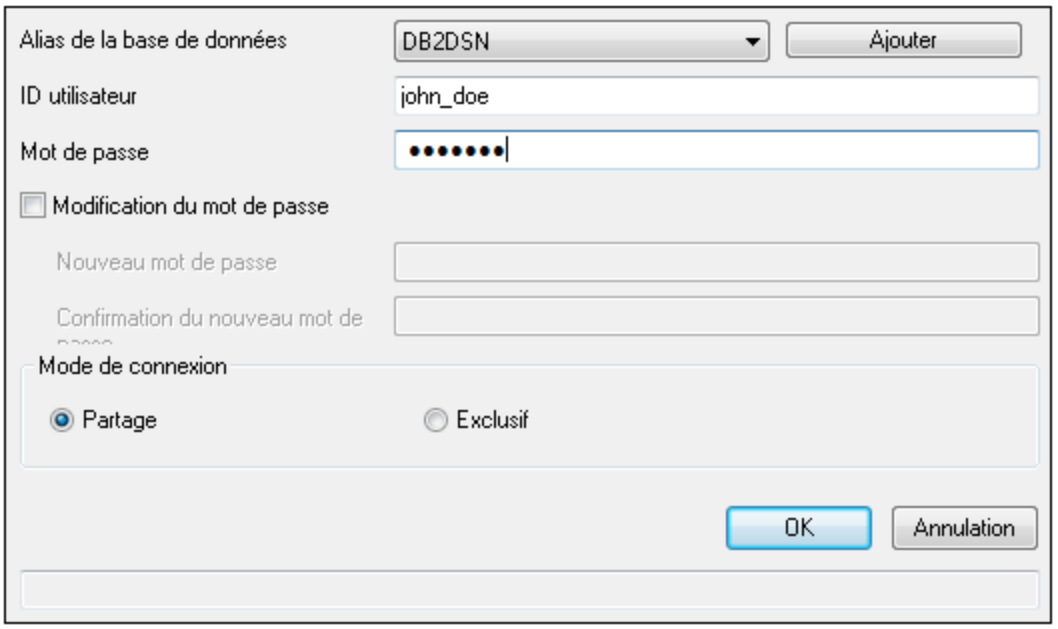

6. Dans l'onglet **Source de données**, saisir le nom d'utilisateur et le mot de passe de la base de données.




7. Dans l'onglet **TCP/IP**, saisir le nom de base de données, un nom pour l'alias, le nom d'hôte et le numéro de port, puis cliquer sur OK.




8. Saisir à nouveau le nom d'utilisateur et le mot de passe puis cliquer sur **OK**.



### 4.2.1.10.5 IBM DB2 pour i (JDBC)

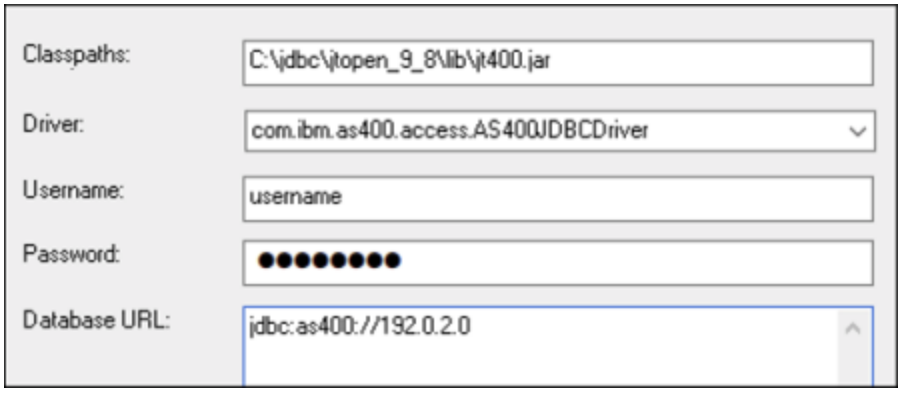
Cet exemple illustre comment se connecter à un serveur de base de données **IBM DB2 for i** par le biais de JDBC.

**Exigences préalables :**

- JRE (Java Runtime Environment) ou Java Development Kit (JDK) doit être installé. Il peut s'agir soit d'Oracle JDK soit d'un build open source comme Oracle OpenJDK. MapForce déterminera le chemin vers la Java Virtual Machine (JVM) depuis les emplacements suivants, dans l'ordre suivant : a) le chemin personnalisé JVM que vous avez éventuellement défini dans l'application **Options**, voir Paramètres Java[1091]; b) le chemin JVM trouvé dans le registre Windows ; c) la variable d'environnement `JAVA_HOME`.
- Veuillez vous assurer que la plateforme de MapForce (32-bit, 64-bit) corresponde à celle du JRE/JDK. Cet exemple utilise l'OpenJDK 11.0 64-bit d'Oracle, et, par conséquent, la version 64-bit de MapForce.
- Le pilote JDBC (un ou plusieurs fichiers .jar qui offrent une connectivité à la base de données) doit être disponible sur votre système d'exploitation. Cet exemple utilise la source ouverte **Toolbox for Java/JTOpen** version 9.8 (http://jt400.sourceforge.net/). Après avoir installé le package et l'avoir décompressé vers un répertoire local, les fichiers .jar requis seront disponibles dans le sous-répertoire **lib**.
- Vous nécessitez les détails de connexion de base de données suivants : hôte, port, nom d'utilisateur et mot de passe.

**Pour se connecter à IBM DB2 pour i par le biais de JDBC :**

1. Lancer l'assistant de connexion de base de données[163].
2. Cliquez sur **JDBC Connections**.
3. À côté de « Classpaths », saisissez le chemin menant au fichier .jar qui fournit une connectivité à la base de données. Dans cet exemple, le fichier .jar exigé se trouve sous le chemin suivant : **C:\jdbc\jtopen_9_8\jt400.jar**. Veuillez noter que vous pouvez laisser en blanc le champ « Chemins de classe » si vous avez ajouté le(s) chemin(s) d'accès du fichier .jar à la variable d'environnement CLASSPATH du système d'exploitation (voir aussi Configuration du CLASSPATH[188]).
4. Dans la case« Driver », choisir **com.ibm.as400.access.AS400JDBCDriver**. Cette entrée devient disponible uniquement si un chemin de fichier .jar valide a été trouvé soit dans le champ de texte « Classpaths », soit dans la variable d'environnement CLASSPATH du système d'exploitation (voir l'étape précédente).



5. Saisissez le nom d'utilisateur et le mot de passe de l'utilisateur de la base de données dans la zone de texte correspondante.
6. Saisissez le string de connexion JDBC dans la zone de texte **Base de données URL** Assurez-vous de remplacer l'**hôte** par le nom de l'hôte ou l'adresse IP de votre serveur de base de données.

```
jdbc:as400://host
```

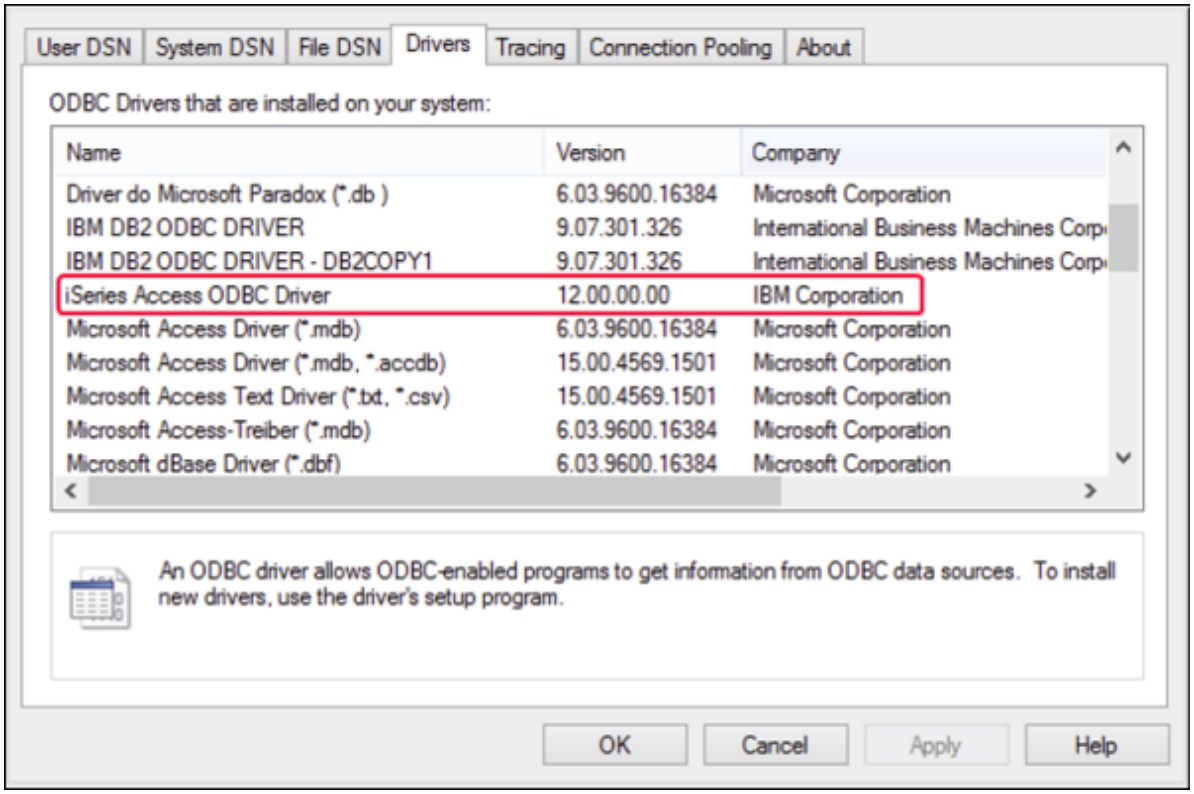
7. Cliquez sur **Connexion**.

### 4.2.1.10.6 IBM DB2 pour i (ODBC)

Cet exemple illustre comment se connecter à une base de données *IBM DB2 pour i* par le biais d'ODBC.

**Exigences préalables :**

- *IBM System i Access pour Windows* doit être installé sur votre système d'exploitation (cet exemple utilise *IBM System i Access pour Windows V6R1M0*). Pour les instructions d'installation, vérifiez la documentation fournie avec votre logiciel *IBM DB2 pour i*. Après l'installation, vérifier si le pilote ODBC est disponible sur votre appareil (voir Consulter les pilotes ODBC disponibles 184).

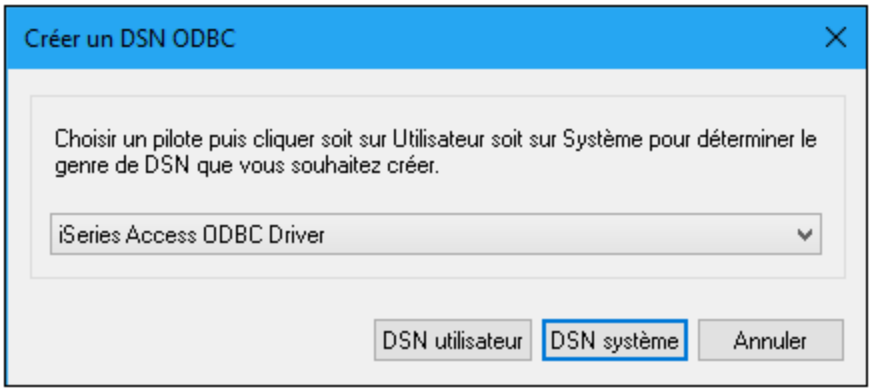



- Vous disposez des détails de connexion à la base de données suivants : l'adresse IP du serveur de la base de données, le nom d'utilisateur de la base de données et le mot de passe.
- Exécuter *System i Navigator* et suivez l'assistant pour créer une nouvelle connexion. Sur invitation à spécifier un système, saisissez l'adresse IP du serveur de la base de données. Après avoir établi la connexion, il est recommandé de la vérifier (cliquez sur la connexion, et choisissez **Fichier** >

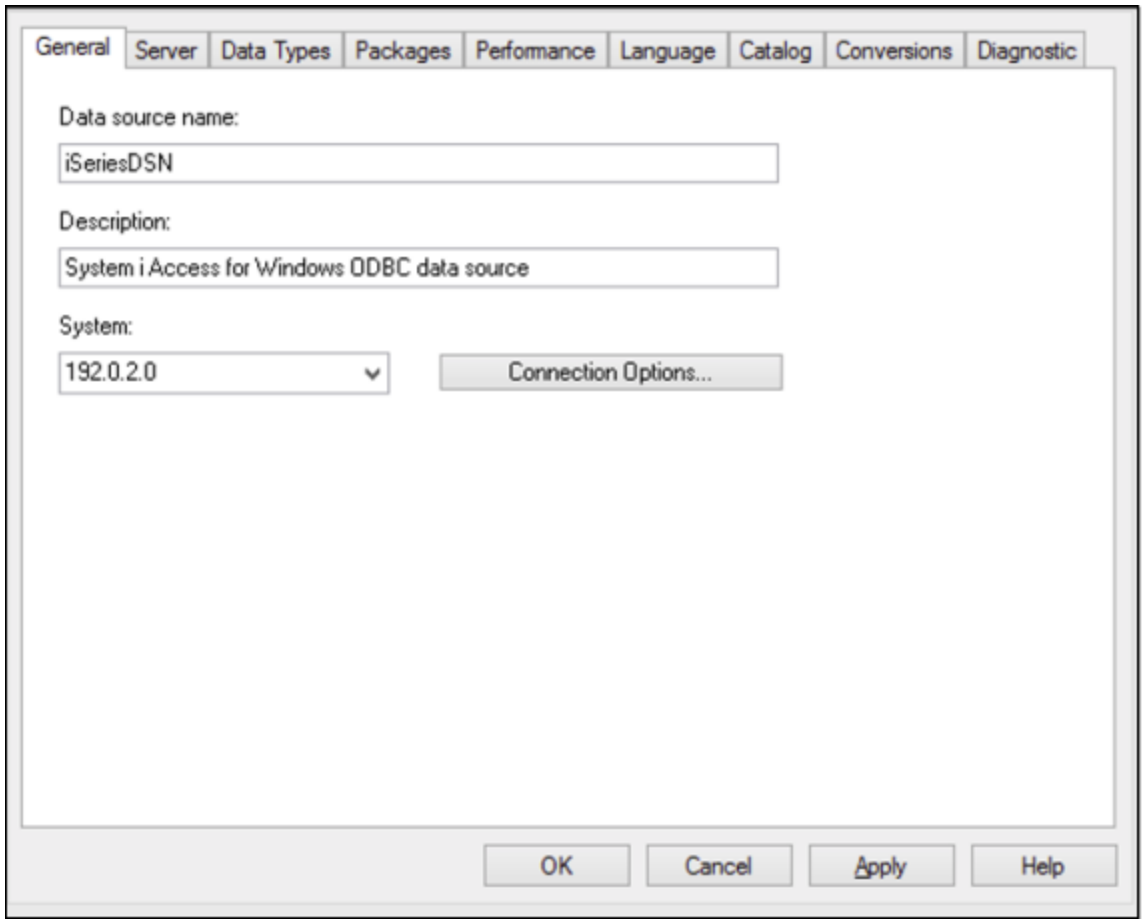
**Diagnostics** > **Vérifier connexion**). Si vous obtenez des erreurs de connectivité, contacter l'administrateur de serveur de la base de données.

### Pour se connecter à IBM DB2 for i:

1. Lancer l'assistant de connexion de base de données 163.
2. Cliquez sur **connexions ODBC**.
3. Cliquez sur**DSN Utilisateur** (en alternative, cliquez sur **DSN Système** ou **DSN Fichier**, dans un tel cas, les instructions suivantes seront semblables).
4. Cliquez sur **Ajouter** .
5. Sélectionnez le **pilote iSeries Access ODBC** depuis la liste et cliquez sur **DSN Utilisateur** (ou **DSN Système**, le cas échéant).

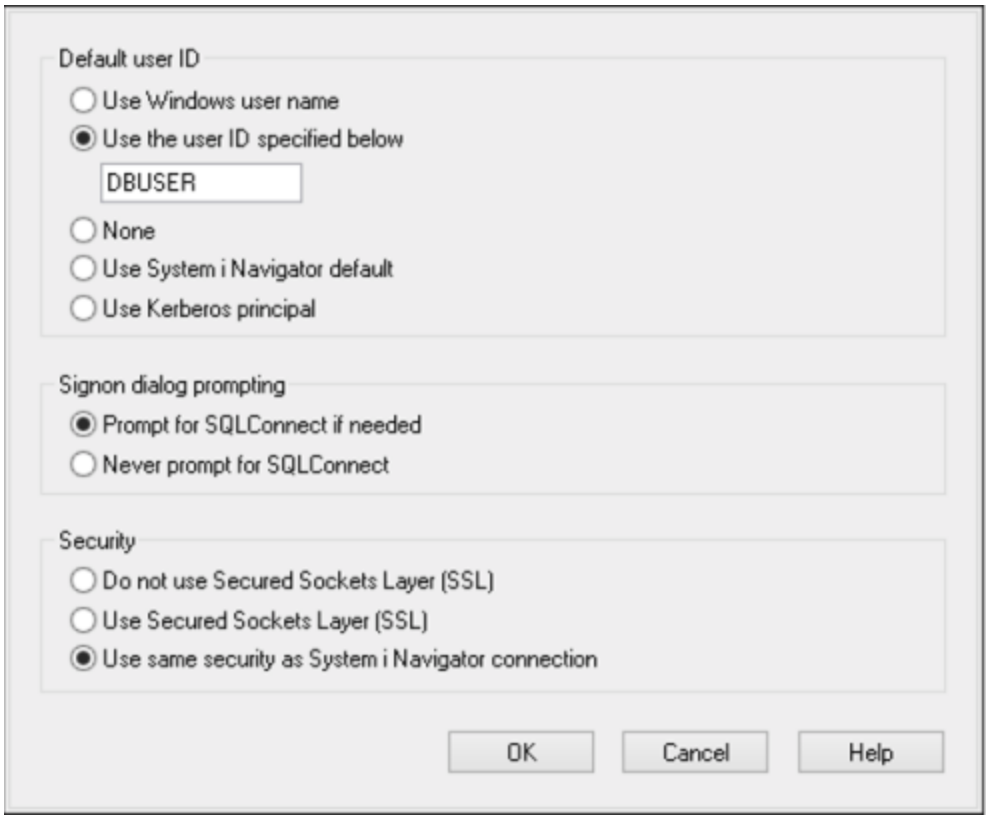



6. Saisissez un nom de source de données et sélectionnez la connexion depuis la zone de liste du Système. Dans cet exemple, le nom de source de données est **iSeriesDSN** et le Système est **192.0.2.0**.




**Note :** en ajoutant une source de données ODBC pour une base de données *IBM DB2 pour i*, un indicateur par défaut est défini qui active des timeouts de requête. Ce paramètre doit être désactivé pour MapForce afin de charger correctement des fichiers de mappage. Pour désactiver le paramètre, sélectionnez l'onglet **Performance**, cliquez sur **Avancé**, et videz la case à cocher **Permettre un timeout de requête**.

7. Cliquez sur les options de connexion, sélectionnez **Utiliser l'ID utilisateur spécifiée ci-dessous** et saisissez le nom de l'utilisateur de la base de données (dans cet exemple, **DBUSER**).




8. Cliquez sur **OK**. La nouvelle source de données devient disponible dans la liste des DSN.
9. Cliquez sur **Connexion**.
10. Saisissez le nom utilisateur et le mot de passe de la base de données quand vous y êtes invité, puis cliquez sur **OK**.

### 4.2.1.10.7 IBM Informix (JDBC)

Cet exemple illustre comment se connecter à un serveur de base de données IBM Informix par le biais de JDBC.

**Exigences préalables :**

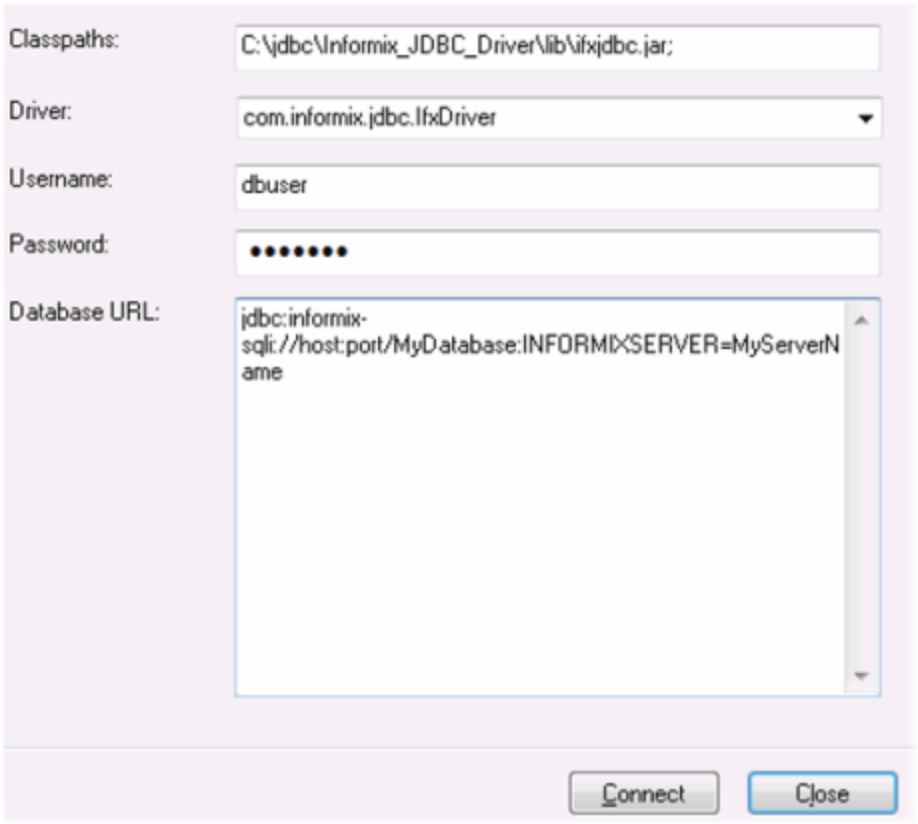
- JRE (Java Runtime Environment) ou Java Development Kit (JDK) doit être installé. Il peut s'agir soit d'Oracle JDK soit d'un build open source comme Oracle OpenJDK. MapForce déterminera le chemin vers la Java Virtual Machine (JVM) depuis les emplacements suivants, dans l'ordre suivant : a) le chemin personnalisé JVM que vous avez éventuellement défini dans l'application **Options**, voir Paramètres Java 1091; b) le chemin JVM trouvé dans le registre Windows ; c) la variable d'environnement `JAVA_HOME`.
- Veuillez vous assurer que la plateforme de MapForce (32-bit, 64-bit) correspond à celle du JRE/JDK.
- Le pilote JDBC (un ou plusieurs fichiers .jar qui offrent une connectivité à la base de données) doit être disponible sur votre système d'exploitation. Dans cet exemple, le pilote IBM Informix JDBC version 3.70 est utilisé. En ce qui concerne les instructions d'installation du pilote, voir la documentation fournie avec le pilote ou bien le manuel « IBM Informix JDBC Driver Programmer's Guide »).

- Vous disposez des détails de connexion à la base de données suivants : hôte, nom du serveur Informix, base de données, port, nom d'utilisateur et mot de passe.

### Pour se connecter à IBM Informix par le biais de JDBC :

1. Lancer l'assistant de connexion de base de données 163.
2. Cliquez sur **JDBC Connections**.
3. À côté de « Classpaths », saisissez le chemin menant au fichier .jar qui fournit une connectivité à la base de données. Si nécessaire, vous pouvez aussi saisir une liste séparée par points-virgules de chemins d'accès de fichiers .jar. Dans cet exemple, le fichier .jar exigé se trouve sous le chemin suivant : **C:\Informix_JDBC_Driver\lib\ifxjdbc.jar**. Veuillez noter que vous pouvez laisser en blanc le champ « Chemins de classe » si vous avez ajouté le chemin d'accès du fichier .jar à la variable d'environnement CLASSPATH du système d'exploitation (voir aussi Configuration du CLASSPATH 188 ).
4. Dans le champ « Pilote », sélectionnez **com.informix.jdbc.IfxDriver**. Veuillez noter que cette entrée est disponible si un chemin d'accès de fichier .jar valide est trouvé soit dans le champ « Chemins de classe », soit dans la variable d'environnement CLASSPATH du système d'exploitation (voir l'étape précédente).




5. Saisissez le nom d'utilisateur et le mot de passe de la base de données dans les champs de saisie correspondants.
6. Saisir la chaîne de connexion au serveur de la base de données dans la zone de texte URL de base de données, en remplaçant les valeurs soulignées avec celles applicables à votre serveur de base de données.

```
jdbc:informix-sqli://hostName:port/databaseName:INFORMIXSERVER=myserver;
```

7. Cliquez sur **Connexion**.

### 4.2.1.10.8 MariaDB (ODBC)

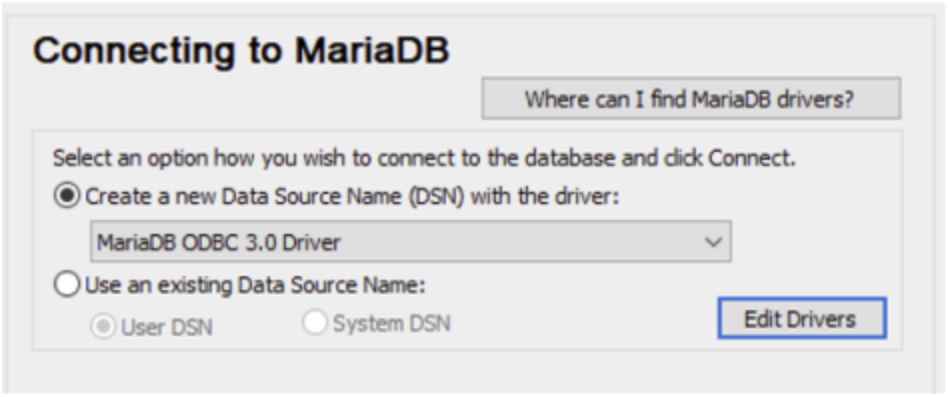
Cet exemple indique comment se connecter à un serveur de base de données MariaDB par le biais d'ODBC.

**Exigences préalables :**

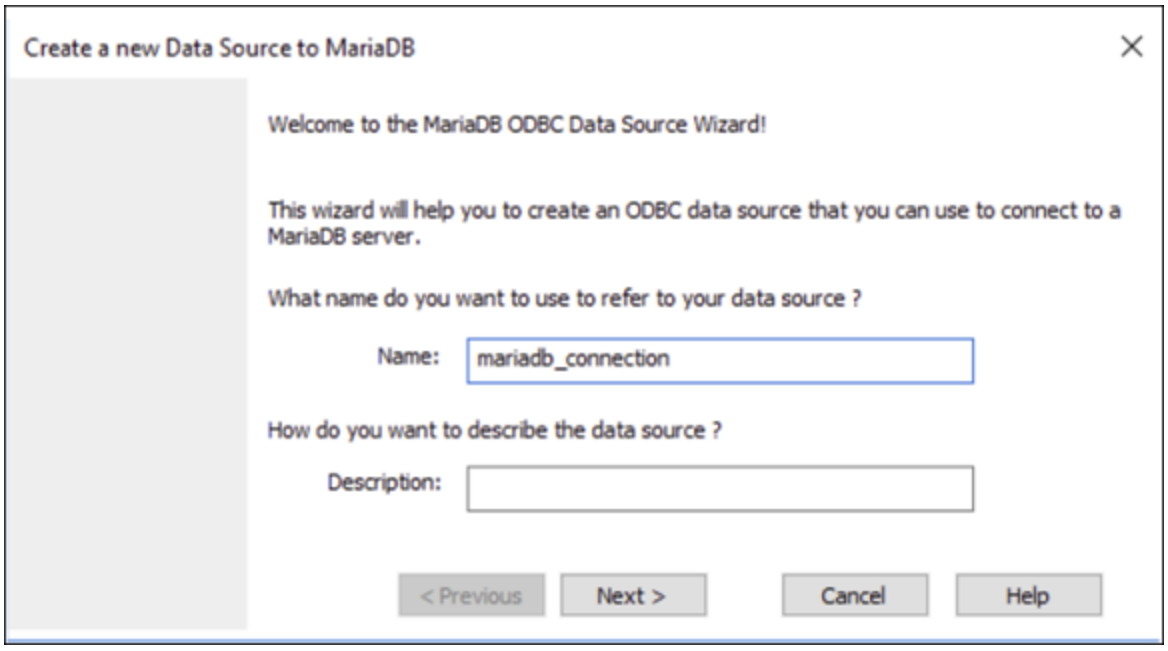
- Le connecteur/ODBC MariaDB (https://downloads.mariadb.org/connector-odbc/) doit être installé.
- Vous disposez des détails de connexion à la base de données suivants : hôte, base de données, port, nom utilisateur et mot de passe.

**Pour se connecter à MariaDB avec ODBC :**

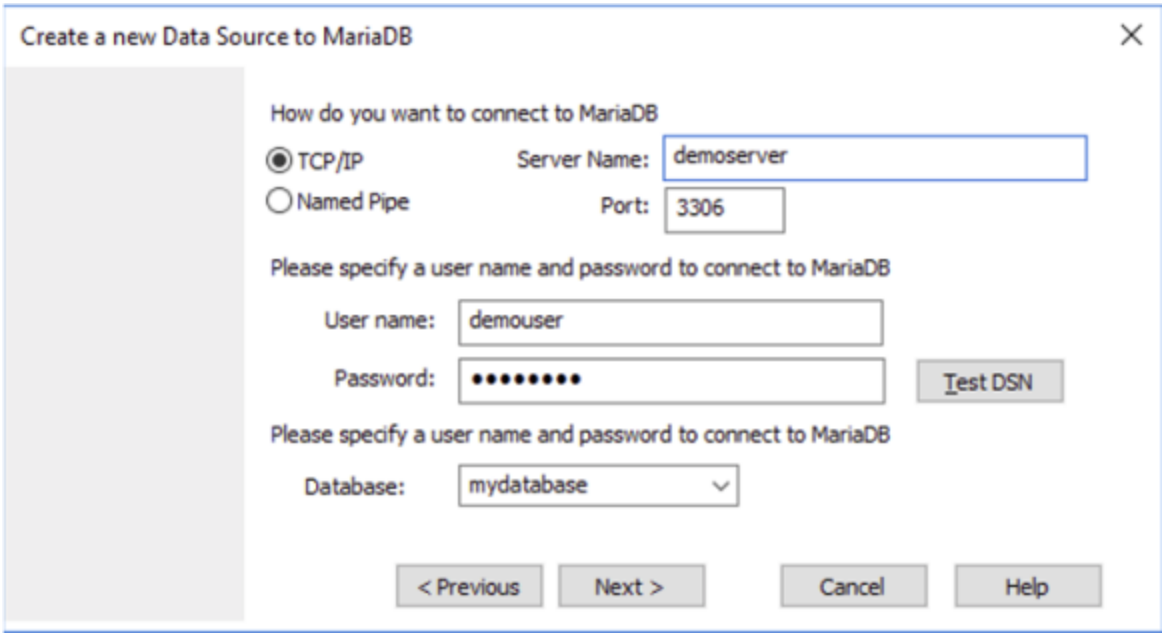
1. Lancer l'assistant de connexion de base de données[163].
2. Choisissez **MariaDB (ODBC)** et cliquez sur **Suivant**.



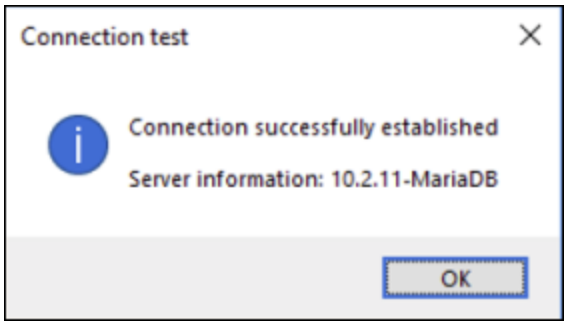

3. Choisissez **Créer un nouveau nom de source de données (DSN) avec le pilote** et choisissez le **pilote MariaDB ODBC 3.0**. Si ce pilote n'est pas disponible dans la liste, cliquez sur **Éditer pilotes**, et choisissez un des pilotes MariaDB disponibles (la liste contient tous les pilotes ODBC installés dans votre système d'opération).
4. Cliquez sur **Se connecter**.




5. Saisir le nom et, éventuellement, une description qui vous aidera à identifier cette source de données ODBC à l'avenir.




6. Remplissez les identifiants de connexion dans la base de données (Serveur TCP/IP, Utilisateur, Mot de passe), choisissez une base de données et cliquez sur **Tester DSN**. Une fois la connexion établie, un message apparaît :




7. Cliquez sur **Suivant** et terminez l'assistant. D'autres paramètres seront éventuellement nécessaires, selon les cas (par exemple, des certificats SSL si vous vous êtes connecté à MariaDB par le biais d'une connexion sécurisée).

**Note :** si le serveur de base de données est à distance, il doit être configuré par l'administrateur du serveur pour accepter les connexions à distance depuis l'adresse IP de votre appareil.

### 4.2.1.10.9 Microsoft Access (ADO)

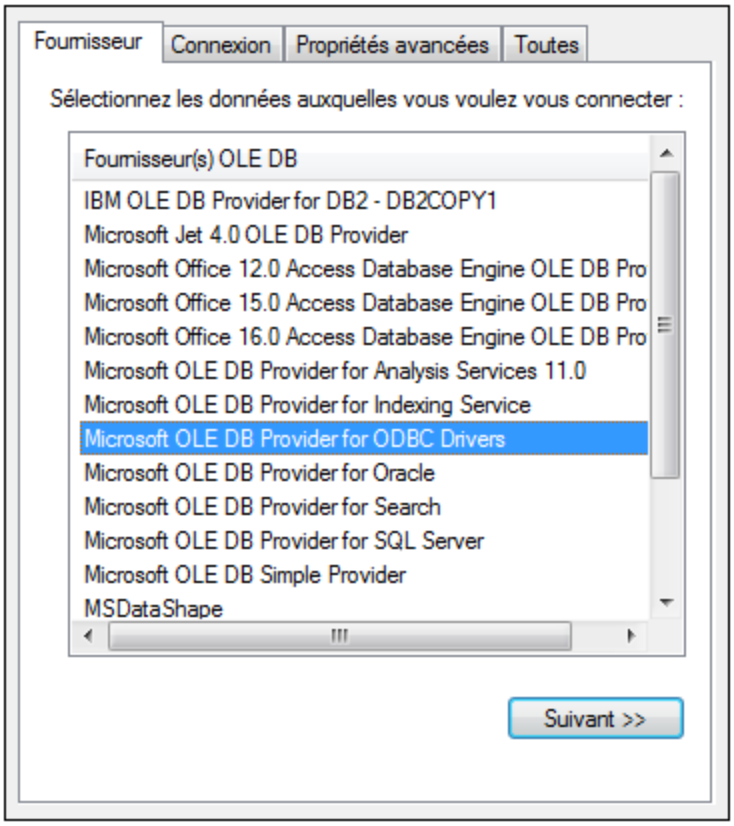
Une manière simple de se connecter à une base de données Microsoft Access est de suivre les instructions de l'assistant et de parcourir le fichier de base de données, tel que montré dans la section Connexion à une base de données Microsoft Access existante 171. Une approche alternative est de définir explicitement une connexion ADO, tel que montré dans cette section. Cette approche est utile si votre base de données est protégée par un mot de passe.

Il est également possible de se connecter à Microsoft Access par le biais d'une connexion ODBC, mais ce scénario a des limites, il vaut donc mieux l'éviter.

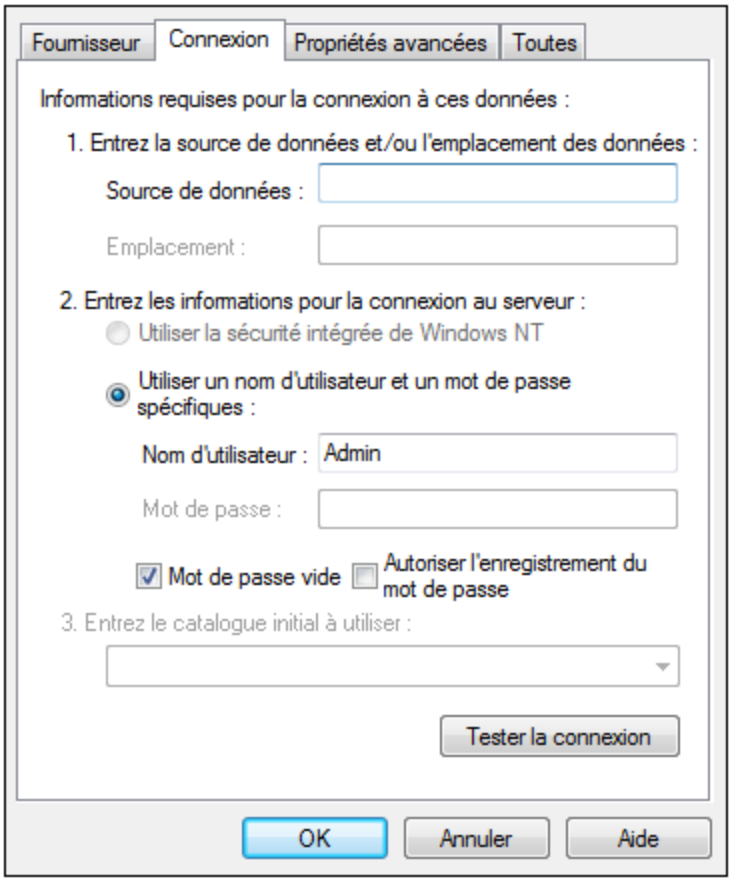
**Pour se connecter à une base de données Microsoft Access protégée par un mot de passe :**

1. Lancer l'assistant de connexion de base de données 163.
2. Cliquez sur **Connexions ADO**.
3. Cliquer sur **Générer**.

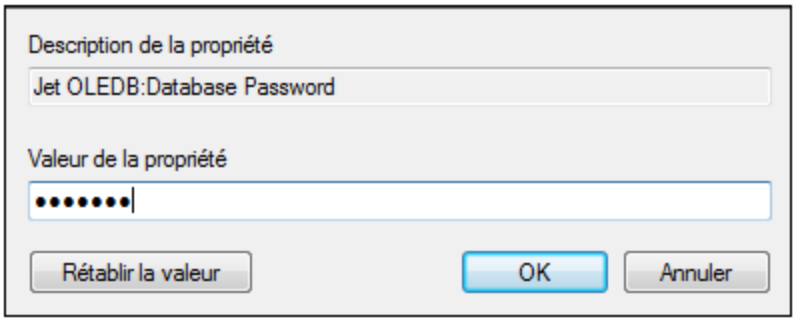
 




4. Sélectionnez le **fournisseur Microsoft Office 15.0 Access Database Engine OLE DB**, puis cliquez sur **Suivant**.




5. Dans la case de source de données, saisissez le chemin vers le fichier Microsoft Access dans le format UNC, par exemple **\\myserver\\mynetworkshare\Reports\Revenue.accdb**, où **myserver** est le nom du serveur et **mynetworkshare** est le nom du réseau partagé.
6. Dans l'onglet **Toutes**, double-cliquez la propriété **Jet OLEDB:Database Password** et saisissez le mot de passe de la base de données en tant que valeur de propriété.




**Note :** si vous ne parvenez toujours pas à vous connecter, localisez le fichier d'information du groupe de travail (**System.MDW**) applicable à votre profil d'utilisateur, et réglez la valeur de la propriété **Jet OLEDB : System database** sur le chemin d'accès du fichier **System.MDW**.

#### 4.2.1.10.10 Microsoft Azure SQL (ODBC)

Afin de vous connecter correctement à une base de données Azure SQL, vous devez installer le dernier SQL Server Native Client.

Pour information sur la connexion d'une base de données Azure SQL dans le cloud, voir cette entrée du blog Altova.

#### 4.2.1.10.11 Microsoft SQL Server (ADO)

Cet exemple indique comment se connecter à une base de données SQL Server avec ADO. Ces instructions sont applicables quand vous utilisez **Microsoft OLE DB Driver for SQL Server (MSOLEDBSQL)** recommandé, qui est disponible pour le téléchargement au lien https://docs.microsoft.com/en-us/sql/connect/oledb/download-oledb-driver-for-sql-server?view=sql-server-ver15.

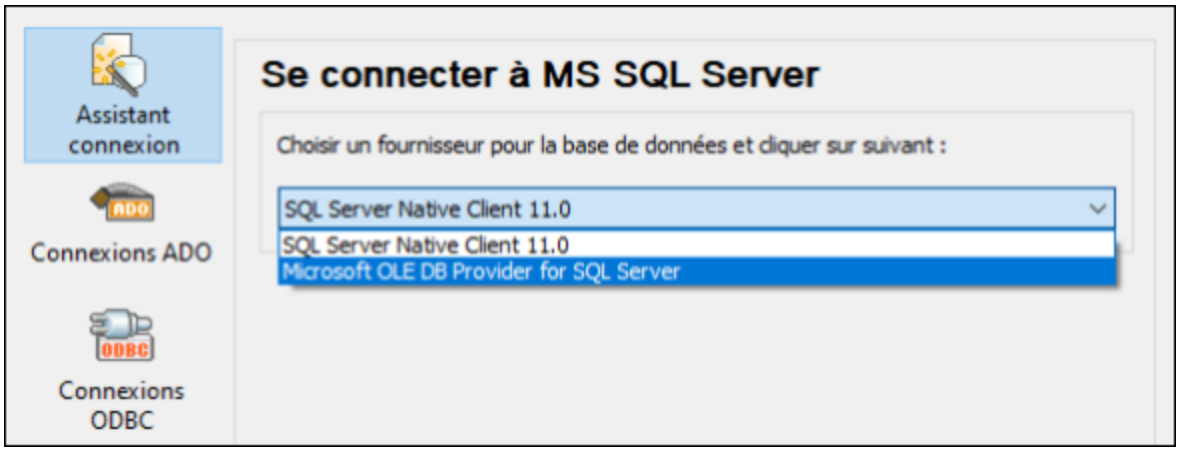
> Avant de suivre ces instructions, assurez-vous avoir téléchargé et installé le fournisseur ci-dessus sur votre poste de travail. Le fournisseur ADO doit correspondre à la plate-forme de MapForce (32-bit ou 64-bit).

Si vous souhaitez utiliser d'autres fournisseurs ADO tels que **SQL Server Native Client (SQLNCLI)** ou **Microsoft OLE DB Provider for SQL Server (SQLOLEDB)**, les instructions sont semblables, mais ces fournisseurs sont dépréciés et donc pas recommandés. Aussi, pour que le connexion réussisse avec un fournisseur, vous allez devoir définir des propriétés de connexion additionnelles telles que décrites dans Configurer les propriétés de SQL Server Data Link 172.

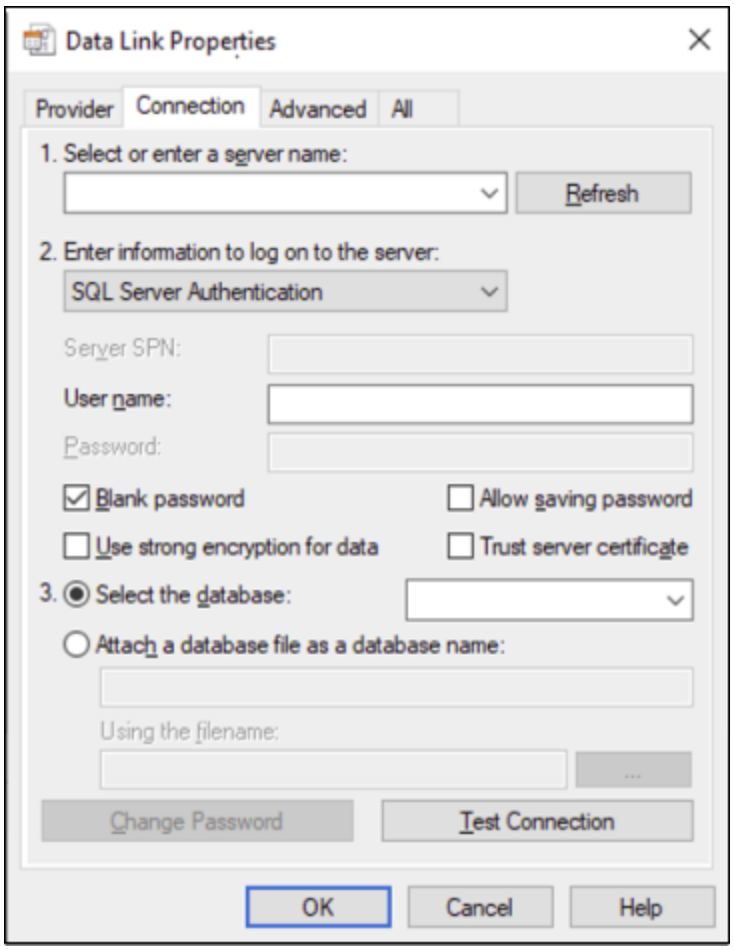
> **Microsoft OLE DB Provider for SQL Server (SQLOLEDB)** est connu pour avoir des soucis avec la liaison de paramètre de requêtes complexes comme Common Table Expressions (CTE) et des déclarations imbriquées SELECT.

**Pour se connecter à SQL Server :**

1. Lancer l'assistant de connexion de base de données 163.
2. Choisissez **Microsoft SQL Server (ADO)**, puis cliquez sur **Suivant**. La liste des fournisseurs ADO disponibles est affichée. Dans cet exemple, le **Microsoft OLE DB Driver for SQL Server** est utilisé. S'il ne figure pas dans la liste, assurez-vous qu'il est installé sur votre ordinateur, tel que mentionné ci-dessus.



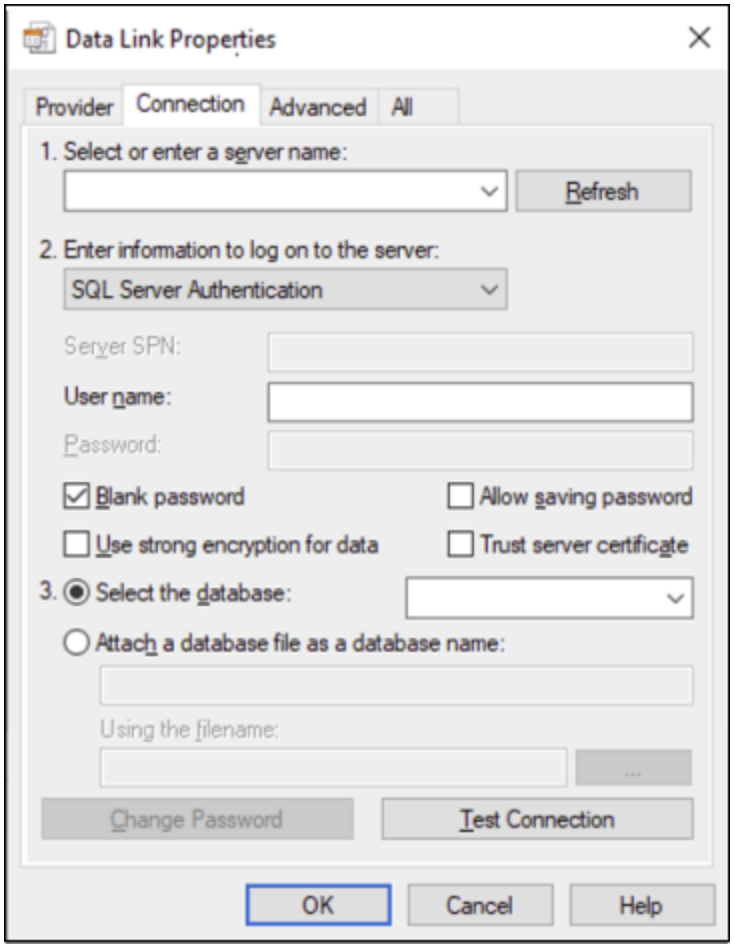

3. Cliquez sur **Next**. Le dialogue Data Link Properties apparaît.




4. Choisissez ou saisissez le nom du serveur de base de données, par exemple **SQLSERV01**. Si vous vous connectez à une instance SQL Server nommée, le nom du serveur ressemblera à **SQLSERV01\SOMEINSTANCE**.
5. Si le serveur de la base de données a été configuré pour permettre des connexions depuis des utilisateurs authentifiés dans le domaine Windows, choisissez **Windows Authentication**. Sinon,

sélectionnez **SQL Server Authentication**, décochez la case **Blank password** et saisissez les identifiants de base de données dans les champs pertinents.

6. Sélectionnez la case **Autoriser enregistrer mot de passe** et la base de données à laquelle vous vous connectez (dans cet exemple, « Nanonull »).




7. Pour tester la connexion à cet endroit, cliquez sur **Tester la connexion**. Il s'agit d'une étape optionnelle recommandée.
8. Cliquez sur **OK**.

### 4.2.1.10.12 Microsoft SQL Server (ODBC)

Cet exemple indique comment se connecter à une base de données SQL Server avec ODBC.

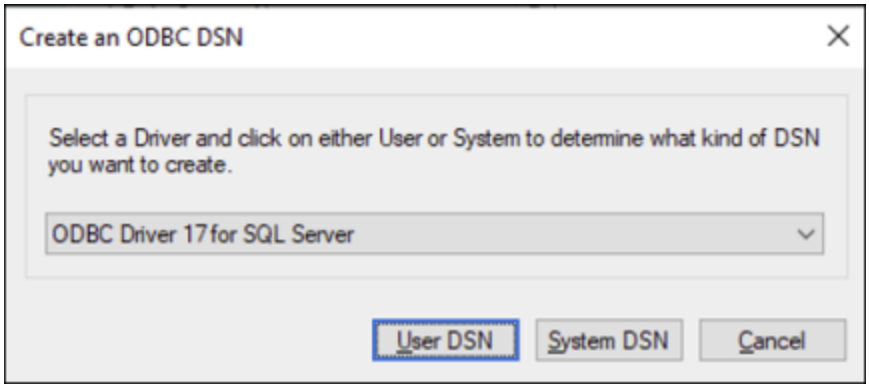
**Exigences préalables :**

- Pour télécharger et installer **Microsoft ODBC Driver for SQL Server** depuis le site web Microsoft, voir https://docs.microsoft.com/en-us/SQL/connect/odbc/download-odbc-driver-for-sql-server. Cet exemple utilise le **Pilote Microsoft ODBC 17 pour SQL Server** pour se connecter à une base de données **SQL Server 2016**. Vous pouvez tout à fait télécharger une version de pilote ODBC différente, selon la version du Serveur SQL auquel vous souhaitez vous connecter. Pour plus d'informations concernant les

versions de pilote ODBC prises en charge par votre base de données Serveur SQL, veuillez consulter les exigences de système du pilote.

### Pour se connecter à SQL Server avec ODBC :

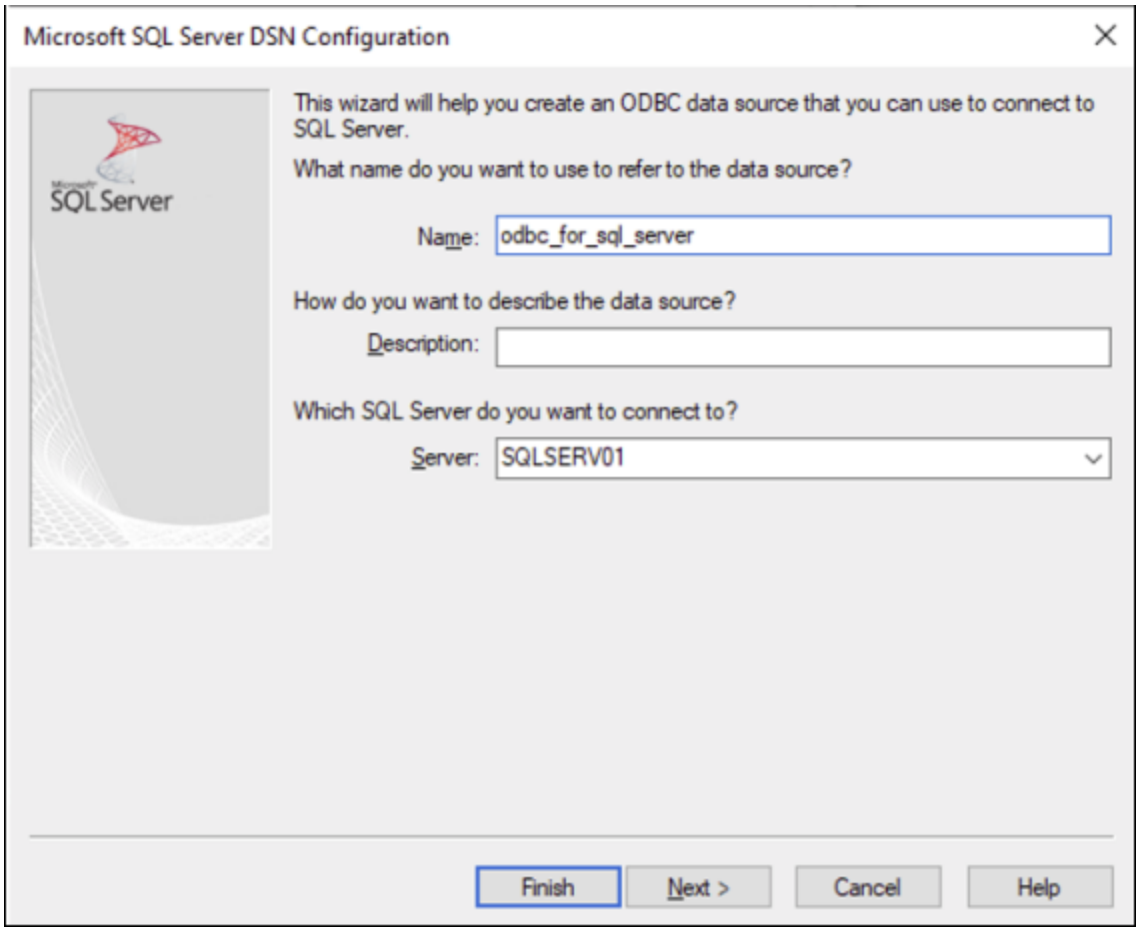
1. Lancer l'assistant de connexion de base de données 163.
2. Cliquez sur **ODBC Connections**.
3. Sélectionnez **DSN Utilisateur** (ou **DSN Système** si vous avez des privilèges d'administrateur), puis cliquez sur **Ajouter** .
4. Sélectionnez le pilote depuis la liste. Veuillez noter que le pilote apparaît dans la liste uniquement après qu'il ait été installé.



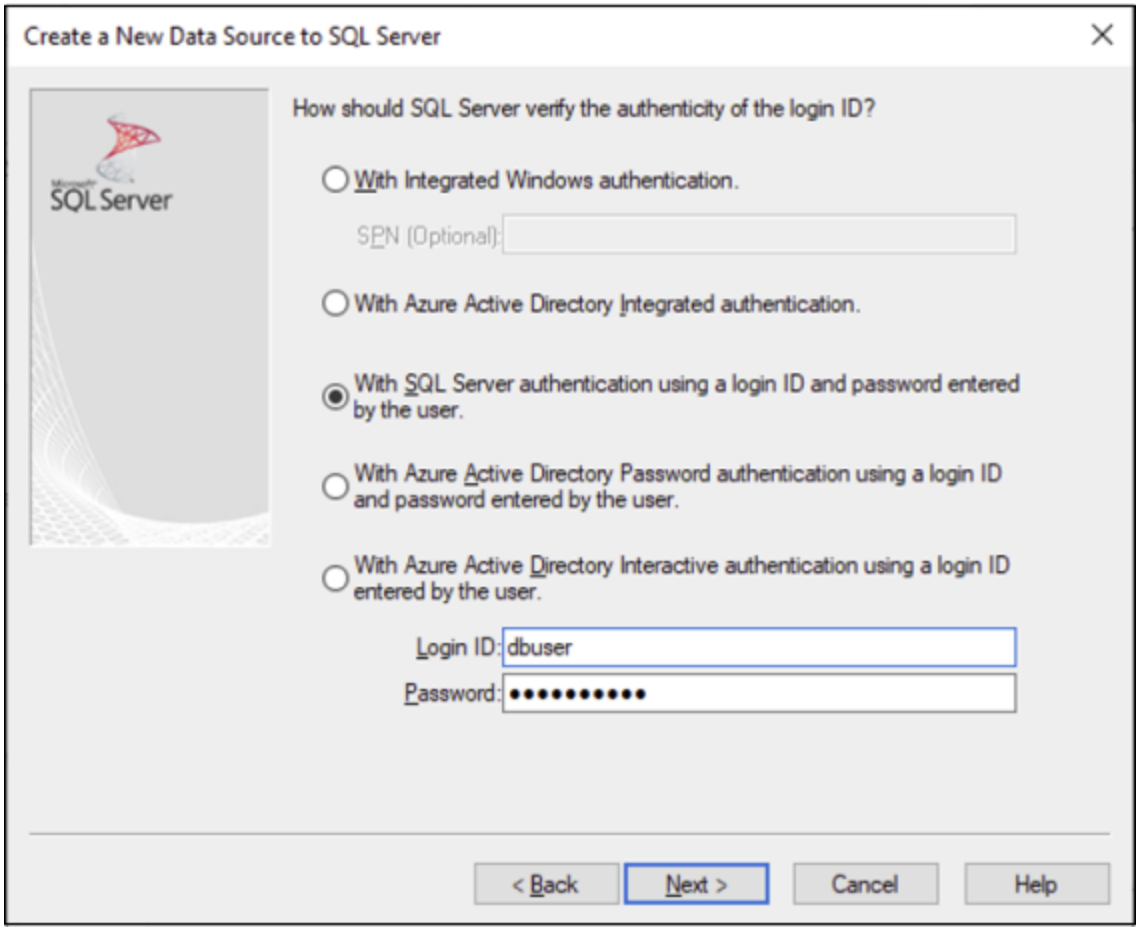

5. Cliquez sur **DSN Utilisateur** (ou **System DSN** si vous créez un DNS Système).

> La création d'un **DNS Système** exige que MapForce soit exécuté en tant qu'administrateur. C'est pourquoi, afin de créer un **DNS Système**, annulez l'assistant, assurez-vous d'exécuter MapForce en tant qu'administrateur et effectuez les étapes ci-dessus à nouveau.

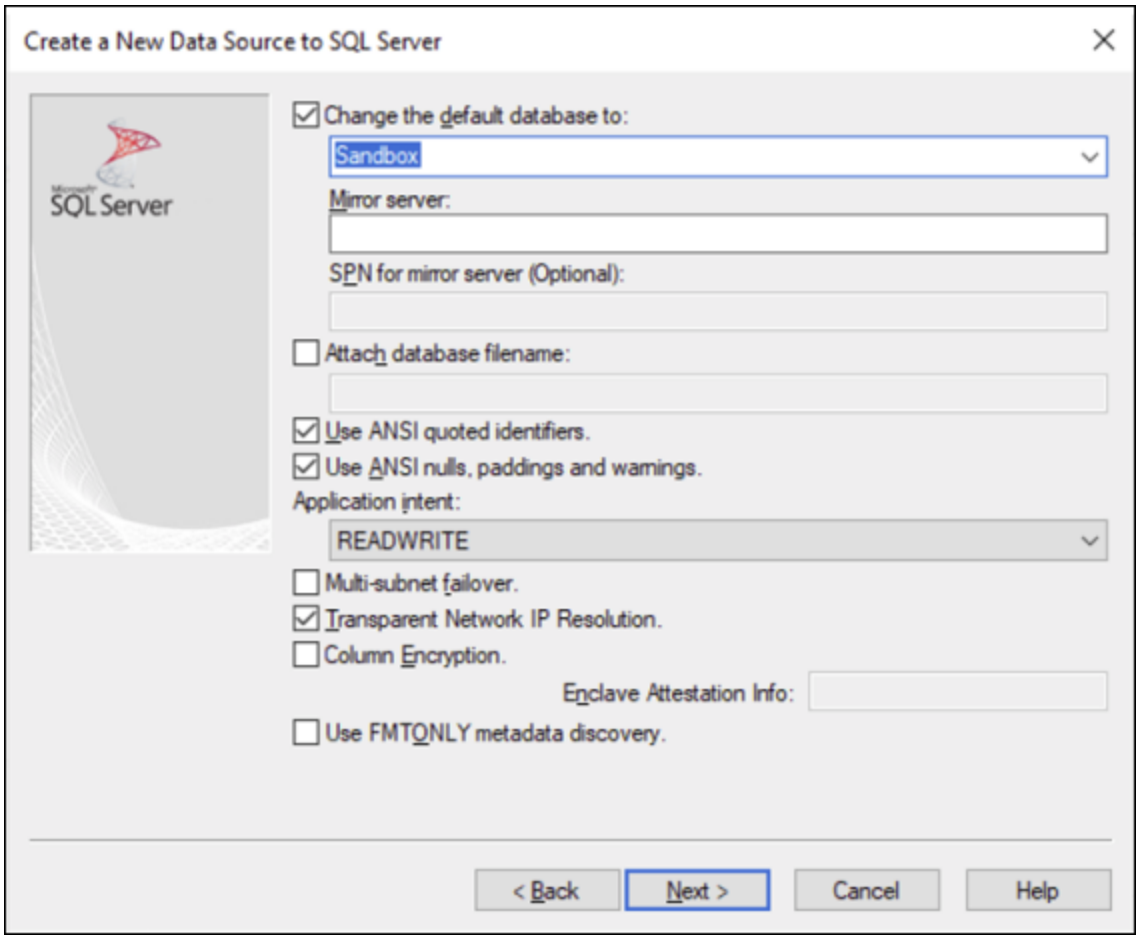
6. Saisissez un nom et, en option, une description pour identifier cette connexion, puis sélectionner le serveur SQL depuis la liste sur lequel vous voulez vous connecter (**SQLSERV01** dans cet exemple).



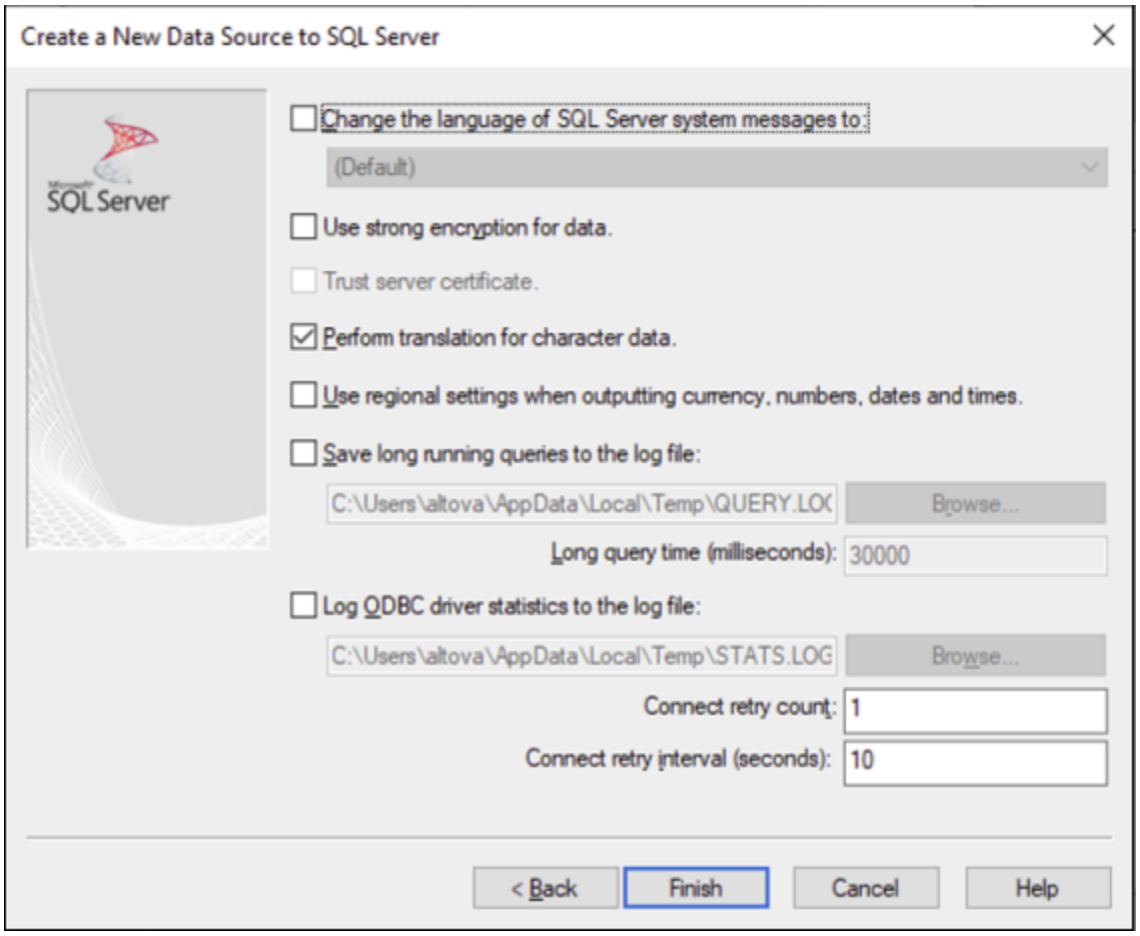

7. Si le serveur de la base de données a été configuré pour permettre des connexions depuis les utilisateurs authentifiés sur le domaine Windows, sélectionnez **Avec l'authentification Windows intégrée**. Sinon, sélectionnez une des autres options, le cas échéant. Cet exemple utilise **Avec authentification SQL Server...** , qui requiert que le nom utilisateur et le mot de passe soient saisis dans les cases pertinentes.



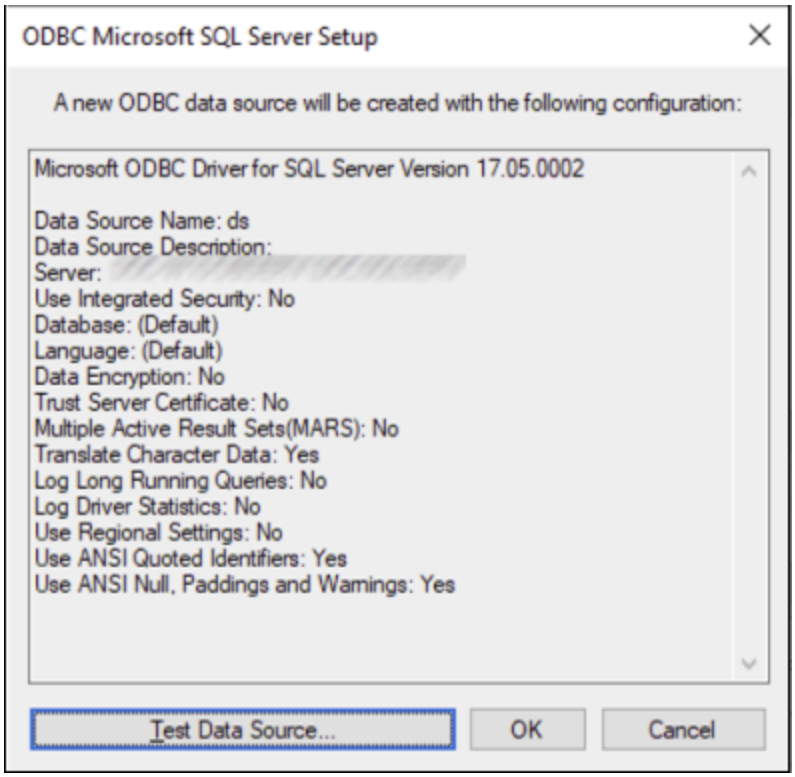

8. En option, cochez la case **Changer la base de données par défaut en** et saisissez le nom de la base de données à laquelle vous vous connectez (dans cet exemple, **Sandbox**).



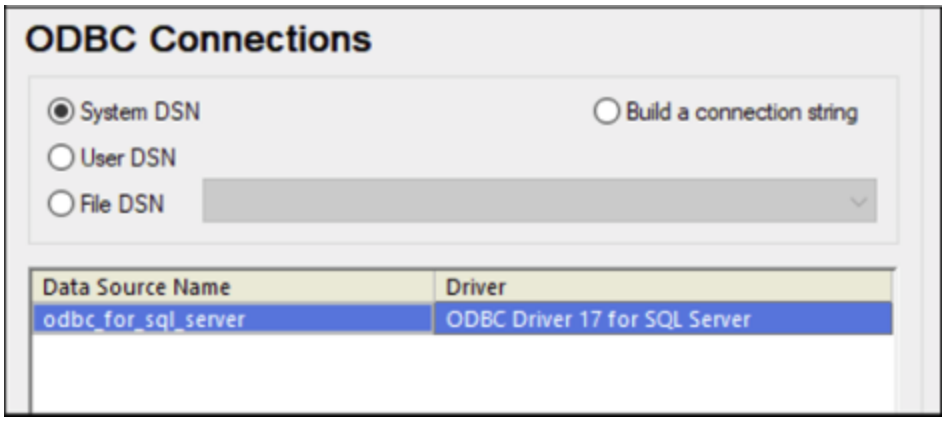

9. Cliquez sur **Suivant** et, en option, configurez des paramètres supplémentaires pour cette connexion.




10. Cliquez sur **Terminer**. Un dialogue de confirmation recensant les détails de connexion s'ouvre.




11. Cliquez sur **OK**. La source de données apparaît maintenant dans la liste de sources de données **Utilisateur** ou **Système**, tel que configuré, par exemple :




### 4.2.1.10.13 MySQL (ODBC)

Cet exemple illustre comment se connecter à un serveur de base de données MySQL depuis une machine Windows par le biais du pilote ODBC. Le pilote ODBC MySQL n'est pas disponible sur Windows, il doit donc être téléchargé et installé séparément. Cet exemple utilise MySQL Connecteur/ODBC 8.0.

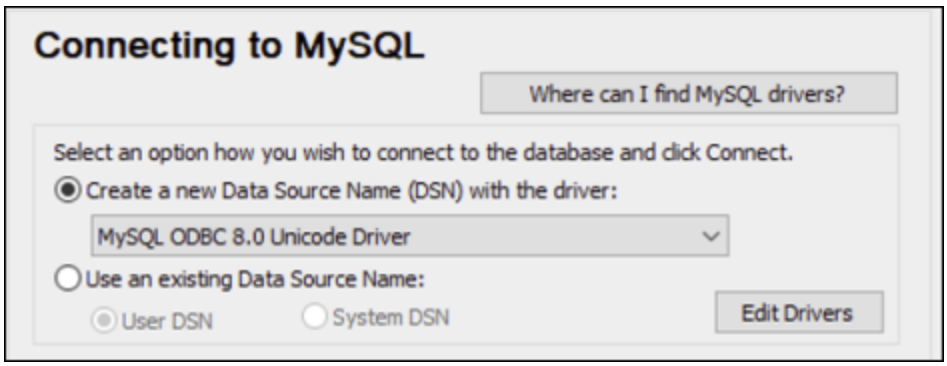
**Exigences préalables :**

- Le pilote ODBC MySQL doit être installé sur votre système d'exploitation. Vérifiez la documentation MySQL pour la version du pilote recommandée de votre version de serveur de base de données (voir https://dev.mysql.com/downloads/connector/odbc/).
- Vous disposez des détails de connexion à la base de données suivants : hôte, base de données, port, nom utilisateur et mot de passe.

Si vous avez installé MySQL Connector/ODBC pour la plateforme 64-bit, assurez-vous d'installer aussi MapForce pour la plateforme 64-bit.

**Pour se connecter à MySQL via ODBC :**

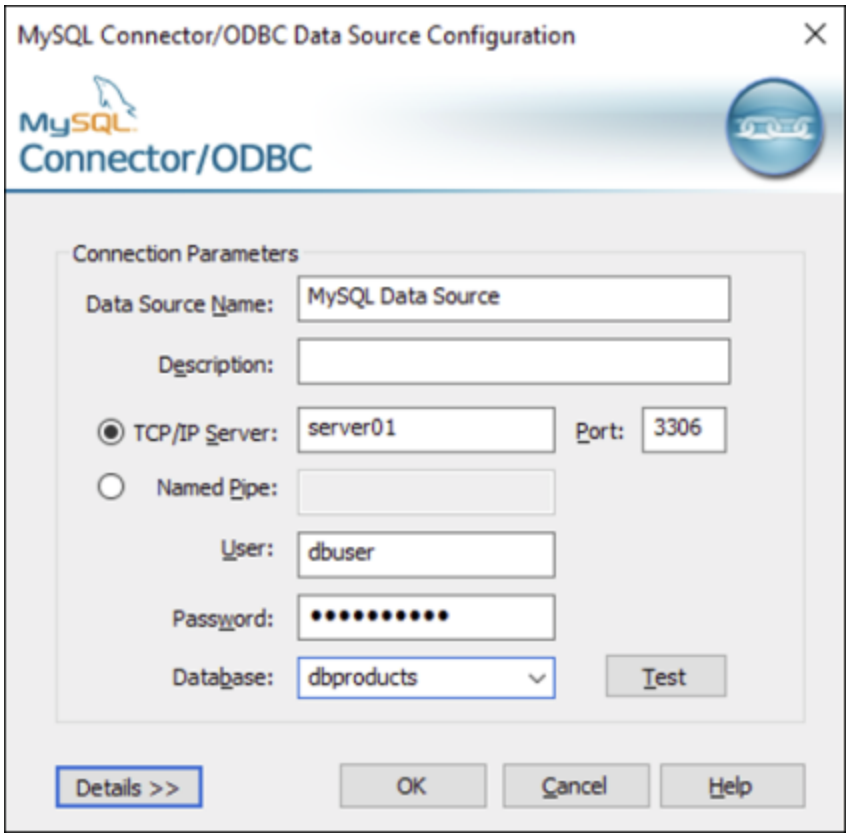
1. Lancer l'assistant de connexion de base de données 163.
2. Sélectionnez **MySQL (ODBC)**, puis cliquez sur **Suivant**.




3. Sélectionnez **Créer un nouveau Nom de source de données (DSN) avec le pilote**, puis sélectionnez un pilote MySQL. Si aucun pilote MySQL n'est disponible dans la liste, cliquez sur **Éditer les pilotes**, puis sélectionnez tout pilote MySQL disponible (la liste contient tous les pilotes ODBC installés sur votre système d'exploitation).

   Si vous avez installé MapForce 64-bit, alors les pilotes ODBC 64-bit sont affichés dans la liste. Sinon, les pilotes ODBC 32-bit sont affichés. Voir aussi Consulter les pilotes ODBC disponibles 184.

4. Cliquez sur **Connexion**.




5. Dans le champ du nom de la source de données, saisissez un nom descriptif qui vous permettra d'identifier cette source de données ODBC à l'avenir.
6. Remplir les identifiants de connexion de base de données (Serveur TCP/IP, Utilisateur, Mot de passe), sélectionnez une base de données et cliquez sur **OK**.

**Note :** si le serveur de base de données est à distance, il doit être configuré par l'administrateur du serveur pour accepter les connexions à distance depuis l'adresse IP de votre appareil. De même, si vous cliquez sur **Détails>>**, vous trouverez plusieurs paramètres supplémentaires disponibles pour la configuration. Vérifiez la documentation du pilote avant de changer leurs valeurs par défaut.

### 4.2.1.10.14 Oracle (JDBC)

Cet exemple montre comment vous connecter à un serveur de base de données Oracle à partir d'une machine client, à l'aide de l'interface JDBC. La connexion est créée en tant qu'une connexion Java pure, par le biais de **Oracle Instant Client Package (Basic)** disponible sur le site Internet Oracle. L'avantage de ce type de connexion est qu'elle ne nécessite que l'environnement Java et les bibliothèques .jar fournies par l'Oracle Instant Client Package, vous évitant d'installer et de configurer un client de base de données plus complexe.

**Exigences préalables :**

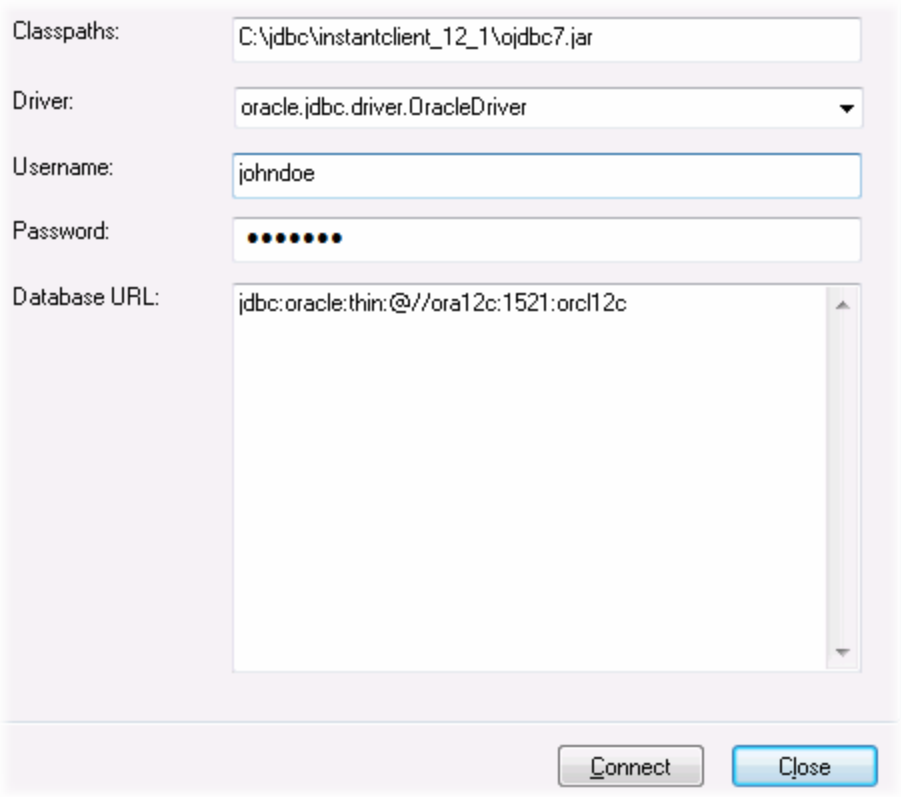
- JRE (Java Runtime Environment) ou Java Development Kit (JDK) doit être installé. Il peut s'agir soit d'Oracle JDK soit d'un build open source comme Oracle OpenJDK. MapForce déterminera le chemin vers la Java Virtual Machine (JVM) depuis les emplacements suivants, dans l'ordre suivant : a) le

chemin personnalisé JVM que vous avez éventuellement défini dans l'application **Options**, voir Paramètres Java[1091]; b) le chemin JVM trouvé dans le registre Windows ; c) la variable d'environnement `JAVA_HOME`.

- Veuillez vous assurer que la plateforme de MapForce (32-bit, 64-bit) correspond à celle du JRE/JDK.
- L'**Oracle Instant Client Package (Basic)** doit être disponible sur votre système d'exploitation. Le paquet peut être téléchargé depuis le site Internet officiel d'Oracle. Cet exemple utilise l'Oracle Instant Client Package version 12.1.0.2.0, pour Windows 32-bit et par conséquent, Oracle JDK 32-bit.
- Vous disposez des détails de connexion à la base de données suivants : hôte, port, nom de service, nom d'utilisateur et mot de passe.

**Pour se connecter à Oracle par le biais de Instant Client Package :**

1. Lancer l'assistant de connexion de base de données[163].
2. Cliquez sur **JDBC Connections**.
3. À côté de « Classpaths », saisissez le chemin menant au fichier .jar qui fournit une connectivité à la base de données. Si nécessaire, vous pouvez aussi saisir une liste séparée par points-virgules de chemins d'accès de fichiers .jar. Dans cet exemple, le fichier .jar exigé se trouve sous le chemin suivant : **C:\jdbc\instantclient_12_1\ojdbc7.jar**. Veuillez noter que vous pouvez laisser en blanc le champ « Chemins de classe » si vous avez ajouté le(s) chemin(s) d'accès du fichier .jar à la variable d'environnement CLASSPATH du système d'exploitation (voir aussi Configuration du CLASSPATH[188]).
4. Dans le champ « Pilote », sélectionnez soit **oracle.jdbc.OracleDriver**, soit **oracle.jdbc.driver.OracleDriver**. Veuillez noter que ces saisies sont disponibles si un chemin d'accès .jar valid est trouvé soit dans le champ de saisie « Classpath », soit dans la variable d'environnement CLASSPATH du système d'exploitation (voir étape précédente).
5. Saisissez le nom d'utilisateur et le mot de passe de la base de données dans les champs de saisie correspondants.




6. Saisissez la chaîne de connexion au serveur de la base de données dans la zone de texte URL de base de données, en remplaçant les valeurs soulignées avec celles applicables à votre serveur de base de données.

```
jdbc:oracle:thin:@//host:port:service
```

7. Cliquez sur **Connexion**.

### 4.2.1.10.15 Oracle (ODBC)

Cet exemple illustre un scénario commun au cours duquel vous vous connectez depuis MapForce à un serveur de base de données Oracle sur une machine de réseau par le biais d'un client de base de données Oracle installé sur le système d'exploitation local.

L'exemple comprend des instructions pour configurer une source de données ODBC (DSN) en utilisant l'assistant de connexion de base de données dans MapForce. Si vous avez déjà créé un DSN, ou si vous préférez le créer directement depuis l'**administrateur de ODBC Data Source** dans Windows, vous pouvez le faire, puis le sélectionner si vous êtes invité à le faire par l'assistant. Pour plus d'informations concernant les sources de données ODBC, voir Configurer une connexion ODBC 181.

**Exigences préalables :**

- Le client de base de données Oracle (qui inclut le pilote ODBC Oracle) doit être installé et configuré sur votre système d'exploitation. Pour obtenir des instructions d'installation et de configuration d'un client de base de données Oracle, veuillez vous référer à la documentation fournie avec votre logiciel Oracle.
- Le fichier **tnsnames.ora** situé dans le répertoire personnel Oracle contient une entrée qui décrit les paramètres de connexion de la base de données dans un format semblable au suivant :

```
ORCL =
  (DESCRIPTION =
    (ADDRESS_LIST =
      (ADDRESS = (PROTOCOL = TCP)(HOST = server01)(PORT = 1521))
    )
    (CONNECT_DATA =
      (SID = orcl)
      (SERVER = DEDICATED)
    )
  )
```

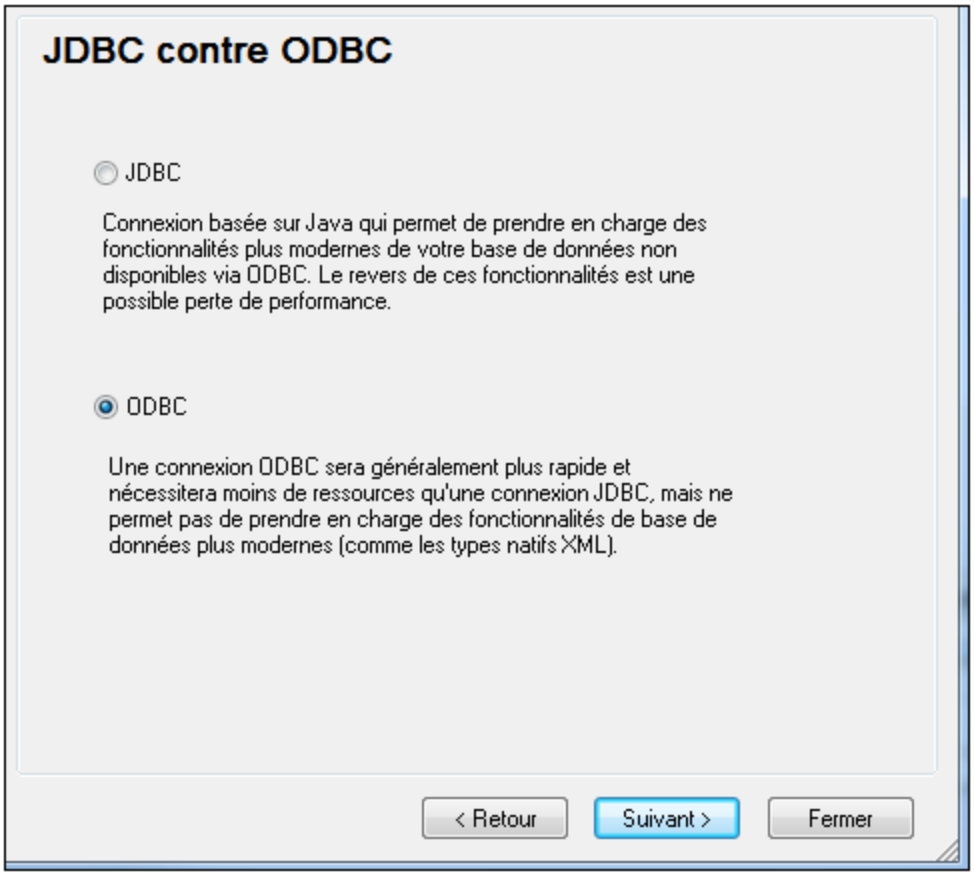
Le chemin d'accès menant au fichier **tnsnames.ora** dépend de l'emplacement où le répertoire personnel Oracle a été installé. Pour le client de base de données Oracle 11.2.0, le chemin d'accès du répertoire personnel Oracle par défaut pourrait être comme suit :

```
C:\app\username\product\11.2.0\client_1\network\admin\tnsnames.ora
```

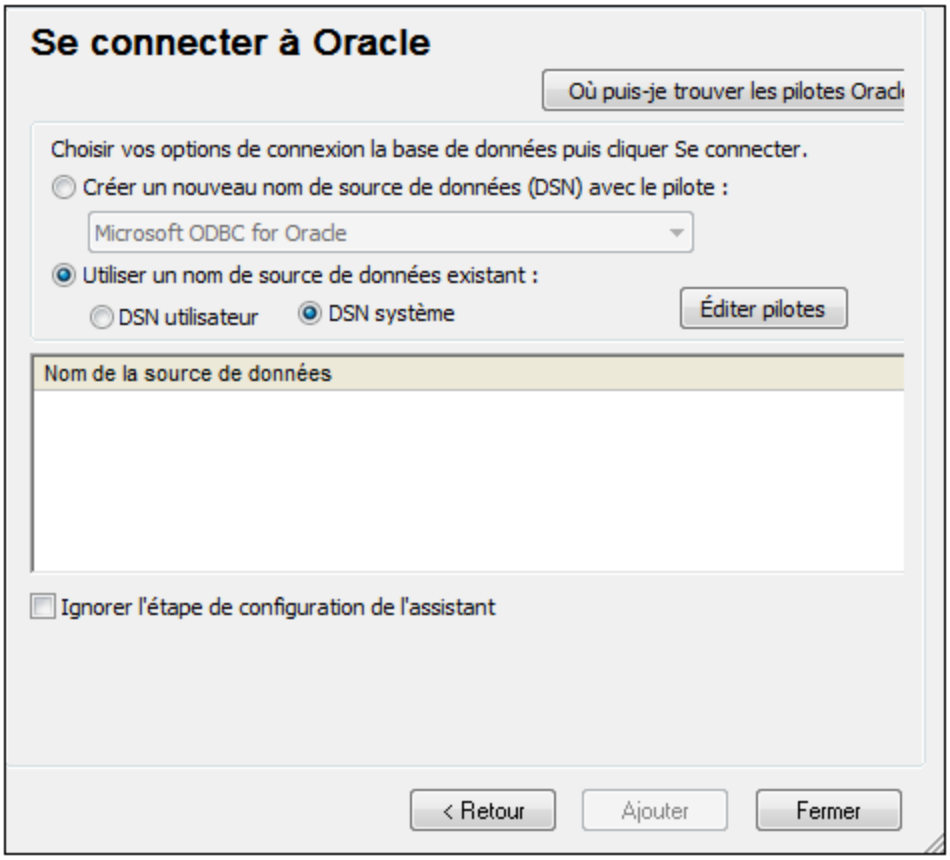
Vous pouvez ajouter de nouvelles entrées au fichier **tnsnames.ora** soit en collant les détails de connexion et en enregistrant le fichier, ou en exécutant l'assistant d'Oracle *Net Configuration Assistant* (si disponible). Si vous voulez que ces valeurs apparaissent dans des listes déroulantes durant le processus de configuration, alors vous allez éventuellement devoir ajouter le chemin au dossier admin en tant que variable d'environnement**TNS_ADMIN**.

**Pour se connecter à Oracle avec ODBC :**

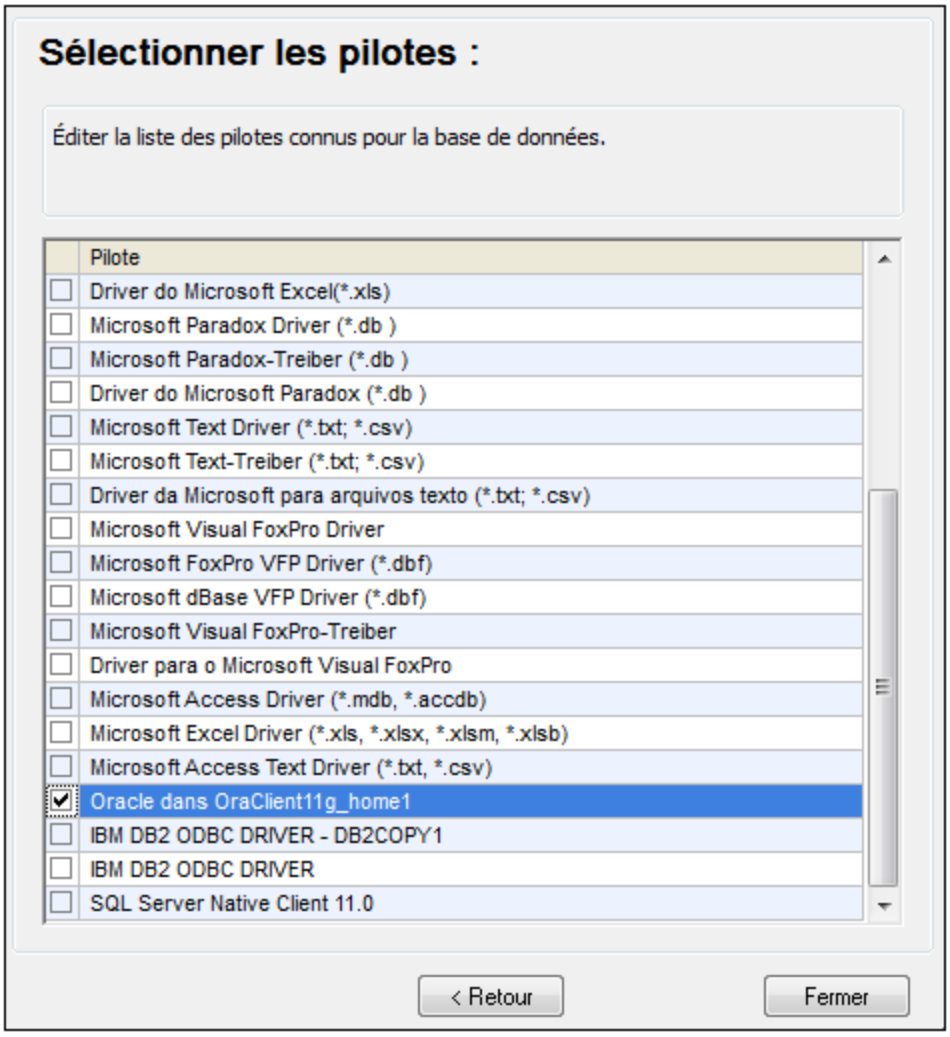
1. Lancer l'assistant de connexion de base de données 163.
2. Sélectionnez **Oracle (ODBC / JDBC)**, puis cliquez sur **Suivant**.



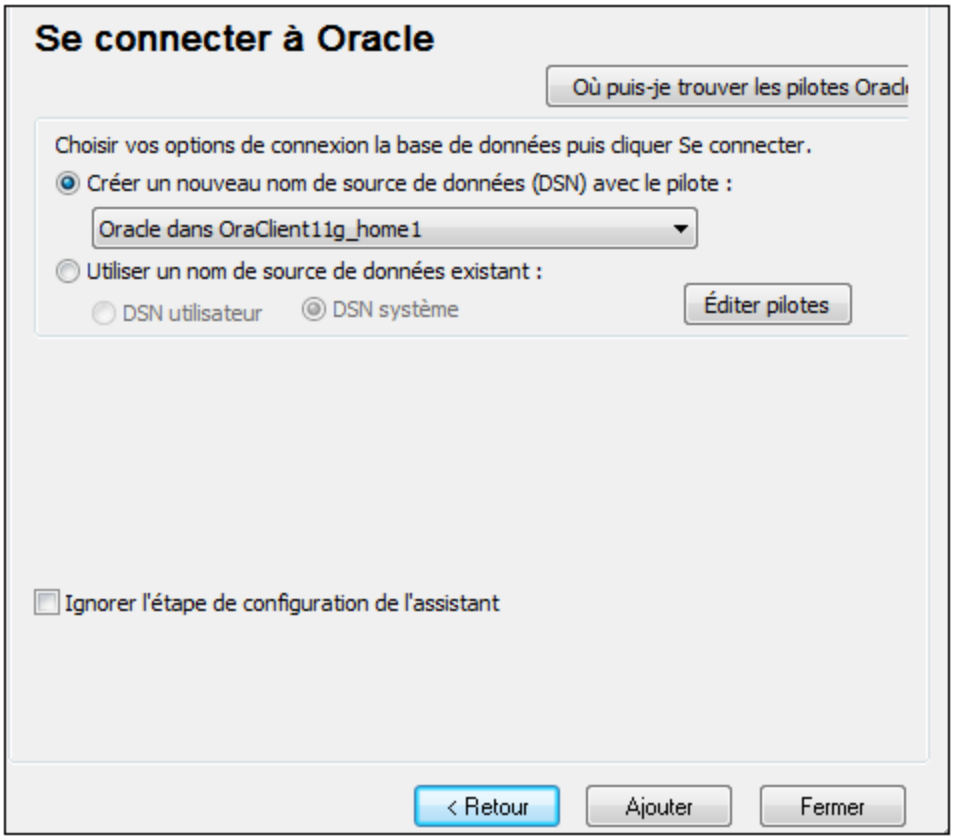

3. Sélectionnez **ODBC**.




4. Cliquez sur **Éditer les pilotes**.

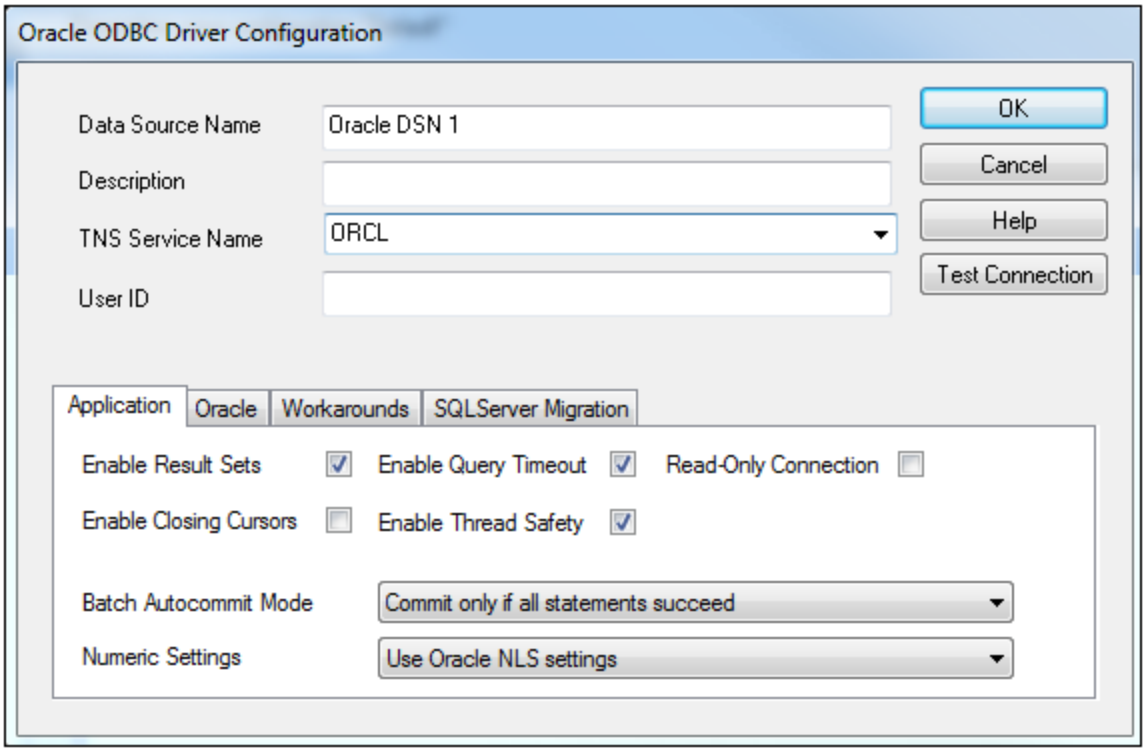



5. Sélectionnez les pilotes Oracle que vous souhaitez utiliser (dans cet exemple, **Oracle dans OraClient11g_home1**). La liste affiche les pilotes disponibles sur votre système après l'installation du client Oracle.
6. Cliquez sur **Retour**.
7. Sélectionner **Créer un nouveau nom de source de données (DSN) avec le pilote**, puis sélectionner le pilote Oracle choisi dans l'étape .4

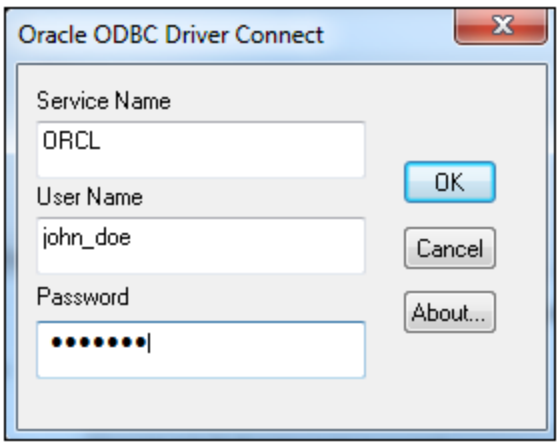



Évitez d'utiliser le pilote fourni par Microsoft appelé **Microsoft ODBC for Oracle**. Microsoft recommande d'utiliser le pilote ODBC fourni par Oracle (voir http://msdn.microsoft.com/en-us/library/ms714756%28v=vs.85%29.aspx)

8. Cliquez sur **Se connecter**.




9. Dans le champ de saisie Nom de source de données, saisissez un nom pour identifier la source de données (dans cet exemple, **Oracle DSN 1**).
10. Dans le champ TNS Service Name, saisissez le nom de connexion tel qu'il est défini dans le fichier **tnsnames.ora** (voir conditions préalables[230]). Dans cet exemple, le nom de connexion est **ORCL**.
*Note :* Si vous souhaitez avoir une liste déroulante de la combo box peuplée avec les valeurs du fichier **tnsnames.ora**, alors vous allez éventuellement devoir ajouter le chemin au dossier admin en tant que variable d'environnement **TNS_ADMIN**.
11. Cliquez sur **OK**.




12. Saisissez le nom d'utilisateur et le mot de passe de la base de données, puis cliquez sur OK.

### 4.2.1.10.16 PostgreSQL (ODBC)

Cet exemple illustre comment se connecter à un serveur de base de données PostgreSQL depuis un appareil Windows machine par le biais des pilotes ODBC. Le pilote ODBC PostgreSQL n'est pas disponible sur Windows, il doit donc être téléchargé et installé séparément. Cet exemple utilise le pilote psqlODBC (version 11.0) téléchargé depuis le site web officiel (voir aussi Aperçu des pilotes de base de données 165).

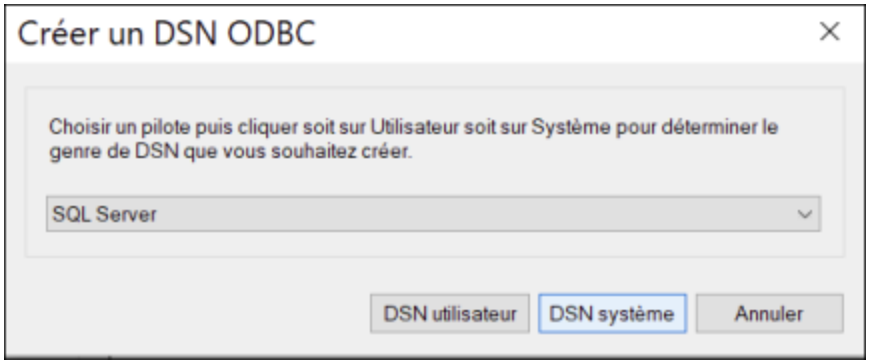
**Note :** vous pouvez aussi vous connecter à un serveur de base de données PostgreSQL directement (sans le pilote ODBC), voir Configurer une connexion PostgreSQL 191.

**Exigences préalables :**

- Le pilote *psqlODBC* doit être installé sur votre système d'exploitation.
- Vous disposez des détails de connexion à la base de données suivants : serveur, port, base de données, nom d'utilisateur et mot de passe.

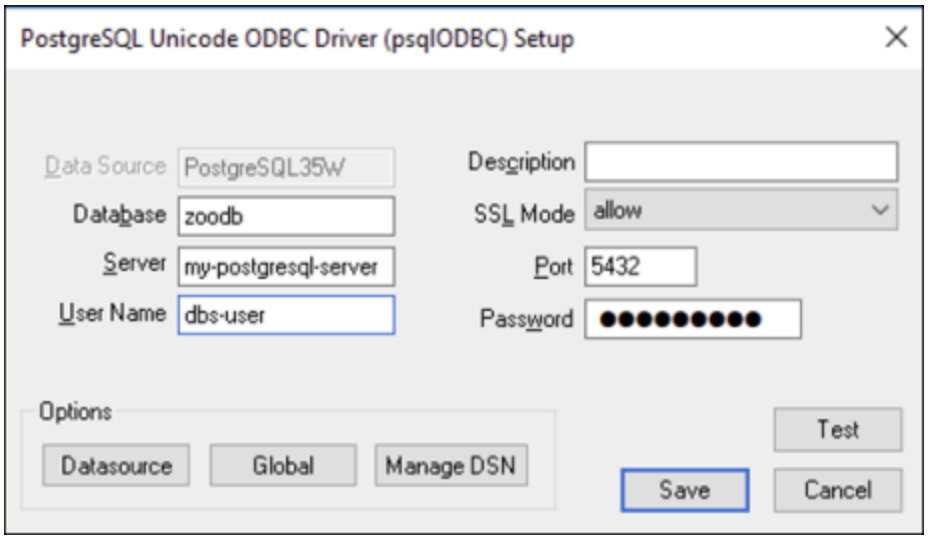
**Pour établir une connexion à PostgreSQL avec ODBC :**

1. Lancer l'assistant de connexion de base de données 163.
2. Cliquez sur **Connexions ODBC**.
3. Sélectionnez l'option **DSN Utilisateur**.
4. Cliquez sur **Créer un nouveau DSN** et sélectionnez le pilote depuis la liste déroulante. Si aucun pilote PostgreSQL n'est disponible dans la liste, veuillez vous assurer que le pilote ODBC PostgreSQL est installé sur votre système d'exploitation, comme mentionné dans les conditions préalables ci-dessus.



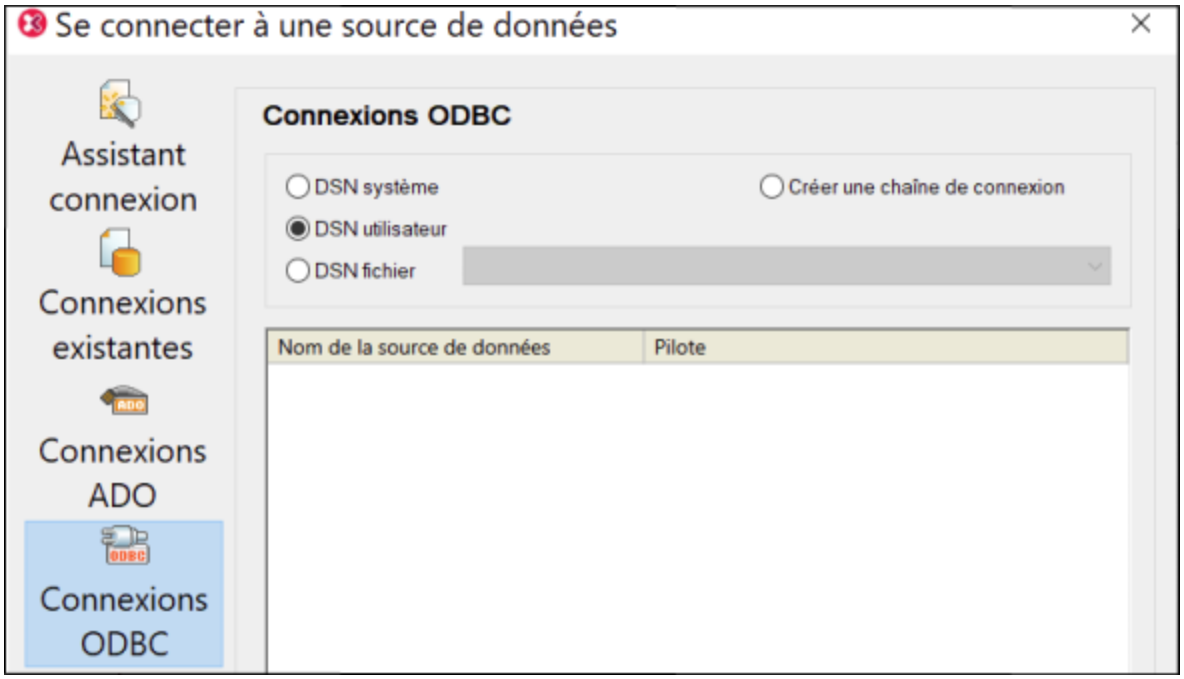

5. Cliquez sur **DSN Utilisateur**.




6. Remplissez les identifiants de connexion à la base de données (ils doivent être fournis par le propriétaire de la base de données), puis cliquez sur **Enregistrer**.

La connexion est désormais disponible dans la liste des connexions ODBC. Pour se connecter à la base de données, vous pouvez soit double-cliquer sur la connexion ou la sélectionner, puis cliquer sur **Connexion**.




### 4.2.1.10.17 Progress OpenEdge (JDBC)

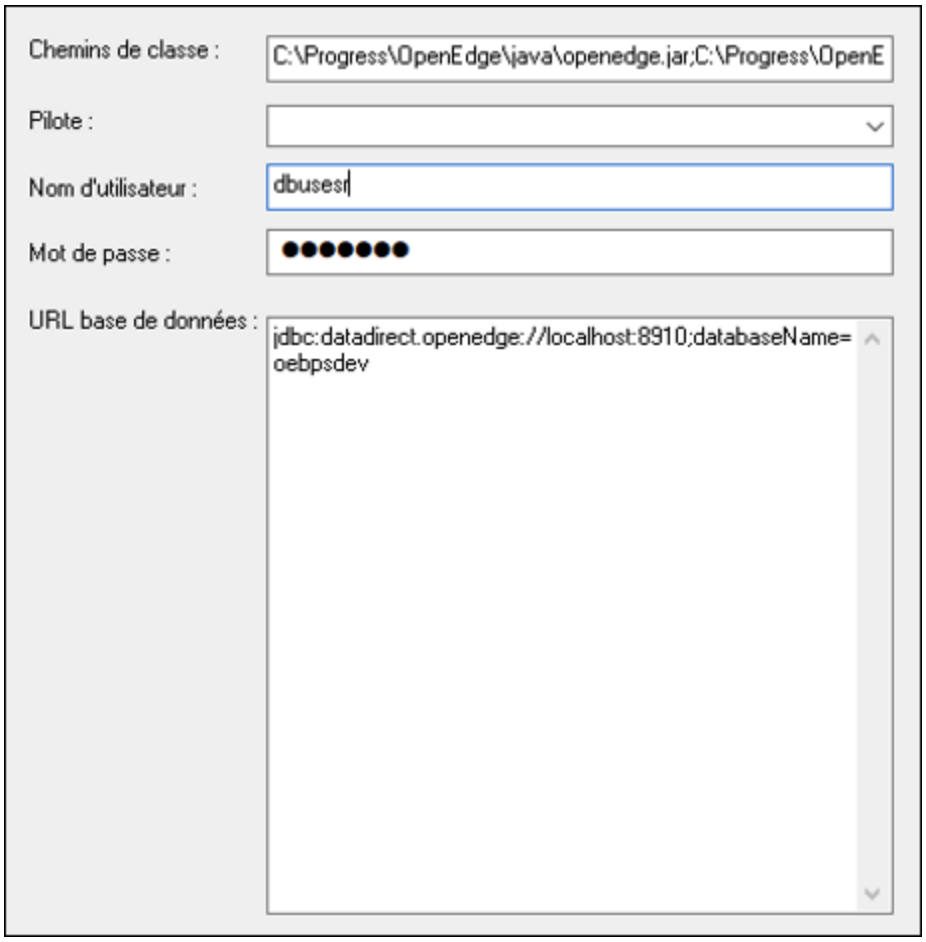
Cet exemple illustre comment se connecter à un serveur de base de données Progress OpenEdge 11.6 par le biais de JDBC.

### Prérequis

- JRE (Java Runtime Environment) ou Java Development Kit (JDK) doit être installé. Il peut s'agir soit d'Oracle JDK, soit d'un build open source comme Oracle OpenJDK. MapForce déterminera le chemin vers la Java Virtual Machine (JVM) depuis les emplacements suivants, dans l'ordre suivant : a) le chemin personnalisé JVM que vous avez éventuellement défini dans l'application **Options**, voir Paramètres Java 1091; b) le chemin JVM trouvé dans le registre Windows ; c) la variable d'environnement `JAVA_HOME`.
- Veuillez vous assurer que la plateforme de MapForce (32-bit, 64-bit) correspond à celle du JRE/JDK.
- La variable d'environnement `PATH` du système d'exploitation doit inclure le chemin vers le répertoire `bin` du répertoire d'installation JRE ou JDK, par exemple `C:\Program Files (x86)\Java\jre1.8.0_51\bin`.
- Le pilote Progress OpenEdge JDBC doit être disponible sur votre système d'exploitation. Dans cet exemple, la connectivité JDBC est fournie par les fichiers de composant de pilote **openedge.jar** et **pool.jar** disponibles dans **C:\Progress\OpenEdge\java** en tant que partie intégrante de l'installation OpenEdge SDK.
- Vous disposez des détails de connexion à la base de données suivants : hôte, port, nom de la base de données, nom d'utilisateur et mot de passe.

### Pour se connecter à OpenEdge par le biais de JDBC

1. Lancer l'assistant de connexion de base de données 163.
2. Cliquez sur **JDBC Connections**.
3. À côté de « Classpaths », saisissez le chemin menant au fichier .jar qui fournit une connectivité à la base de données. Si nécessaire, vous pouvez aussi saisir une liste séparée par points-virgules de chemins d'accès de fichiers .jar. Dans cet exemple, les chemins de fichier .jar exigés se trouvent sous : **`C:\Progress\OpenEdge\java\openedge.jar;C:\Progress\OpenEdge\java\pool.jar;`**. Veuillez noter que vous pouvez laisser en blanc le champ « Chemins de classe » si vous avez ajouté le chemin d'accès du fichier .jar à la variable d'environnement CLASSPATH du système d'exploitation (voir aussi Configuration du CLASSPATH 188).
4. Dans le champ « Driver », choisir **com.ddtek.jdbc.openedge.OpenEdgeDriver**. Veuillez noter que cette entrée est disponible si un chemin d'accès de fichier .jar valide est trouvé soit dans le champ « Chemins de classe », soit dans la variable d'environnement CLASSPATH du système d'exploitation (voir l'étape précédente).




5. Saisissez le nom d'utilisateur et le mot de passe de la base de données dans les champs de saisie correspondants.
6. Saisissez la chaîne de connexion au serveur de la base de données dans la zone de texte URL de base de données, en remplaçant les valeurs soulignées avec celles applicables à votre serveur de base de données.

```
jdbc:datadirect:openedge://host:port;databaseName=db_name
```

7. Cliquez sur **Connexion**.

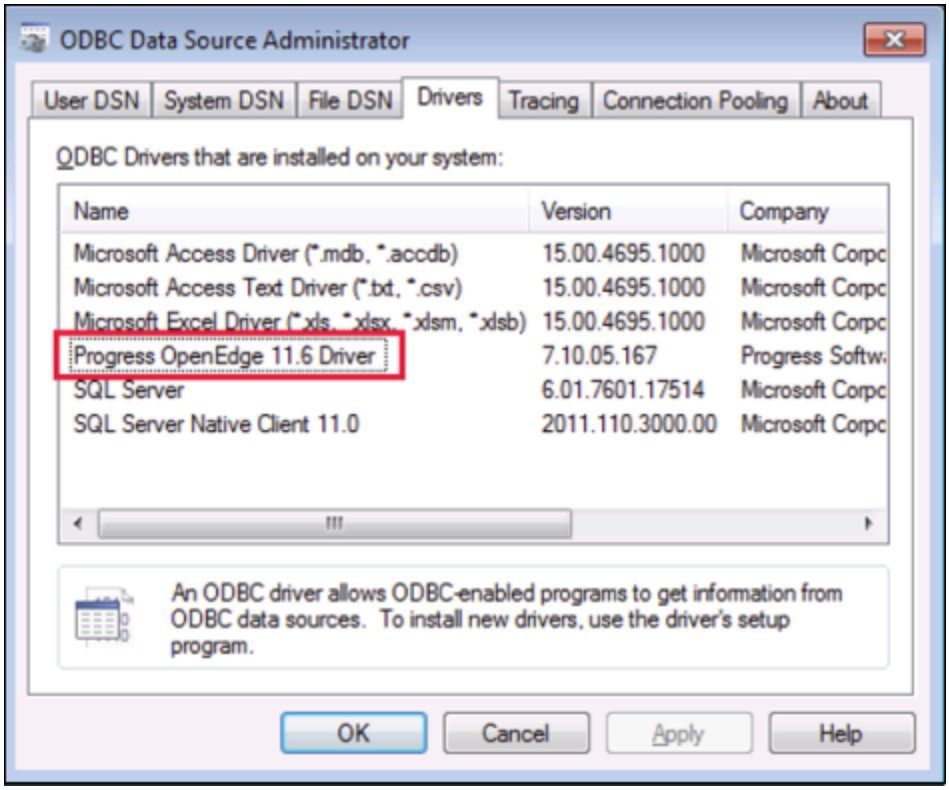
### 4.2.1.10.18 Progress OpenEdge (ODBC)

Cet exemple illustre comment se connecter à un serveur de base de données Progress OpenEdge par le biais du pilote Progress OpenEdge 11.6 ODBC.

**Exigences préalables :**

- Le pilote *ODBC Connector for Progress OpenEdge* doit être installé sur votre système d'exploitation. Le pilote Progress OpenEdge ODBC peut être téléchargé depuis le site web du vendeur (voir aussi

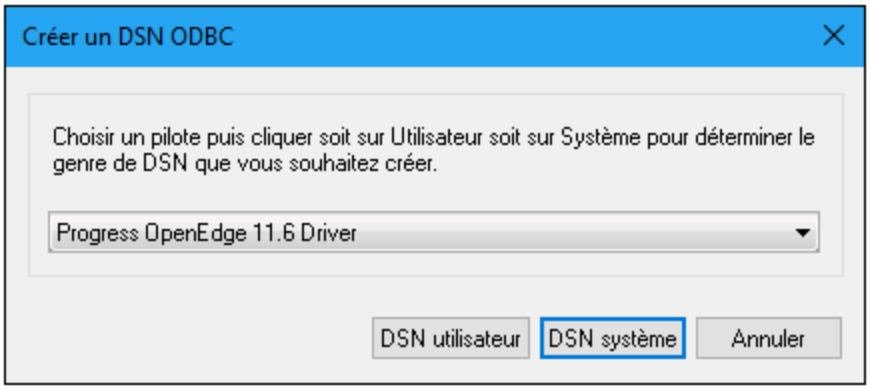
Aperçu des pilotes de base de données[165]). Veillez à télécharger le pilote 32-bit lorsque vous exécutez la version 32-bit de MapForce, et le pilote 64-bit lorsque vous exécutez la version 64-bit. Après l'installation, vérifiez si le pilote ODBC est disponible sur votre appareil (voir aussi Aperçu des pilotes ODBC disponibles[184]).




- Vous disposez des détails de connexion de base de données suivants : nom d'hôte, numéro de port, nom de base de données, ID utilisateur et mot de passe.

### Pour se connecter à Progress OpenEdge par le biais d'ODBC

1. Lancer l'assistant de connexion de base de données[163].
2. Cliquez sur **Connexions ODBC**.
3. Cliquez sur**DSN Utilisateur** (en alternative, cliquez sur **DSN Système** ou **DSN Fichier**, dans un tel cas, les instructions suivantes seront semblables).
4. Cliquez sur **Ajouter** .
5. Sélectionnez le **pilote Progress OpenEdge** depuis la liste et cliquez sur **DSN Utilisateur** (ou **DSN Système**, le cas échéant).




6. Remplissez les identifiants de connexion à la base de données (base de données, serveur, port, nom utilisateur, mot de passe) et cliquez sur **OK**. Pour vérifier la connectivité avant d'enregistrer les données saisies, cliquez sur **Tester la connexion**.


7. Cliquer sur OK. La nouvelle source de données apparait désormais dans la liste des sources de données d'ODBC.


8. Cliquez sur **Se connecter**.

### 4.2.1.10.19 Sybase (JDBC)

Cet exemple illustre comment se connecter à un serveur de base de données Sybase par le biais de JDBC.

**Exigences préalables :**

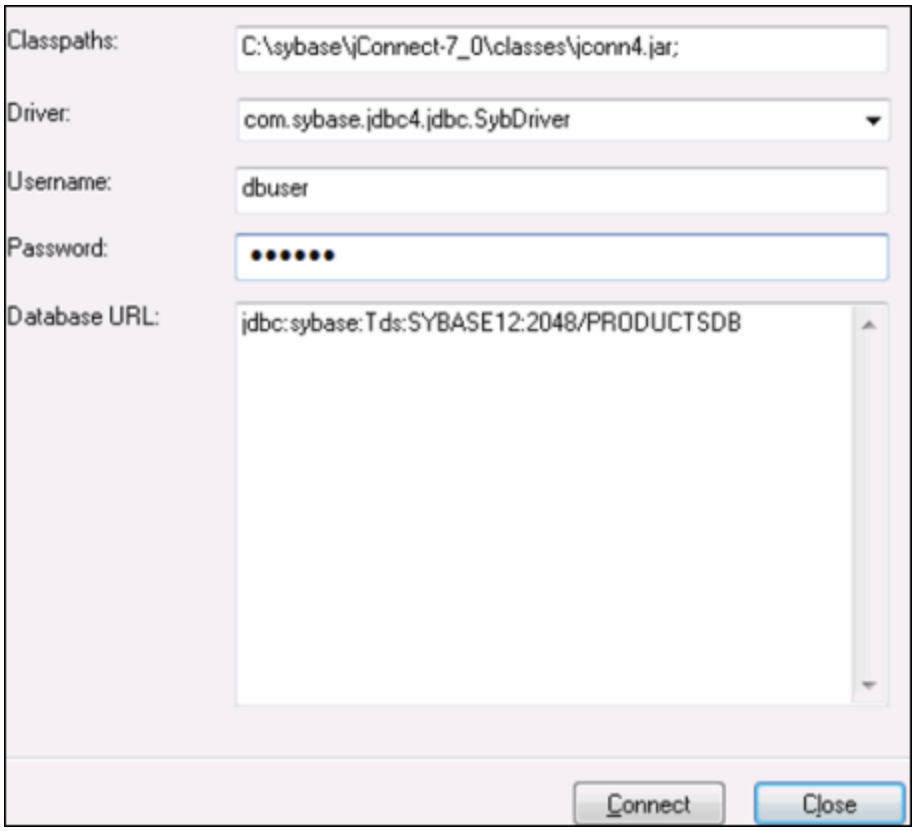
- JRE (Java Runtime Environment) ou Java Development Kit (JDK) doit être installé. Il peut s'agir soit d'Oracle JDK, soit d'un build open source comme Oracle OpenJDK. MapForce déterminera le chemin vers la Java Virtual Machine (JVM) depuis les emplacements suivants, dans l'ordre suivant : a) le chemin personnalisé JVM que vous avez éventuellement défini dans l'application **Options**, voir Paramètres Java[1091]; b) le chemin JVM trouvé dans le registre Windows ; c) la variable d'environnement `JAVA_HOME`.
- Assurez-vous que la plateforme de MapForce (32-bit, 64-bit) correspond à celle du JRE/JDK.
- Le composant Sybase *jConnect* doit être installé sur votre système d'exploitation (dans cet exemple, *jConnect 7.0* est utilisé, installé en tant que partie intégrante de l'installation *Sybase Adaptive Server Enterprise PC Client*). En ce qui concerne les instructions d'installation du client de base de données, référez-vous à la documentation Sybase.

- Vous disposez des détails de connexion à la base de données suivants : hôte, port, nom de la base de données, nom d'utilisateur et mot de passe.

### Connexion à Sybase par le biais de JDBC :

1. Lancer l'assistant de connexion de base de données 163.
2. Cliquez sur **JDBC Connections**.
3. À côté de « Classpaths », saisissez le chemin menant au fichier .jar qui fournit une connectivité à la base de données. Si nécessaire, vous pouvez aussi saisir une liste séparée par points-virgules de chemins d'accès de fichiers .jar. Dans cet exemple, le chemin de fichier .jar exigé se trouve sous : **C:\sybase\jConnect-7_0\classes\jconn4.jar**. Veuillez noter que vous pouvez laisser en blanc le champ « Chemins de classe » si vous avez ajouté le(s) chemin(s) d'accès du fichier .jar à la variable d'environnement CLASSPATH du système d'exploitation (voir aussi Configuration du CLASSPATH 188 ).
4. Dans la case "Driver", choisir **com.sybase.jdbc4.jdbc.SybDriver**. Veuillez noter que cette entrée est disponible si un chemin d'accès de fichier .jar valide est trouvé soit dans le champ « Chemins de classe », soit dans la variable d'environnement CLASSPATH du système d'exploitation (voir l'étape précédente).




5. Saisissez le nom d'utilisateur et le mot de passe de la base de données dans les champs de saisie correspondants.
6. Saisissez la chaîne de connexion au serveur de la base de données dans la zone de texte URL de base de données, en remplaçant les valeurs soulignées avec celles applicables à votre serveur de base de données.

```
jdbc:sybase:Tds:hostName:port/databaseName
```

7. Cliquez sur **Connexion**.

### 4.2.1.10.20 Teradata (JDBC)

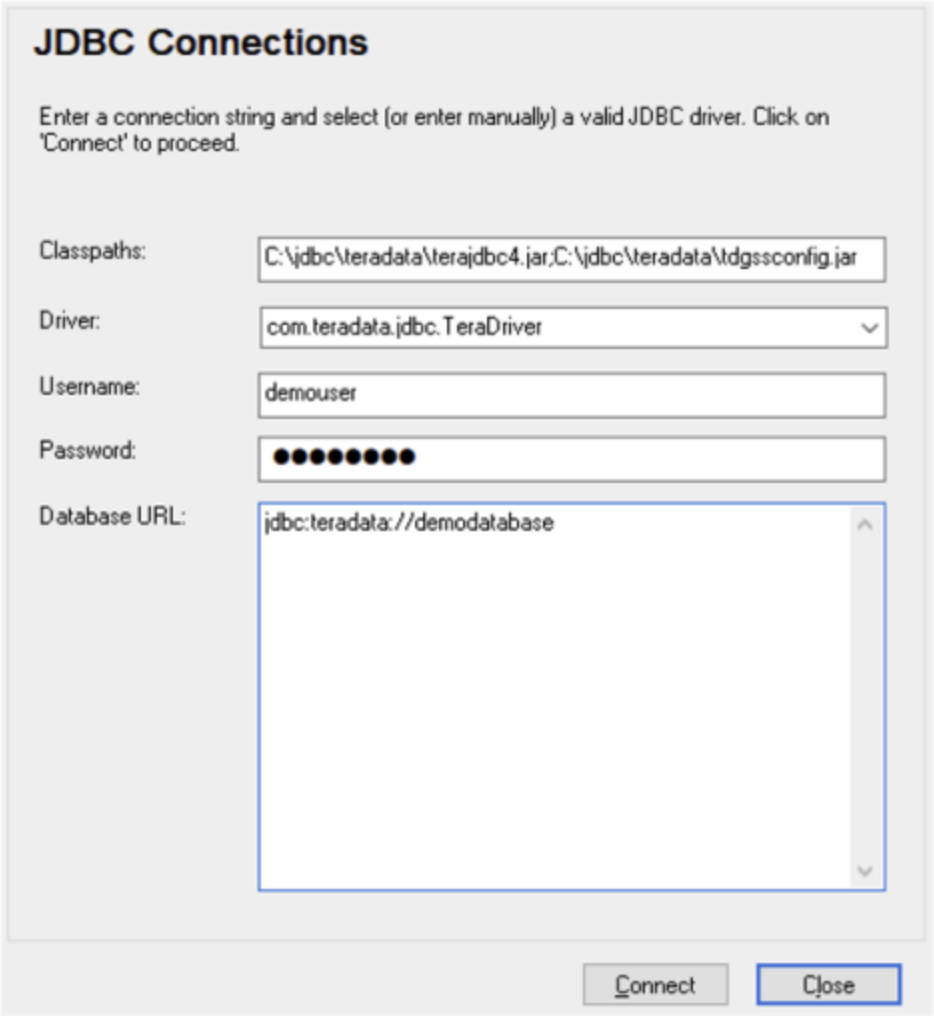
Cet exemple indique comment se connecter à un serveur de base de données Teradata avec JDBC.

**Exigences préalables :**

- JRE (Java Runtime Environment) ou Java Development Kit (JDK) doit être installé. Il peut s'agir soit d'Oracle JDK, soit d'un build open source comme Oracle OpenJDK. MapForce déterminera le chemin vers la Java Virtual Machine (JVM) depuis les emplacements suivants, dans l'ordre suivant : a) le chemin personnalisé JVM que vous avez éventuellement défini dans l'application **Options**, voir Paramètres Java[1091]; b) le chemin JVM trouvé dans le registre Windows ; c) la variable d'environnement `JAVA_HOME`.
- Assurez-vous que la plateforme de MapForce (32-bit, 64-bit) correspond à celle du JRE/JDK.
- Le pilote JDBC (un ou plusieurs fichiers .jar qui fournissent une connectivité à la base de données) doit être installé sur votre système d'exploitation. Dans cet exemple, le pilote JDBC Teradata 16.20.00.02 est utilisé. Pour plus d'informations, voir https://downloads.teradata.com/download/connectivity/jdbc-driver.
- Vous disposez des détails de connexion à la base de données suivants : hôte, base de données, port, nom utilisateur et mot de passe.

**Connexion à Teradata par le biais de JDBC :**

1. Lancer l'assistant de connexion de base de données[163].
2. Cliquez sur **JDBC Connections**.
3. À côté de « Classpaths », saisissez le chemin menant au fichier .jar qui fournit une connectivité à la base de données. Si nécessaire, vous pouvez aussi saisir une liste séparée par points-virgules de chemins d'accès de fichiers .jar. Dans cet exemple, le fichier .jar exigé se trouve sous le chemin suivant : **C:\jdbc\teradata**. Veuillez noter que vous pouvez laisser en blanc le champ « Chemins de classe » si vous avez ajouté le(s) chemin(s) d'accès du fichier .jar à la variable d'environnement CLASSPATH du système d'exploitation (voir aussi Configuration du CLASSPATH[188]).
4. Dans le champ « Pilote », choisir **com.teradata.jdbc.TeraDriver**. Veuillez noter que cette entrée est disponible si un chemin d'accès de fichier .jar valide est trouvé soit dans le champ « Chemins de classe », soit dans la variable d'environnement CLASSPATH du système d'exploitation (voir l'étape précédente).




5. Saisissez le nom d'utilisateur et le mot de passe de la base de données dans les champs de saisie correspondants.
6. Saisissez la chaîne de connexion du serveur de la base de données dans le champ de saisie Database URL, en remplaçant les valeurs soulignées avec celles applicables à votre serveur de base de données.

```
jdbc:teradata://databaseServerName
```

7. Cliquez sur **Connexion**.

### 4.2.1.10.21 Teradata (ODBC)

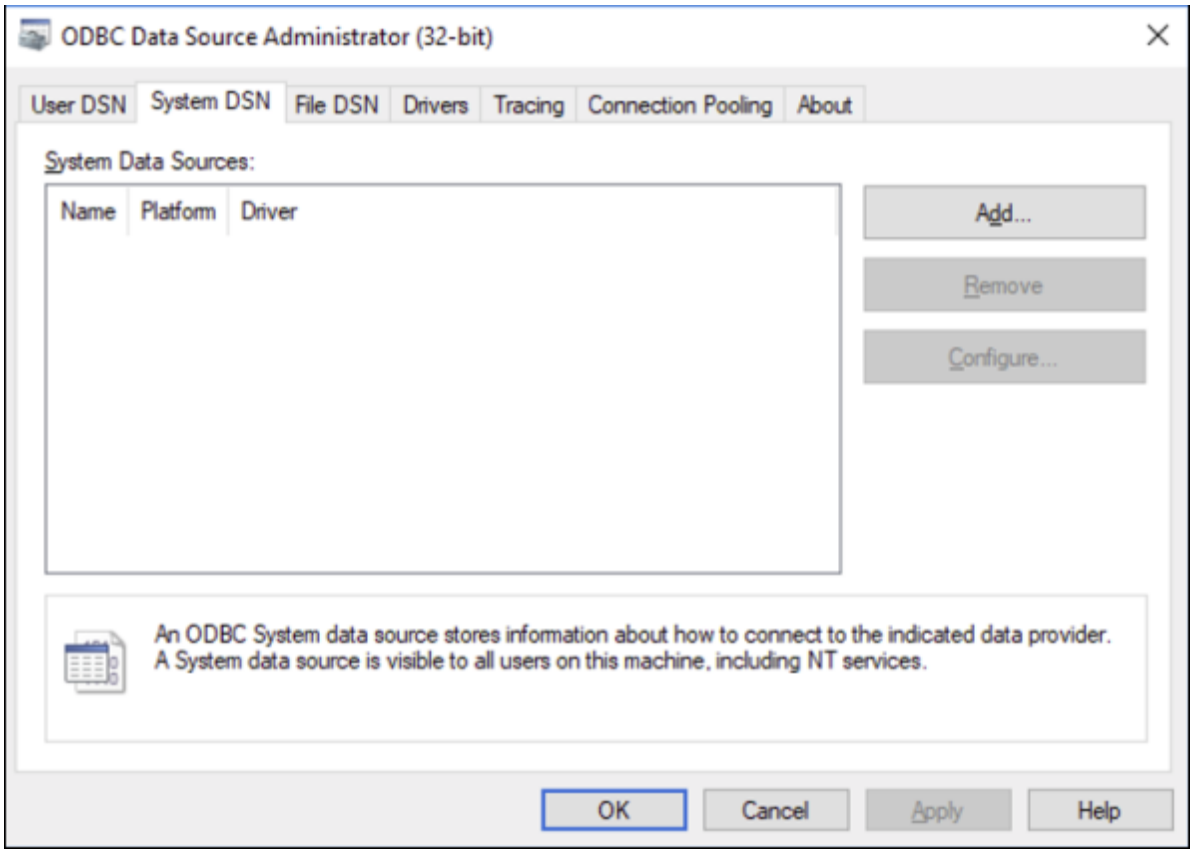
Cet exemple indique comment se connecter à un serveur de base de données Teradata avec ODBC.

**Exigences préalables :**

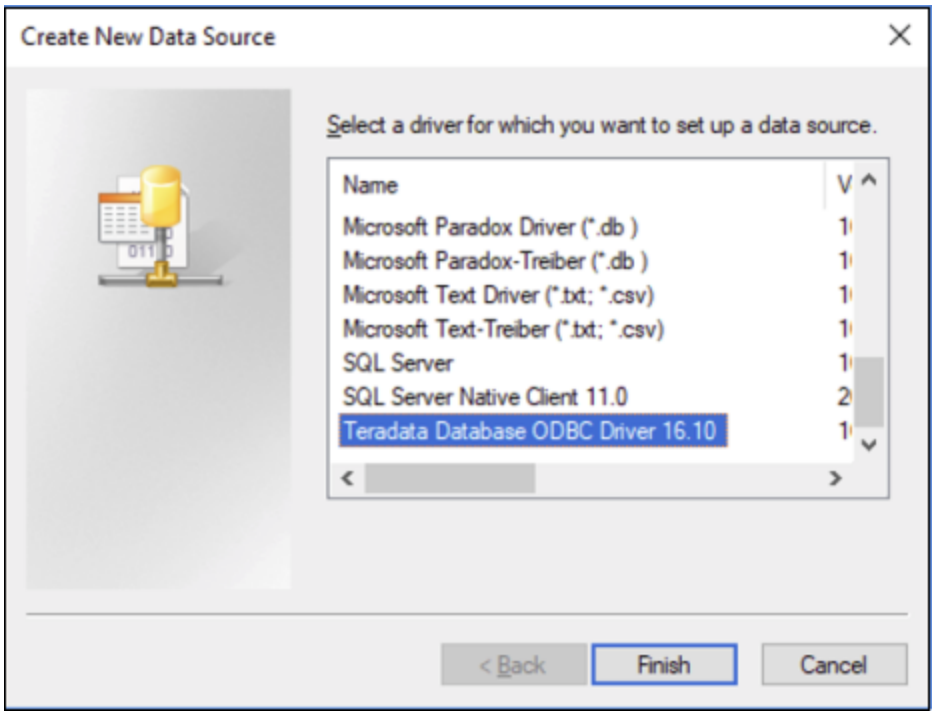
- Le pilote Teradata ODBC doit être installé (voir https://downloads.teradata.com/download/connectivity/odbc-driver/windows. Cet exemple utilise le pilote Teradata ODBC pour Windows version 16.20.00.
- Vous nécessitez les détails de connexion de base de données suivants : hôte, port, nom d'utilisateur et mot de passe.

**Connexion à Teradata par le biais de ODBC :**

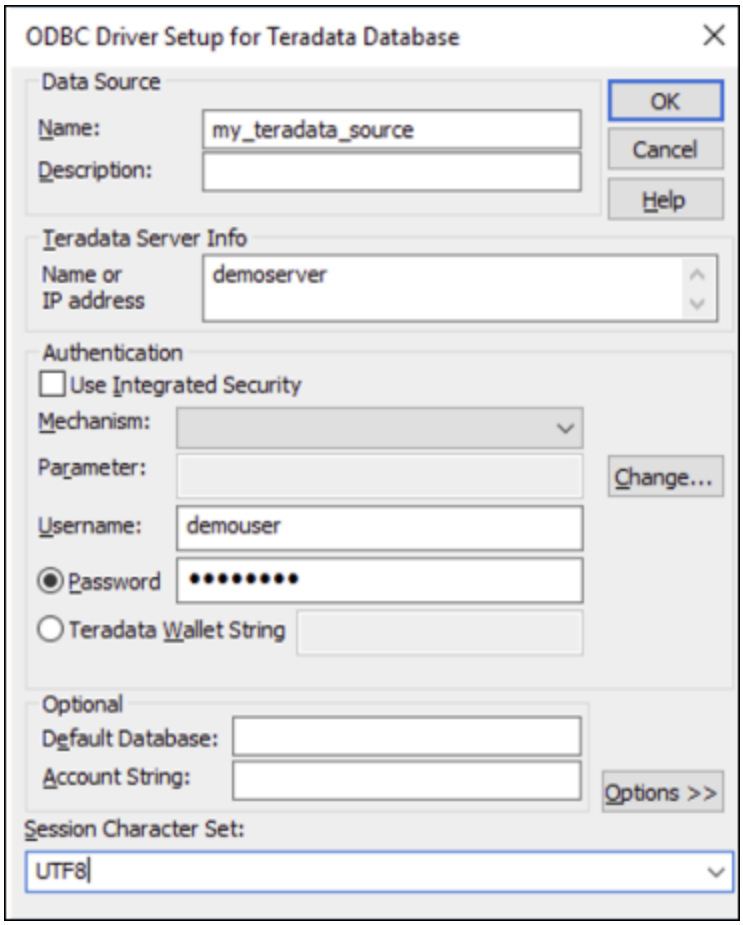
1. Appuyez sur la clé **Windows**, saisissez « ODBC » et sélectionnez **Configurer les sources de données ODBC (32-bit)** depuis la liste de suggestions. Si vous disposez d'un pilote 64-bit, sélectionnez **Configurer les sources de données ODBC (64-bit)** et utilisez MapForce 64-bit aux étapes suivantes.



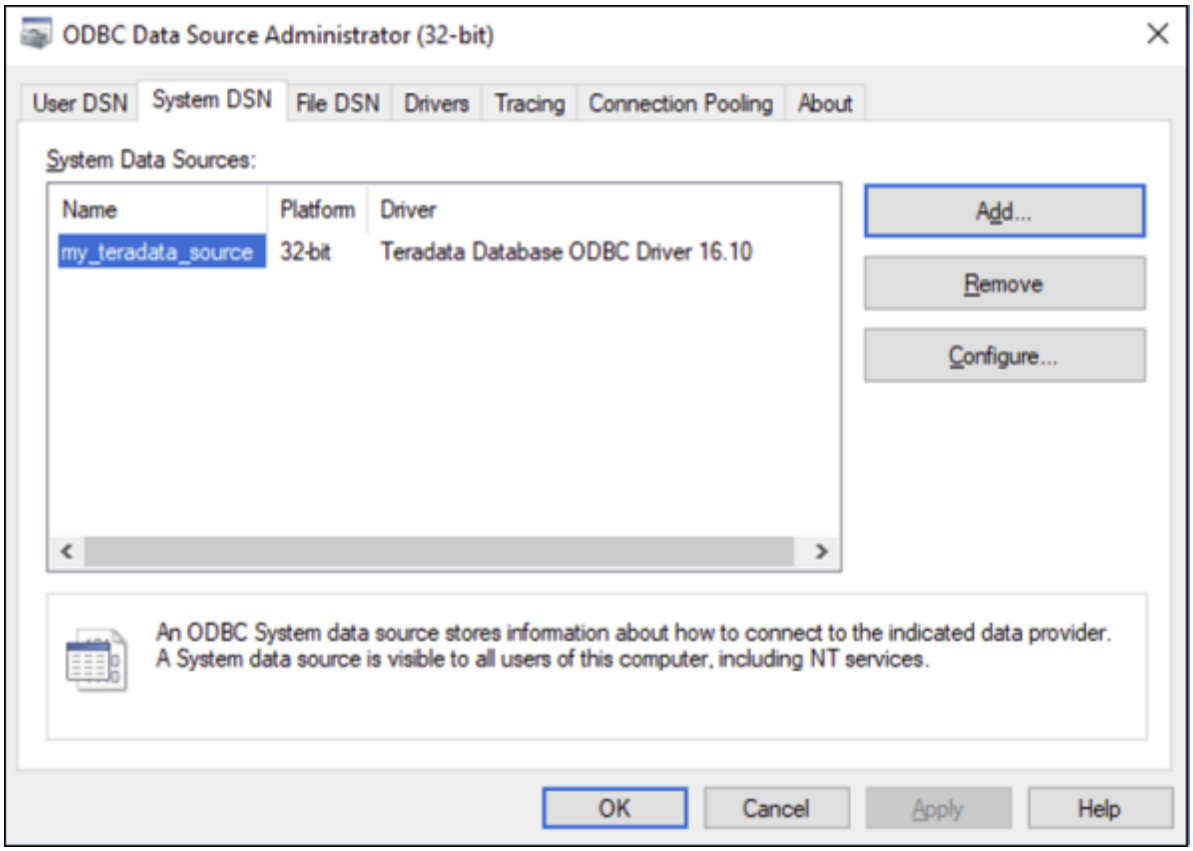

2. Cliquez sur **System DSN**, puis cliquez sur **Ajouter**



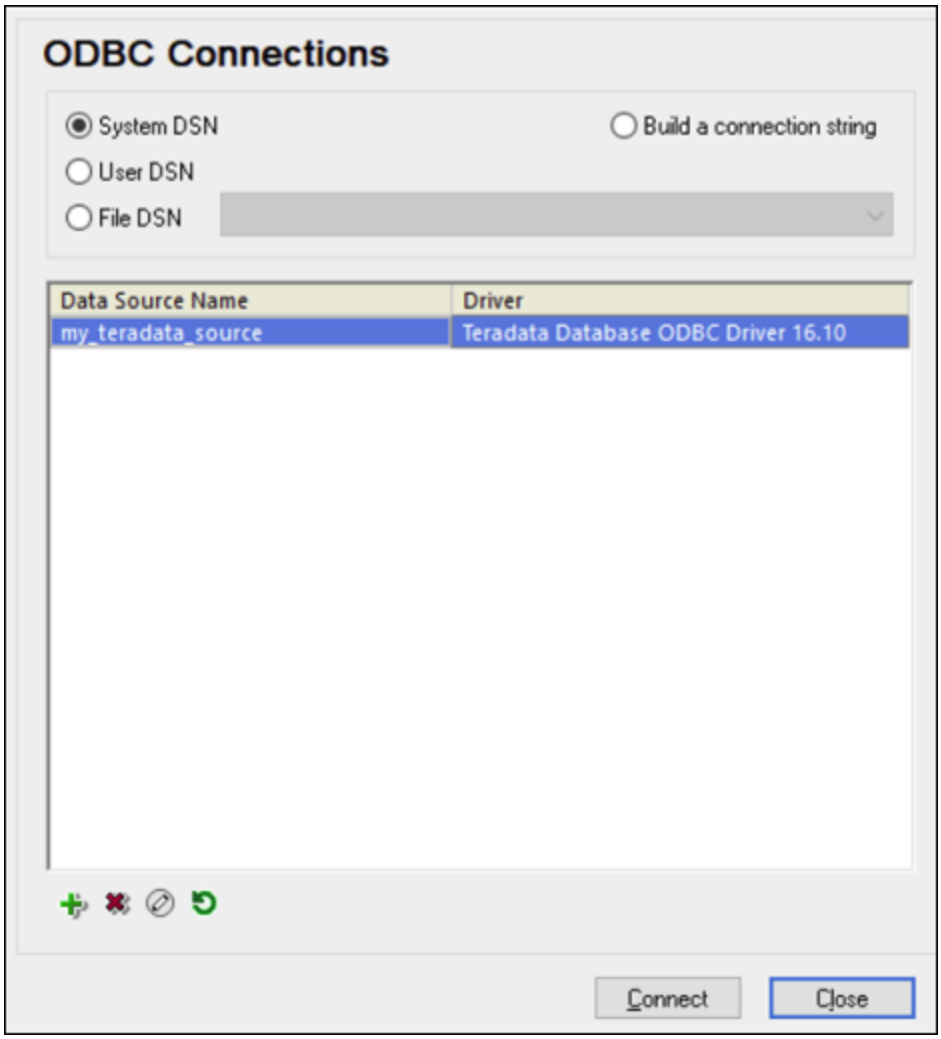

3. Choisissez le **pilote Teradata Database ODBC** et cliquez sur **Terminer**.




4. Saisir le nom et, éventuellement, une description qui vous aidera à identifier cette source de données ODBC à l'avenir. De même, saisissez les identifiants de connexion de base de données (serveur de base de données, utilisateur, mot de passe), et, en option, choisissez une base de données.
5. Cliquez sur **OK**. La source de données apparaît maintenant dans la liste.




6. Exécutez MapForce et [lancez l'assistant de connexion à la base de données](#) 163.
7. Cliquez sur **Connexions ODBC**.




8. Cliquez sur **System DSN**, sélectionnez la source de données créée précédemment, puis cliquez sur **Se connecter**.

Note : si vous obtenez l'erreur suivante : « Le pilote a retourné (ou n'a pas pu retourné) SQL_DRIVER_ODBC_VER: 03.80 invalide », veuillez vous assurer que le chemin menant au client ODBC (par exemple, **C:\Program Files\Teradata\Client\16.10\bin**, si vous l'avez installé à cet endroit, par exemple) existe bien dans votre variable d'environnement PATH de votre système. Si ce chemin est manquant, ajoutez-le manuellement.

## 4.2.2 Procédures générales

Cette section explique comment ajouter une base de données à votre mappage, sélectionner, supprimer et éditer les objets de base de données, gérer les relations de base de données, et configurer les divers paramètres de la base de données.

## Icônes de colonne de base de données

Les tables de bases de données sont représentées par l'icône . Les colonnes de bases de données sont représentées par l'icône . S'il existe un ensemble de contrainte pour la colonne, l'icône de colonne aura un symbole supplémentaire. Si une colonne a plus d'une contrainte assignée, seule la contrainte avec la priorité la plus élevée est décrite dans l'icône de colonne. La priorité des contraintes est décrite dans la table ci-dessous, en commençant par la priorité la plus élevée.

| | |
|---|---|
| | Cette colonne est utilisée comme clé primaire de la table. |
| | Cette colonne a une contrainte unique. |
| | Cette colonne a une de clé étrangère assignée qui référence la clé primaire d'une table différente. |
| | La colonne contient des données XML 310. |
| | Il existe une valeur par défaut définie pour cette colonne. Si aucune valeur n'est fournie dans cette colonne, la valeur par défaut sera insérée à la place. |

## Ajouter une base de données au mappage

Pour pouvoir ajouter une base de données au mappage, vous devez sélectionner un des langages de transformation 22 suivants : Built-in, C++, C#, ou Java. Les bases de données SQLite sont prises en charge uniquement dans Built-In. Si vous avez l'intention de déployer le mappage sur FlowForce Server, exécutez-le avec MapForce Server, ou utiliser des fonctions comme Bulk Transfer 286 et des procédures stockées 321, vous devez sélectionner Built-In.

Une fois que le langage de transformation désiré a été sélectionné, vous pouvez ajouter une base de données au mappage d'une des manières suivantes :

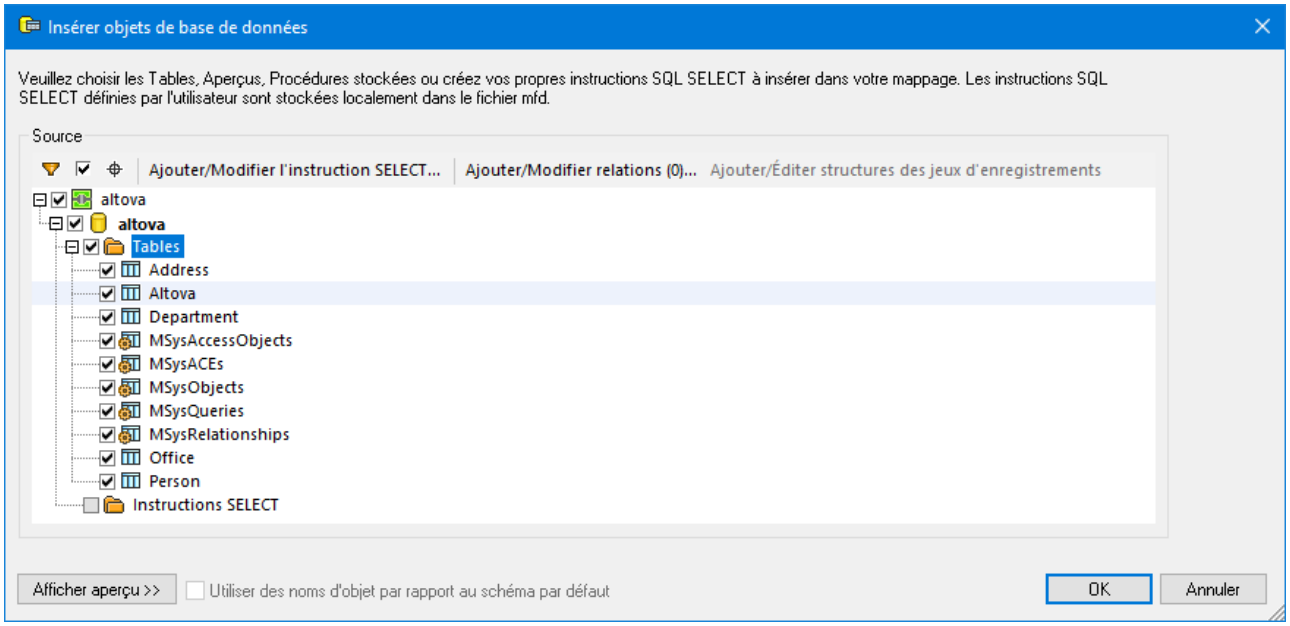
- Sélectionnez **Base de données** dans le menu **Insérer**.
- Cliquez sur le bouton de la barre d'outils.

En choisissant une de ces actions, assistant de connexion de la base de données 163 apparaît, vous guidant à travers les étapes requises pour se connecter à la base de données. Pour plus d'informations concernant la connexion à la base de données, voir Connexion à une base de données 161. Une fois que la connexion de base de données a été établie avec succès, vous êtes invité à sélectionner des objets de base de données qui devraient apparaître dans le mappage (*voir les sous-sections ci-dessous*).

Les bases de données peuvent aussi être ajoutées au mappage comme variables 413. Lorsque vous ajouter une structure de base de données en tant que variable, le même assistant de connexion de base de données que l'assistant apparaît.

## Ajouter des objets de base de données

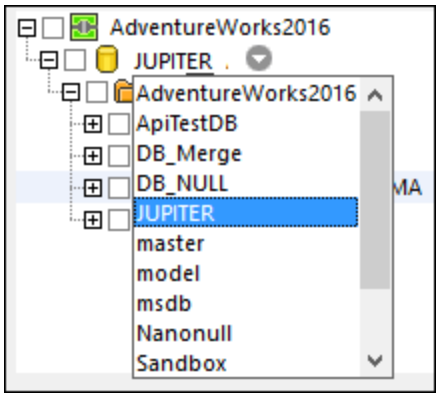
Dès que vous vous êtes connecté à la source de données, vous êtes invités à sélectionner des objets de données que vous voulez inclure dans votre mappage. Le dialogue **Insérer Objets de base de données** affiche la structure de la base de données **`Altova.sqlite`**. Afin d'inclure un objet de base de données dans les mappages, sélectionnez la case à cocher et cliquer sur **OK**. Dans notre exemple, nous avons inclus toutes les tables utilisateur.




*Structure du dialogue Insérer Objets Base de données.*

Le nœud supérieur dans la structure indique la connexion à la base de données. La structure qui s'ensuit varie selon le type de la base de données. Par exemple, les bases de données Oracle et IBM DB2 ont un nœud de schéma sous le nœud de connexion, alors que d'autres types de base de données ont un nœud de catalogue (base de données) . Des caractères gras indiquent le catalogue par défaut (base de données) ou le schéma, tel qu'applicable.

Si votre compte d'utilisateur de base de données a accès à plusieurs bases de données ou schémas sur le serveur, vous pouvez passer à la base de données nécessaire en cliquant sur l'icône (*voir ci-dessous*).



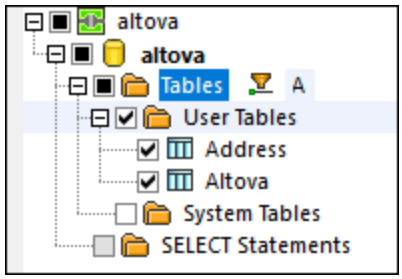

*Options disponibles dans le dialogue Insérer les Objets Base de données*

Les options disponibles dans le dialogue **Insérer Objets de base de données** sont décrits ci-dessous.

- Filtre

  La touche ( **Filter** ) vous permet de filtrer des objets par leur nom. Une fois que vous cliquez sur la touche **Filter**, une icône de filtre est disponible à côté des objets qui prennent en charge les filtres (dans

cet exemple, `Tables`). Cliquez sur l'icône du filtre pour choisir une des options suivantes : *No Filter*, *Contains*, *Does Not Contain*, *Start With*, *Ends With*, *Equals*. Dans notre exemple, nous avons décidé d'inclure uniquement les tables dont les noms commencent par `A` (*voir ci-dessous*).




- Afficher uniquement les objets cochés

  Le bouton (**Afficher uniquement les objets activés**) affiche uniquement des items avec les cases à cocher.

- Localisateur d'objet

  La touche (**Object Locator**) vous permet de trouver les items de base de données spécifiques. Sélectionner un objet particulier ou saisir son nom dans la liste de choix qui apparaît dans la zone inférieure du dialogue.

- Ajouter/Éditer instruction SELECT

  La touche **Ajouter/Modifier une instruction SELECT** vous permet d'ajouter et d'éditer des instructions SELECT personnalisées pour la base de données actuelle. Les données retournées par ces instructions deviennent disponibles en tant que source de mappage. Pour plus d'information, voir Personnaliser Instructions SELECT[264].

- Ajouter/Éditer relations

  Le bouton **Ajouter/Modifier relations** vous permet de définir des relations de clé primaire locale et de clé étrangère entre les champs dans la base de données, en plus de celles qui peuvent déjà existent dans la base de données. Pour plus d'informations, voir Relations locales[274].

- Ajouter/Éditer les structures « recordset »

  La touche **Ajouter/Modifier structures des jeux d'enregistrements** s'applique aux bases de données qui prennent en charge les procédures stockées[321]. Le bouton est activé uniquement si une procédure stockée est actuellement sélectionnée avant l'arborescence de la base de données.

- Afficher Aperçu

  La touche **Afficher aperçu** vous permet d'apercevoir rapidement les données de la table ou du mode sélectionné actuellement. Notez que vous pouvez aussi choisir et requêter une base de données indépendamment du processus de mappage, en utilisant le Navigateur de base de données. Pour plus d'information, voir Bases de données Requête[300].

- Utiliser des noms d'objet relatifs au schéma par défaut

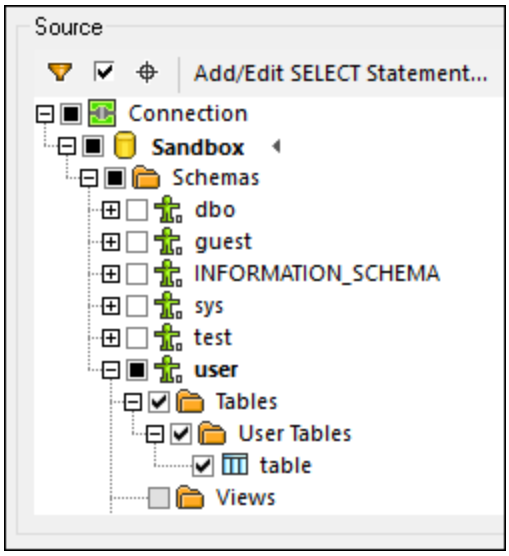
Dans MapForce, rendre des noms d'objets de la base de données relatifs au schéma est important si vous prévoyez de passer vers une base de données plus tard. Cela est également utile si le schéma de base de données a été renommé sur le serveur ; et vous devrez mettre à jour le mappage. Si le nouveau schéma a la même structure que celle utilisée au moment du design de mappage, vous pouvez y passer sans devoir changer les connexions de mappage manuellement.

Veuillez noter les points suivants :

- Utiliser les noms d'objet relatifs au schéma par défaut est possible pour des bases de données qui prennent en charge les schémas : IBM DB2, IBM Informix, IBM Db2 for i (iSeries), Oracle, PostgreSQL, Progress OpenEdge, SQL Server et Sybase.
- Il n'est pas possible d'utiliser des noms relatifs si le composant de base de données inclut des relations locales[274] ou des instructions SELECT en tant que tables virtuelles[264].
- La case à cocher *Utiliser des noms d'objets relatifs au schéma par défaut* affecte le code de programmation C#, C++, and Java généré. Si cette case est sélectionnée, toutes les références de base de données également deviennent relatives dans le code généré.

Pour rendre les noms d'objets de la base de données relatifs au schéma par défaut, suivez les étapes suivantes :

1. Ouvrez le dialogue **Insérer Objets Base de données** ou cliquez avec la touche de droite sur la barre de titre d'un composant de base de données et sélectionnez **Ajouter/Supprimer/Éditer des objets de base de données** depuis un menu contextuel.
2. Sélectionnez un ou plusieurs objets qui *appartiennent au schéma par défaut* ou pour le *catalogue par défaut (base de données) et le schéma*. La base de données par défaut et le schéma sont affichés en gras. Dans l'exemple ci-dessous, le catalogue par défaut est `Sandbox`, et le schéma par défaut est `user`. Cette structure est spécifique pour les bases de données SQL Server et peut varier dans d'autres types de base de données.




3. Cocher la case *Utiliser les noms d'objet relatifs au schéma par défaut*. Veuillez noter que cette case à cocher est grisée si la base de données ne prend pas en charge les noms d'objet relatifs.

Si les objets que vous requérez dans le mappage se trouvent dans un schéma différent (de celui par défaut), vous disposez des alternatives suivantes :

- Se connecter en tant qu'un autre utilisateur de base de données qui a accès au schéma par défaut requis.
- Si vous avez les privilèges requis, reconfigurez le serveur de base de données de manière à changer le schéma par défaut de l'utilisateur de la base de données existante.

L'exemple ci-dessous affiche comment modifier le schéma par défaut de l'utilisateur de la base de données. L'exemple est basé sur le Serveur SQL et assume que le catalogue est `Sandbox` et l'utilisateur ainsi que le schéma existe déjà

```
USE [Sandbox]
GO
ALTER USER [test_user] WITH DEFAULT_SCHEMA=[test_schema]
GO
```

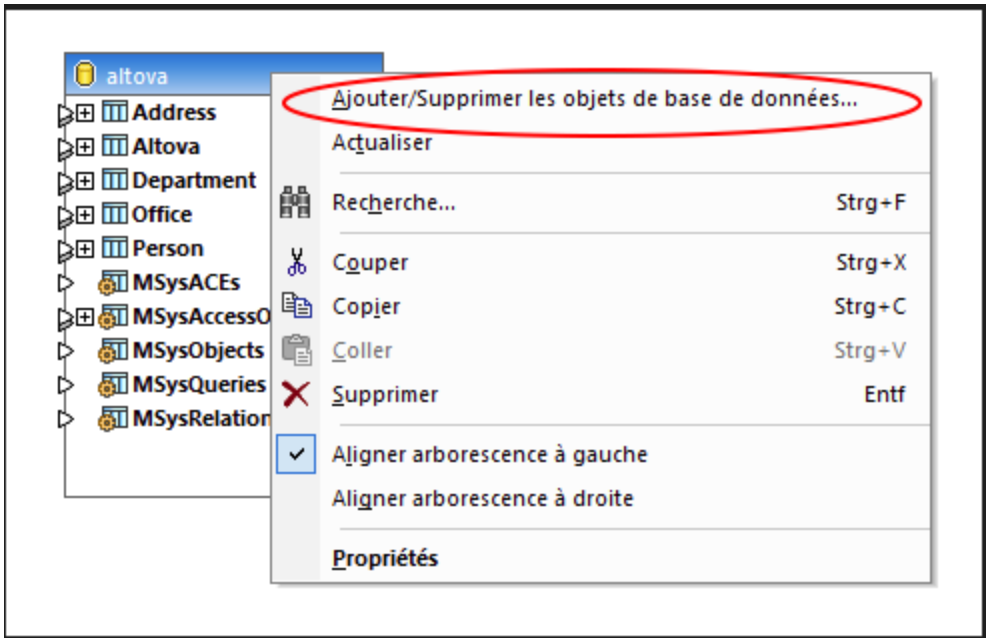
*Passer à un schéma/BD sans perdre de connexions de mappage*
Lorsque les noms d'objets de la base de données sont relatifs au schéma, vous pouvez passer à une nouvelle base de données ou un schéma *sans perdre les connexions de mappage*. Les options suivantes sont disponibles :

- Ouvrir les paramètres de composant de base de données[256] et cliquer sur **Changer**. Suivre les instructions pour vous connecter à la nouvelle base de données *en tant que nouvel utilisateur*. Si la nouvelle base de données a la même structure, toutes les connexions dans le mappage seront mises à jour automatiquement. Cela signifie que ces connexions correspondront au *catalogue et schéma par défaut du nouvel utilisateur de base de données*.
- Si vous souhaitez une nouvelle base de données de manière récurrente, il est recommandé de définir la connexion de base de données en tant que Ressource Globale[881]. Par exemple, la Ressource globale pourrait avoir deux configurations : une par défaut pour la base de données de développement et une configuration de production.

Si des objets de base de données apparaissent en rouge après avec changé de base de données, cela indique qu'ils n'existent pas dans le nouveau schéma de base de données.

## Éditer les objets Base de données

Pour modifier des objets de base de données, cliquez avec la touche droite de la souris sur le composant de base de données et sélectionnez **Ajouter/Supprimer/Éditer Objets de base de données** depuis un menu contextuel(*voir ci-dessous*). Ceci ouvre le dialogue **Ajouter/Supprimer/Éditer objets de base de données**, qui vous permet de définir les mêmes paramètres et propriétés que dans le dialogue **Insert Database Objects**.




### Suggestions de remplissage automatique SQL

Quand vous saisissez des instructions SQL dans certains contextes, MapForce peut suggérer des saisies de texte automatiquement. La saisie automatique est disponible dans l'Éditeur SQL (voir volet Requête BD[300]), la zone de texte *SQL personnalisé* dans le dialogue Actions de table de base de données[280], et le dialogue Ajouter Instruction SELECT[264].

Pour désactiver les suggestions de saisie semi-automatique, suivez les étapes suivantes :

1. Choisir l'item menu **Outils | Options** ou appuyez sur **Ctrl+Alt+O**.
2. Ouvrez la section *Base de données | Éditeur SQL*.
3. Effacez la case à cocher *ouvrir automatiquement* dans la section *Assistants à la saisie*.

Pour invoquer manuellement les suggestions de remplissage automatique, appuyez sur **Ctrl+Space**. :

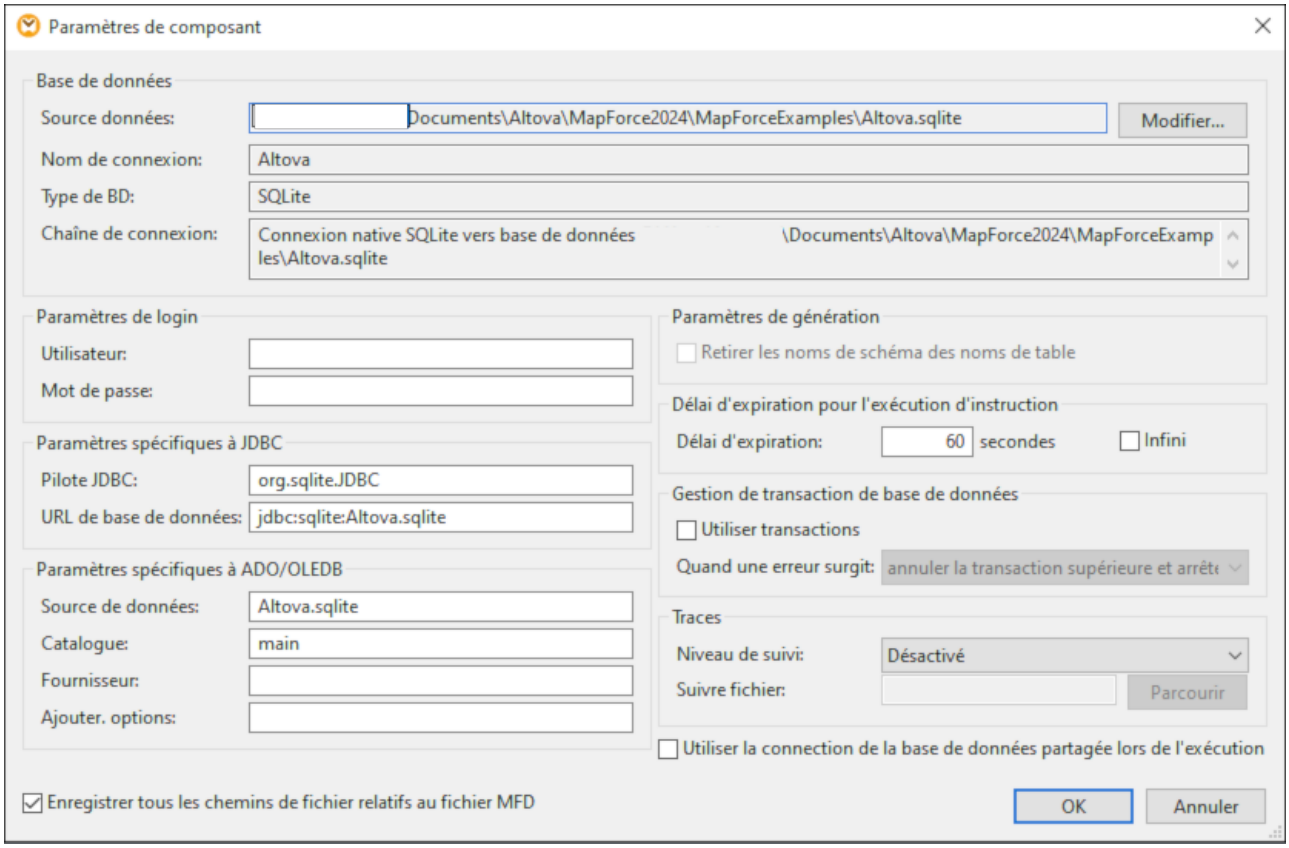
Pour en savoir plus sur les paramètres associés à la base de données, voir Base de données[1092].

## 4.2.2.1 Paramètres de composant de base de données

Après avoir ajouté un composant de base de données dans la zone de mappage, vous pouvez configurer divers paramètres de bases de données 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 :

- Double-cliquez sur la barre de titre du composant.
- Cliquez avec la touche de droite sur le composant, puis cliquez sur **Propriétés**.

- Sélectionnez le composant dans le mappage. Cliquez sur le menu **Composant**, et sélectionnez **Propriétés** depuis le menu contextuel.




Les paramètres disponibles sont recensés ci-dessous.

## Base de données

Ce groupe affiche des informations de connexion de base de données. Cliquez sur **Changer** pour sélectionner une base de données différente ou pour redéfinir les objets de la base de données dans le composant de base de données existant. Les connecteurs vers les tables portant le même nom seront retenus. Vous pouvez aussi changer les tables dans le composant, en cliquant sur le composant de base de données et en sélectionnant **Ajouter/SupprimerÉditer Objets de base de données**.

### Source de données

Spécifie le chemin absolu ou relatif de la source de données actuelle.

### Nom de connexion

Spécifie le nom de la connexion. Ce nom est généré automatiquement par MapForce. Généralement, il s'agit du même nom que celui du nom de source de données, mais il peut s'agir d'un nom « alias » si vous vous connectez à des Ressources Globales Altova[881]. S'il existe de multiples composants de base de données avec la même connexion dans le mappage, le nom de connexion aura l'aspect suivant : `<connection1>`, `<connection2>`, etc.

- Type de base de données

  Spécifie le type de base de données (par ex., SQLite).

- String de connexion

  Affiche le string de connexion de la base de données actuelle. Ce champ en lecture seule est généré sur la base de l'information que vous fournissez lors de la création ou de la modification de la connexion de base de données.

## Paramètres de login

Les paramètres de login sont utilisés pour les cibles de la génération du code et le moteur d'exécution built-in.

- Utilisateur

  Vous permet de changer de nom d'utilisateur pour vous connecter à la base de données. Obligatoire si la base de données nécessite un nom d'utilisateur pour pouvoir se connecter.

- Mot de passe

  Vous permet de changer le mot de passe pour vous connecter à la base de données. Obligatoire si la base de données nécessite un mot de passe pour pouvoir se connecter.

## Paramètres spécifiques à JDBC

Les paramètres spécifiques à JDBC sont pertinents quand le mappage contient une connexion JDBC et est exécutée en générant le code Java ou par MapForce Server.

**Note :** Les connexions ADO, ADO.NET et ODBC sont converties en JDBC (et les paramètres JDBC ci-dessous s'appliquent) lorsque le mappage est exécuté sur un appareil Linux ou macOS. Pour plus de détails, voir *Mappages de base de données dans des environnements d'exécution variés* 159.

- Pilote JDBC

  Affiche le pilote actuellement actif pour le composant de base de données. Le pilote par défaut est saisi automatiquement lorsque vous définissez le composant de base de données. Vous pouvez modifier le pilote saisi ici selon vos besoins. Assurez-vous que la syntaxe de l'entrée dans le champ *URL de base de données* est conforme au pilote spécifique que vous avez choisi.

- URL de base de données

  URL de la base de données actuellement sélectionnée. Assurez-vous que cette entrée soit conforme à la syntaxe du pilote JDBC du pilote spécifique saisi dans le champ du pilote *JDBC*

## Paramètres spécifiques à ADO/OLEDB

Les paramètres spécifiques à ADO/OLEDB sont pertinents quand le mappage contient une connexion ADO et est exécutée par le code C# généré, le code C++, ou par MapForce Server exécuté sur Windows. Pour plus de détails, voir *Mappages de base de données dans des environnements d'exécution variés* 159. Les paramètres *Source de données* et *Catalogue* ne sont pas utilisés par le moteur d'exécution intégré.

- Source de données

  Affiche le nom de la source de données ADO

- Catalogue

  Affiche le nom du catalogue ADO.

- Fournisseur

  Affiche le fournisseur actuellement actif pour le composant de base de données.

- Ajouter des options

  Affiche toute option de base de données supplémentaire.

## Paramètres de génération

Les paramètres de génération s'appliquent à toutes les cibles de génération de code ainsi qu'à l'appareil d'exécution intégré. L'option *Retirer les noms de schéma depuis les noms de table* vous permet de supprimer des noms de schéma de base de données du code généré et ne retiennent que les noms de table. Veuillez noter que cette option ne fonctionne que pour les instructions SELECT SQL générées par MapForce. Les instructions SQL définies par l'utilisateur[264] ne seront pas modifiées.

L'option *paramètres de génération* est prise en charge pour la compatibilité inversée doit être évitée. Afin de rendre les noms d'objet de table de données en que relatives au schéma par défaut, utilisez l'approche décrite dans *Utiliser les noms d'objet relatifs au schémas*[253].

## Timeout pour l'exécution d'instruction

Lorsqu'une base de données est utilisée en tant que composant cible, des « timeouts » d'exécution peuvent se produire en raison de la disponibilité du serveur, du trafic, des déclencheurs à longue durée, et d'autres facteurs.

- Délai d'expiration

  Définit une période de temps, en secondes, pendant laquelle le moteur d'exécution doit attendre pour une réponse de la base de données avant d'abandonner l'exécution de l'instruction de la base de données. Le délai d'expiration par défaut est de 60 secondes.

- Infini

  Si elle est activée, cette option indique au moteur d'exécution de ne jamais effectuer de time out.

**Note :** le timeout pour l'exécution de l'instruction n'est pas applicable pour les bases de données SQLite.

## Gestion de la transaction de base de données

Au cours de l'exécution d'un mappage qui a un composant de base de données, il pourrait y avoir diverses erreurs liées à la base de données (par ex., clés d'indice doubles, valeurs NULL insérées dans les colonnes non-NULL, etc.). Pour pouvoir restaurer vos données de base de données en cas d'erreur, activez la gestion de

transaction de la base de données. Vous pouvez activer une restauration de transaction au niveau du composant de base de données (*paramètre actuel*), au niveau de l'action de table[284], et au niveau de la procédure stockée[334]. Pour plus d'information sur les possibles scénarios de gestion de la transaction, voir Restauration de la transaction : Scénarios[293].

Certains mappages peuvent contenir des composants de bases de données multiples qui peuvent avoir les mêmes ou différentes connexions de bases de données. Le résultat de ces mappages en cas d'erreurs liées à la base de données dépend du moteur d'exécution :

- Si le mappage est effectué avec MapForce, seul un composant de cible peut être exécuté lorsque le mappage est exécuté. Il s'agit du composant dans lequel la touche  est activée. Si une erreur de base de donnée se produit dans ce composant, et la coche *Utiliser les transactions* est activée, tous les changements effectués par le composant seront annulés.
- Si le mappage est effectué avec MapForce Server ou un programme généré par MapForce, tous les composants de cible sont exécutés, de manière séquentielle. Dans ce cas, lorsqu'une erreur de base de données se produit, l'annulation se produira pour le composant de base de données à l'endroit où l'erreur s'est produite. Le mappage s'arrêtera ou continuera à exécuter le composant cible suivant selon la valeur que vous avez sélectionnée depuis la liste déroulante *Lorsqu'une erreur se produit* ( *voir ci-dessous*).

Utiliser des transactions

Permet le traitement de la transaction pour un composant de base de données cible. Le processus de transaction est activé pour toutes les tables du composant de base de données lorsque vous sélectionnez cette option. Activez une gestion de transaction au niveau du composant de la base de données comprend tous les changements de base seul de données qui seront restaurés dans le cas d'une erreur associée à la base de données.

Lorsqu'une erreur survient

Si vous avez sélectionné la case à cocher *Utiliser transaction*, vous pouvez choisir quoi faire lorsqu'une erreur de base de données survient :

- *Restaurer transaction du haut et arrêter :* La transaction qui comprend tous les changements de base de données est renvoyée, et l'exécution du mappage s'arrête.
- *Annuler cette transaction et continuer :* Comme ci-dessus, mais le mappage continue d'exécuter après l'annulation (par ex., pour traiter un deuxième composant cible).

Au niveau de composant de base de données, vous contrôlez si le traitement devrait continuer pour d'autres composants cible. Par exemple, dans un mappage XML-DB-JSON, une erreur est apparue dans le composant de base de données. Dans ce cas, le fichier JSON peut encore être traité et extrait si vous avez activé l'option *Restaurer la transaction du haut et continuer*.

## Traçages

Lorsqu'un mappage écrit des données dans une base de données, vous pouvez activer le traçage et la journalisation d'erreur de la base de données. Le traçage est utile si vous souhaitez suivre tous les changements devenir dans la base de données lors de l'exécution du mappage. Les changements effectués dans la base de données sont journalisés dans un rapport de trace. S'il existe des erreurs pendant l'exécution, ces erreurs seront également journalisées. Le traçage est compatible uniquement avec le langage de transformation Built-In.

Vous pouvez permettre le traçage à différents niveaux :

- *Niveau de composant de base de données :* Le traçage à ce niveau pourrait être utile pour les mappages qui ont des composants de base de données cible et vous devez assurer le traçage uniquement pour certains d'entre eux. L'activation du traçage au niveau du composant de base de données l'active automatiquement pour toutes les tables et les procédures stockées dans le composant. Afin de pouvoir être tracé, les tables pertinentes et procédures stockées doivent être connectées aux nœuds source.
- *Table*[285] *ou niveau de procédure stockée*[334]*:* Vous pouvez décider si vous voulez activer le traçage pour une table spécifique ou une procédure stockée. Au niveau de la table, le traçage inclut des événements associés aux actions de table (par ex., *Insert All*). Dans le cas de procédures stockées, les événements liés à l'appel de procédure stockée sont tracés.
- *Niveau de champ de base de données*[286] *:* Par défaut, tous les champs sont tracés, mais vous pouvez exclure certains champs du rapport de trace ou choisir d'inclure certains champs uniquement dans le cas d'une erreur.

De manière importante, les trois niveaux ci-dessus sont hiérarchiques. Ceci signifie qu'afin de définir le traçage à un niveau inférieur, vous devez d'abord activer le traçage au niveau du parent. Par exemple, si vous devez définir le traçage au niveau de la table, vous devez d'abord activer le traçage au niveau du composant de base de données. Le même principe s'applique lorsque vous réduisez le niveau de traçage. Par exemple, si vous limitez le traçage uniquement à des erreurs au niveau du composant de base de données, il n'est pas possible d'utiliser le traçage complet au niveau de la table de la procédure stockée.

MapForce vous permet de définir les options de traçage suivantes :

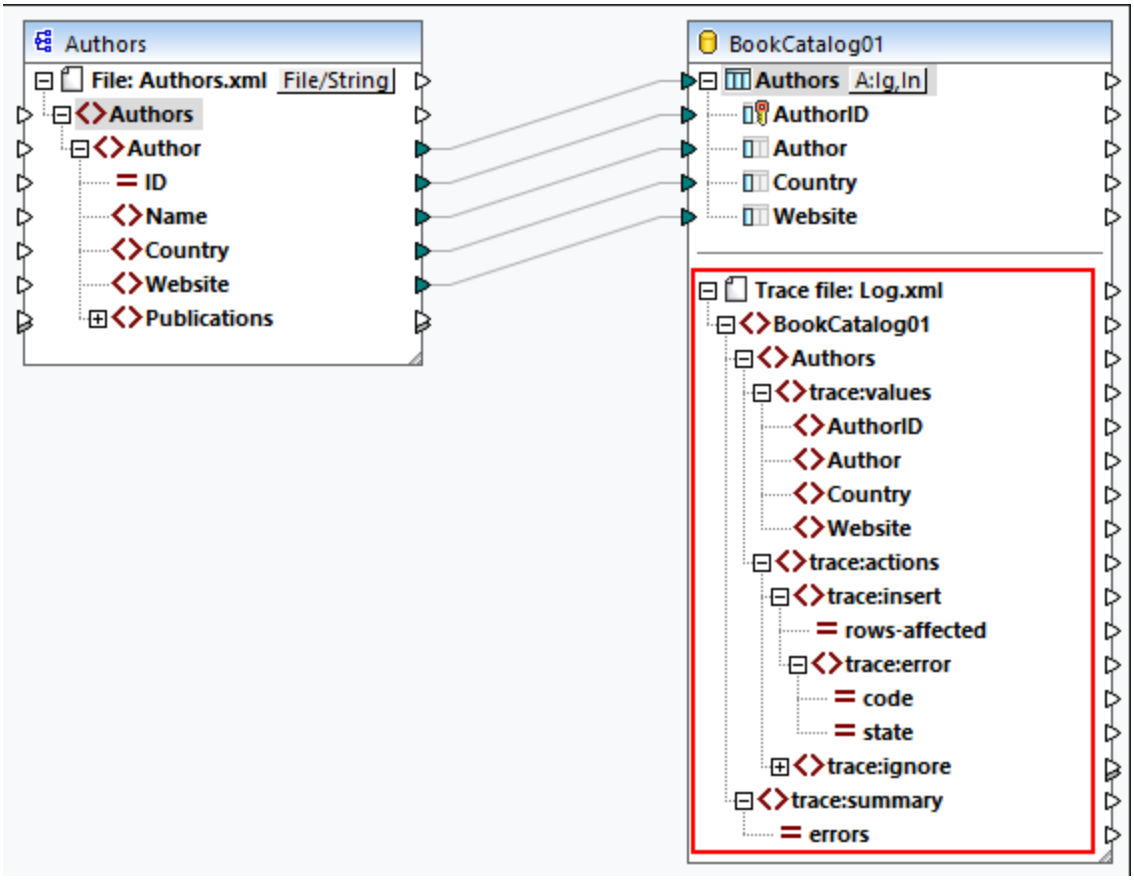
Niveau de traçage

Si le traçage est activé, les actions effectuées par le mappage par rapport à la base de données sont consignées dans un fichier de traçage. Vous pouvez choisir de journaliser toutes les actions, uniquement les erreurs ou désactiver le traçage complètement.

Fichier de traçage

Spécifie le fichier sur lequel les informations de traçage de base de données seront écrites lorsque le mappage est exécuté. Ce chemin peut être absolu ou relatif et est influé par la case à cocher du fichier *Enregistrer tous les chemins relatifs au fichier MFD*. Le fichier de traçage est en format XML. Si vous préférez que le fichier log soit sous un autre format que XML, vous pouvez mapper les données en de celui-ci vers un autre genre de composant pris en charge par MapForce (par ex., un fichier texte, une autre base de données, etc.).

*Structure du fichier de traçage*
Lorsque le traçage est activé pour un composant de base de données, une structure de traçage devient disponible dans la partie inférieure du composant (*capture d'écran ci-dessous*).




La capture d'écran ci-dessus affiche que le nœud du niveau supérieur dans la structure de traçage est le nom du fichier de traçage (`Log.xml`). Le reste de la structure de traçage est modelé basé sur la structure des tables de base de données ou des procédures stockées qui participent au mappage. Dans cet exemple, l'élément top a le même nom que la base de données. L'élément `BookCatalog01` a deux éléments enfant : `Authors`, Books, et `trace:summary`. Les éléments `Authors` reflètent la structure des tables qui sont ajoutées au composant de la base de données. L'élément `trace:summary` contient un attribut `errors` qui rapporte le nombre d'erreurs rencontrées.

L'élément `Authors` contient les deux éléments enfant : `trace:values` et `trace:actions`. La structure `trace:values` affiche toutes les colonnes de la table de la base de données. Par défaut, toutes les colonnes sont tracées, mais vous pouvez changer ceci en configurant le traçage au niveau du fichier de base de données (*voir ci-dessus*). En ce qui concerne les procédures stockées, cette structure affiche les paramètres de la procédure stockée.

L'élément `trace:actions` inclut l'information sur toutes les actions qui sont définies pour cette table de base de données particulière. Dans notre exemple, deux actions ont été configurées pour la table `Authors` : *Ignore If* et *Insert Rest*. Chaque action de traçage possède un attribut `rows-affected` qui spécifie combien de lignes seront touchées par l'action de base de données respective. L'élément `trace:error` est rempli uniquement si une erreur de base de données se produit pendant l'exécution de mappage. Cet élément de produit contient deux attributs : `code` et `state`. Le texte de l'erreur et les valeurs d'attribut sont fournis par le pilote de base de données et seront donc différents pour différentes bases de données.

*Fichier de traçage dans le volet Sortie*
Pour consulter le fichier de traçage, cliquez sur le volet Sortie. Notez que le rapport de trace affiché dans le volet Sortie est à des fins d'information et ne reflète pas les résultats d'exécution actuels. Pour produire un rapport de trace actuel, exécutez le script SQL [279] depuis le volet Sortie. Un exemple d'un fichier de traçage est en fonction ci-dessous.

```
<BookCatalog01>
      <Authors>
            <trace:values>
                  <Author>Neil Gaiman</Author>
                  <Website>www.neilgaiman.com</Website>
            </ trace:values>
            <trace:actions>
                  <trace:ignore/>
            </ trace:actions>
      </ Authors>
      <Authors>...</Authors>
      <Authors>...</Authors>
      <trace:summary errors="0"/>
</BookCatalog01>
```

Pour plus d'information sur le traçage, voir le *Scénario 1* dans la Restauration de la transaction : Scénarios [293].

## 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**). Utiliser des chemins relatifs si vous avez l'intention d'exécuter le mappage avec MapForce Server sur un système d'exploitation différent. Voir aussi Utiliser des chemins relatifs et absolus [47].

## Utiliser une connexion de base de données partagée au moment de l'exécution

Cette option vous permet de choisir si plusieurs composants de base de données qui utilisent la source de données et figurent dans le même mappage devrait partager la même connexion de base de données. Par défaut, cette option est désactivée ; autrement, elle peut changer le comportement du mappage, en particulier quand la connexion est partagée entre un ou plusieurs composants source et cible.

Partager la même connexion de la base de données vous permet de résoudre de multiples questions, par exemple, avec les verrouillages de table/ligne, l'isolation de la transaction, et le nombre de connexions de serveur (*voir les détails ci-dessous*).

- Lorsque vous lisez une ligne depuis la table et tentez de mettre à jour la même ligne, il se pourrait (dépendant du vendeur) que vous ayez une erreur de verrouillage de la table/ligne. Avec une connexion de base de données partagée, la question peut être évitée.
- Avec le partage de connexion activé, vous serez à même de changer déjà les lignes qui sont wrappées dans une transaction. Avec les connexions séparées, seuls les changements validés sont visibles.
- Le partage de connexion aide également à réduire le nombre de connexions de la base de données, qui vous permet de réduire à son tour le temps total de traitement pour les mappages avec un grand nombre de composants de base de données, mais utilise la même source de données. La procédure

de connexion à la base de données peut être longue, en particulier avec les instances du serveur cloud sur un serveur lent ou lorsque le serveur de la base de données est occupé.

### 4.2.2.2 Instructions SELECT personnalisées

MapForce vous permet de créer des instructions SELECT SQL avec ou sans paramètres. Ces instructions sont représentées comme structures table-like, desquelles vous pouvez mapper des données dans d'autres composants. Par exemple, vous pouvez créer une instruction personnalisée pour joindre des tables, filtrer vos données de base de données, et définir les paramètres qui peuvent accepter les valeurs d'un autre composant dans le mappage.

Les instructions SELECT SQL sans paramètres sont pris en charge dans C++, C#, Java, et les langues Built-In. Les instructions SELECT SQL avec des paramètres d'entrée sont compatibles uniquement avec un langage de transformation Built-In.

*Créer/Éditer/Supprimer une instruction SELECT*
Pour ajouter une instruction SELECT au composant de base de données, suivez les instructions ci-dessous.

1. Cliquer sur la barre de titre d'un composant de base de données et choisir **Ajouter/Supprimer/Éditer des objets de base de données** depuis un menu contextuel. Comme alternative, sélectionnez le composant de base de données sélectionnez la commande de menu **Composant | Ajouter/Supprimer/Éditer les objets de base de données**.
2. Dans le dialogue **Ajouter/Supprimer/Éditer les objets de base de données**, suivez une des étapes suivantes :

   - Pour saisir une instruction SELECT personnalisée, cliquez sur la touche **Ajouter/Éditer instruction SELECT**.
   - Pour générer l'instruction SELECT depuis une table existante, cliquez avec la touche de droite sur une table et choisir **Générer et ajouter une instruction SQL** depuis le menu contextuel. Vous pourrez éditer l'instruction générée ultérieurement.

Pour modifier une instruction SELECT existante, procédez comme suit :

- Cliquer avec la touche de droite sur l'instruction SELECT que vous souhaitez supprimer et choisir **Éditer l'instruction SELECT**.
- Cliquez avec la touche de droite à de la base du composant et sélectionnez **Ajouter/Supprimer/ Éditer Filtre** depuis le menu contextuel. Puis double-cliquez sur l'instruction SELECT pertinente dans le dialogue **Ajouter/Supprimer/Éditer Objets de base de données**.
- Dans le dialogue **Ajouter/Supprimer/Éditer les objets de base de données**, sélectionnez l'instruction SELECT pertinente et cliquez sur **Ajouter/Éditer l'instruction SELECT**.
- Dans le dialogue **Ajouter/Supprimer/Éditer les objets de base de données**, sélectionnez l'instruction SELECT pertinente et cliquez sur **Ajouter/Éditer l'instruction SELECT**.

Pour supprimer une instruction SELECT, suivez les étapes ci-dessous :

1. Cliquez avec la touche de droite sur le composant de la base de données et choisir **Ajouter/Supprimer/Éditer des objets de base de données**.
2. Cliquez avec la touche de droite sur l'instruction SELECT que vous souhaitez supprimer et sélectionnez **Supprimer instruction SELECT** depuis le menu contextuel.

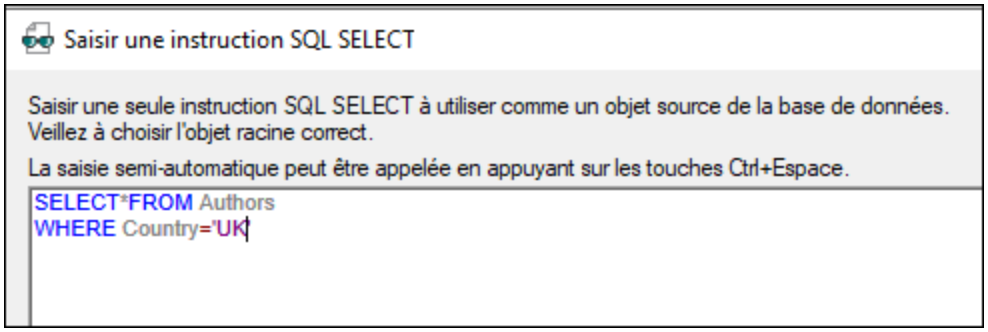
*Notes importantes*
Veuillez prendre note des points suivants :

- Toutes les expressions calculées dans l'instruction SELECT doivent avoir un nom de corrélation unique (par exemple **SELECT** `*, (Quantity*UnitPrice)` **AS** `Price`) pour être disponible en tant qu'items mappables.
- Si vous vous connectez à une base de données Oracle ou IBM DB2 utilisant JDBC, l'instruction, l'instruction `SELECT` ne doit pas avoir le point-virgule final.

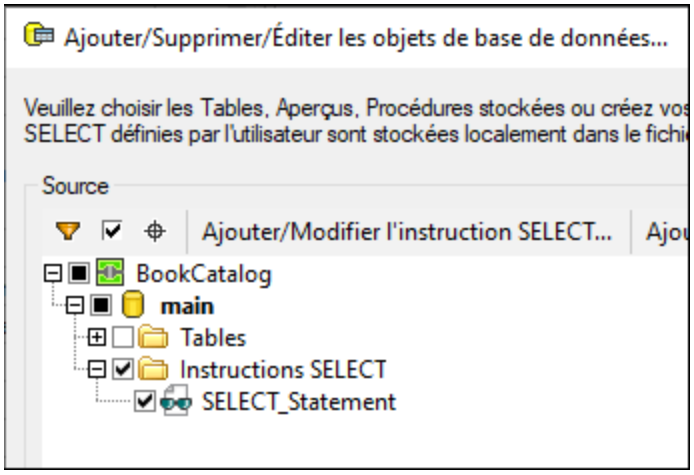
## Instructions SQL SELECT sans paramètres

L'exemple ci-dessous affiche comment travailler avec les instructions personnalisées SELECT sans paramètres. Dans le mappage ci-dessous, nous mappons des données de base de données vers un fichier texte. La base de données **`BookCatalog.sqlite`** a une table parent appelée `Authors` et une table enfant appelée `Books`. Toutefois, seule l'instruction SELECT avec une structure d'arborescence est affichée dans le compostant. La structure de l'arborescence dépend de la requête SQL que vous définissez dans le dialogue **Saisir une instruction SQL SELECT**. Puisque rien ne sera mappé des tables `Authors` et `Books`, ces tables sont absentes du composant.


*Instruction SELECT*
Pour le composant de base de données, nous avons ajouté l'instruction SQL suivante (*voir instructions dans Créer/Éditer/Supprimer une instruction SELECT*):




L'instruction SQL sélectionne toutes les tables de la table `Authors` et filtre les données de base de données pour inclure uniquement des auteurs de la GB. Dès que nous avons ajouté cette instruction dans le dialogue **Saisir une instruction SQL SELECT**, l'instruction devient disponible dans le dialogue **Ajouter/Supprimer/Éditer les objets de base de données** (*capture d'écran ci-dessous*). L'instruction devient également visible dans le composant de base de données (*voir le mappage ci-dessus*). Le nombre de lignes visibles de l'instruction SELECT peut être configuré dans le dialogue Options[1086] (l'option *Limiter l'affichage d'annotation*).

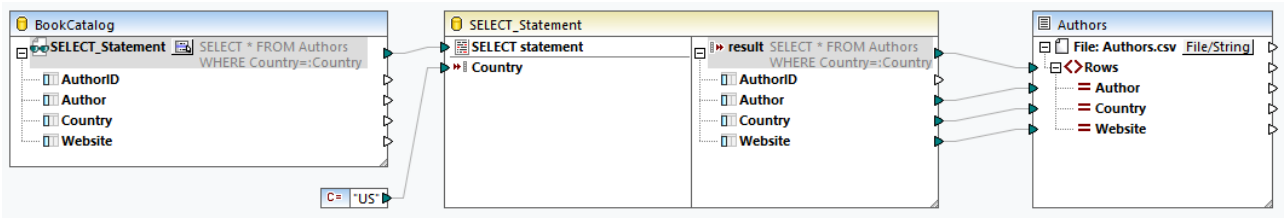



*Sortie*
La sortie affiche la liste de valeurs séparées par virgule qui incluent les auteurs uniquement de la GB (*liste de code ci-dessous*).

```
Auteur, Pays, Site web
Bram Stoker,UK,www.bramstoker.org
Charles Dickens,UK,www.charlesdickensinfo.com
Emily Brontë,UK,n/a
James Herbert,UK,www.james-herbert.co.uk
Neil Gaiman,UK,www.neilgaiman.com
Terry Pratchett,UK,www.terrypratchettbooks.com
Agatha Christie,UK,www.agathachristie.com
Roald Dahl,UK,www.roalddahlfans.com
David Walliams,UK,www.worldofdavidwalliams.com
Kenneth Grahame,UK,n/a
Philip Pullman,UK,www.philip-pullman.com
J.K. Rowling,UK,www.jkrowling.com
Ann Cleeves,UK,www.anncleeves.com
```

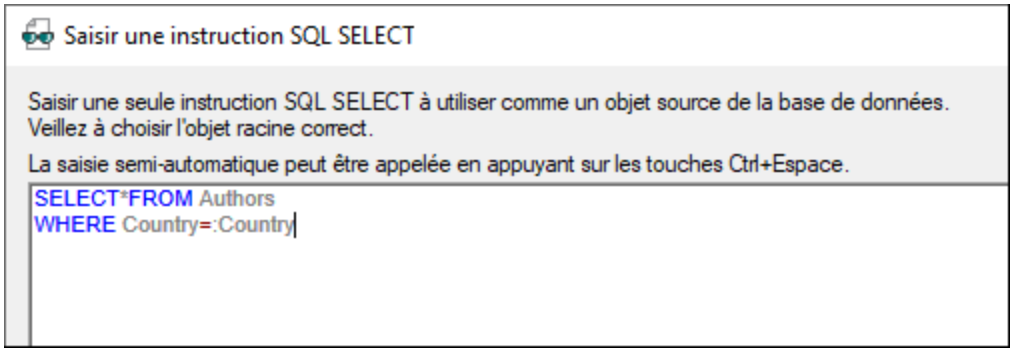
## Instructions SQL SELECT sans paramètres

Notre deuxième exemple illustre un mappage dans lequel le composant de base de données a une instruction personnalisée SELECT avec un paramètre (*voir la capture d'écran ci-dessous*).




*Instruction SELECT*
Pour le composant `BookCatalog`, nous avons saisi l'instruction SQL suivante :




L'instruction utilise le paramètre `Country`. Ce paramètre acceptera les valeurs d'une constante (*under the BookCatalog component*). Pour pouvoir mapper les données depuis l'instruction SELECT avec le paramètre, cliquez sur le bouton à côté du nœud `SELECT_Statement` dans le composant de base de données (*mappage ci-dessus*) et sélectionnez **Insérer appel avec Paramètres** depuis le menu contextuel. Ceci insère un composant Appel avec des paramètres (*composant central dans mappage ci-dessus*). Le composant Appel consiste en deux parties : La partie gauche accepte un paramètre d'entrée (dans notre cas, `Country`), et la partie droite réplique l'instruction SELECT avec la structure d'arborescence depuis le composant de base de données. Les données filtrées sont ensuite mappées dans le fichier texte `Authors`.

*Sortie*
La sortie affiche maintenant des auteurs uniquement des États-Unis d'Amérique (*liste de code ci-dessous*).

```
Auteur, Pays, Site web
Stephen King,US,www.stephenking.com
Frank Herbert,US,n/a
Isaac Asimov,US,www.asimovonline.com
Blake Crouch,US,www.blakecrouch.com
Ray Bradbury,US,www.raybradbury.com
Joe Hill,US,www.joehillfiction.com
Josh Malerman,US,www.joshmalerman.com
George R. R. Martin,US,www.georgerrmartin.com
A. J. Finn,US,n/a
Dan Brown,US,www.danbrown.com
Dean Koontz,US,www.deankoontz.com
```

## Exemples de fichiers

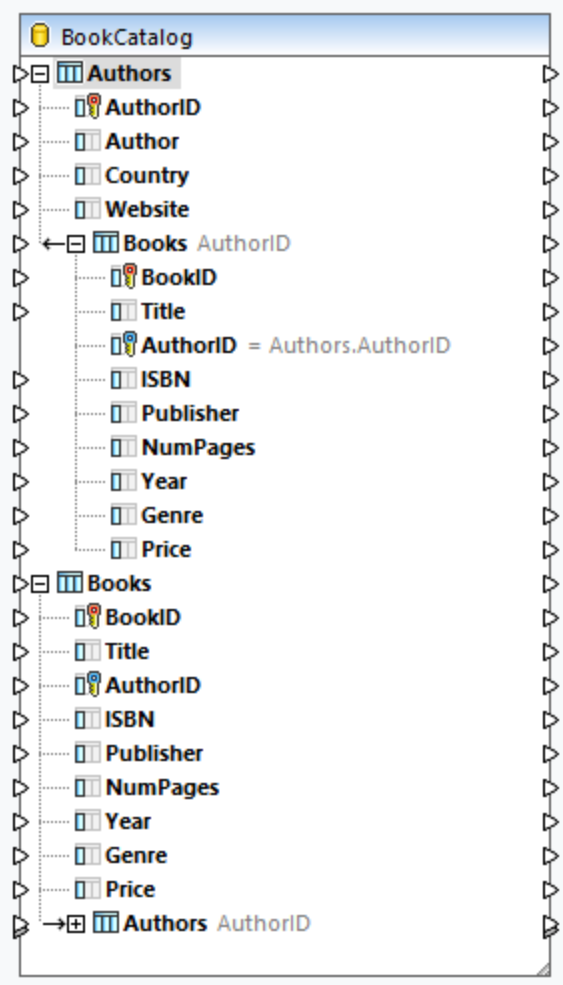
Pour plus d'information sur les mappages qui utilisent des instructions personnalisées SQL SELECT comme entrée, voir les exemples suivants dans le dossier **`MapForceExamples`** :

- **`DB_EmployeeListByTitle.mfd`**
- **`DB_MostExpensiveArticle.mfd`**
- **`DB_ManagerList_AllOffices.mfd`**
- **`DB_ManagerList_SelectedDepartment.mfd`**
- **`DB_ManagerList_SelectedOffice.mfd`**

### 4.2.2.3 Relations de base de données

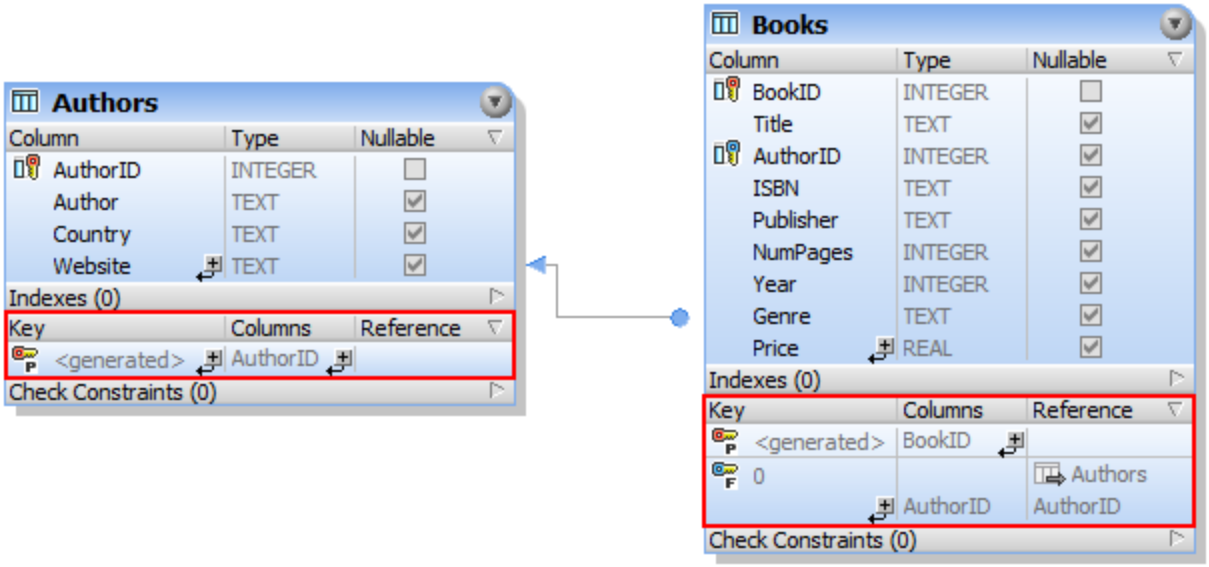
Quand vous ajoutez une base de données[251] comme composant de source à votre mappage, chaque table apparaît comme table racine (*capture d'écran ci-dessous*). Quand vous cliquez sur l'icône plus de la table racine, vous pouvez voir toutes les tables connexes en dessous de la table racine. Le composant de table ci-dessous affiche deux types de flèches qui signifient la chose suivante :

- La flèche pointant vers la gauche ( ← ) indique que la table `Books` est une table enfant de la table `Authors`.
- La flèche pointant vers la droite ( → ) affiche que la table `Authors` est le parent de la table `Books`.




*Structure de BookCatalog.sqlite*

Dépendant de vos besoins professionnels, vous pouvez utiliser différents scénarios de mappage. Les sous-sections ci-dessous illustrent quelques scénarios possibles. Tous les scénarios décrits ci-dessous affichent une base de données hiérarchique appelée **`BookCatalog.sqlite`**. La base de données contient deux tables (`Authors` et `Books`) qui ont une relation de clé étrangère. La capture d'écran ci-dessous affiche que la table `Books` a une clé étrangère appelée `AuthorID` qui référence la clé primaire dans la table `Authors`.




*Exemple de données de BookCatalog.sqlite*
Les extraits des tables `Author` et `Books` sont comme suit :

| | AuthorID | Author | Country | Website |
|---|---|---|---|---|
| 1 | 1 | Stephen King | US | www.stephenking.com |
| 2 | 2 | Ragnar Jonasson | Iceland | www.ragnarjonasson.com |
| 3 | 3 | Bram Stoker | UK | www.bramstoker.org |
| 4 | 4 | Charles Dickens | UK | www.charlesdickensinfo.com |
| 5 | 5 | Fyodor Dostoyevsky | Russia | n/a |
| 6 | 6 | Emily Brontë | UK | n/a |
| 7 | 7 | Frank Herbert | US | n/a |

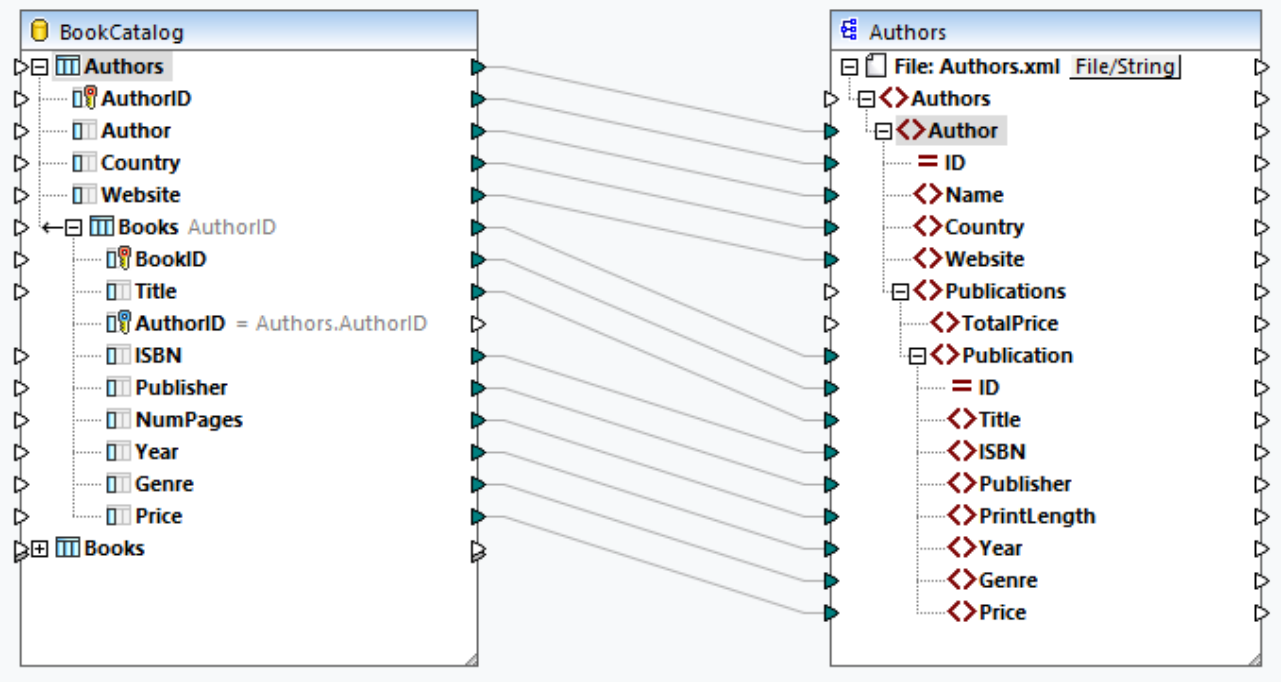
*Table Authors*

| | BookID | Title | AuthorID | ISBN | Publisher | NumPages | Year | Genre | Price |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | Misery | 1 | 1501143107 | Scribner | 368 | 2016 | Horror | 11.99 |
| 2 | 2 | Nightblind | 2 | 9781910633113 | Orenda Books | 231 | 2016 | Crime & Mystery | 9.99 |
| 3 | 3 | Blackout | 2 | 1910633461 | Orenda Books | 276 | 2016 | Crime & Mystery | 8.49 |
| 4 | 4 | Outsider | 1 | 1501180983 | Scribner | 576 | 2018 | Horror | 12.79 |
| 5 | 5 | Dracula | 3 | 9781435142817 | Barnes & Noble | 512 | 2013 | Classics | 13.69 |
| 6 | 6 | The Mystery of Edwin Drood | 4 | 9781400043286 | Everyman's Library | 336 | 2004 | Classics | 19.79 |
| 7 | 7 | Crime and Punishment | 5 | 0679420290 | Everyman's Library | 608 | 1993 | Classics | 14.99 |
| 8 | 8 | Wuthering Heights | 6 | 979-8469527794 | Independently published | 200 | 2021 | Classics | 22.99 |
| 9 | 9 | Dune (Dune Chronicles, Book 1) | 7 | 0441013597 | Penguin Publishing Group | 704 | 2005 | Sci-Fi | 14.99 |

*Table Books*

## Scénario 1 : Préservez la hiérarchie

Dans notre premier scénario, nous mappons les données de **BookCatalog.sqlite** vers **Authors.xsd** (*voir la capture d'écran ci-dessous*). Dans ce mappage, `Authors` est la table racine. Notre objectif est de préserver la relation hiérarchique et obtenir tous les auteurs avec leurs livres correspondants dans la sortie.



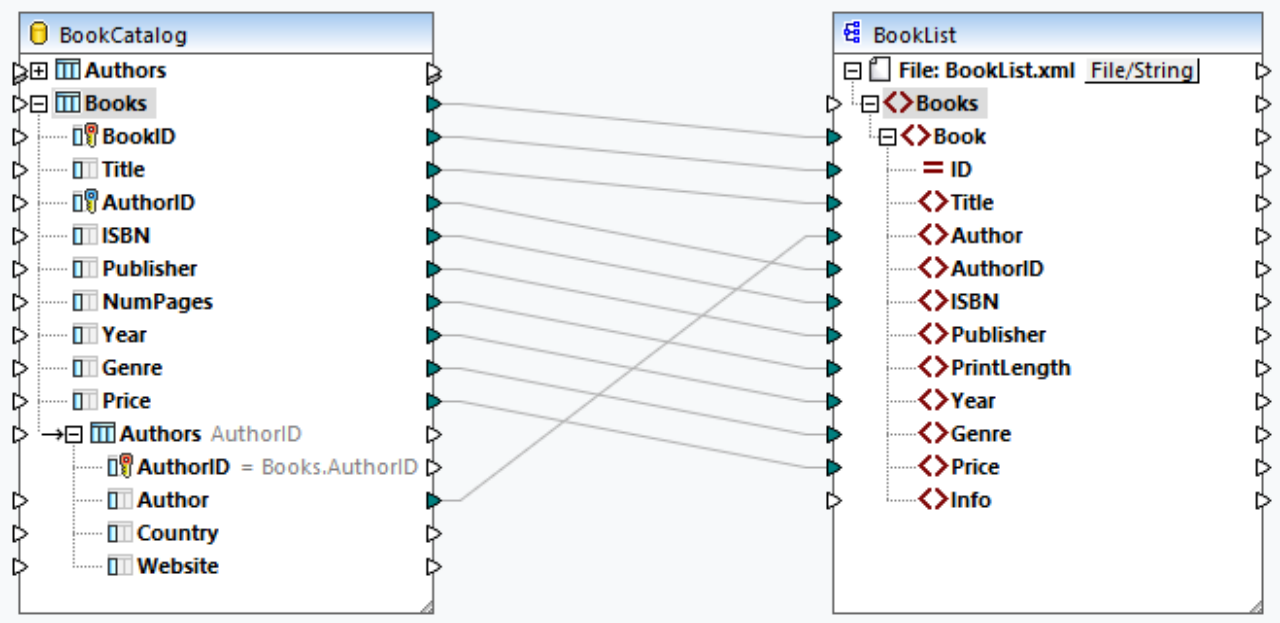

La liste de codes ci-dessous affiche un extrait de la sortie :

```
<Authors>
	<Author ID="23">
		<Name>Fredrik Backman</Name>
		<Country>Sweden</Country>
		<Website>www.fredrikbackmanbooks.com</Website>
		<Publications>
			<Publication ID="26">
				<Title>Anxious People</Title>
				<ISBN>978-1-4059-3025-3</ISBN>
				<Publisher>Penguin Books Ltd</Publisher>
				<PrintLength>416</PrintLength>
				<Year>2021</Year>
				<Genre>Humor</Genre>
				<Price>9.99</Price>
			</Publication>
			<Publication ID="27">
				<Title>A Man Called Ove</Title>
				<ISBN>9781444775815</ISBN>
				<Publisher>Sceptre</Publisher>
				<PrintLength>320</PrintLength>
				<Year>2015</Year>
				<Genre>Humor</Genre>
```

```
                        <Price>11.46</Price>
                    </Publication>
                </ Publications>
            </Author>
        </ Authors>
```

## Scénario 2 : Basculer les tables

Dans le deuxième scénario, notre objectif est d'obtenir une liste de livres et leurs détails dans le fichier sortie. Pour obtenir l'objectif, nous allons utiliser `Books` comme table de racine. Les relations de table resteront intacts. Le design de mappage a l'air de ceci :



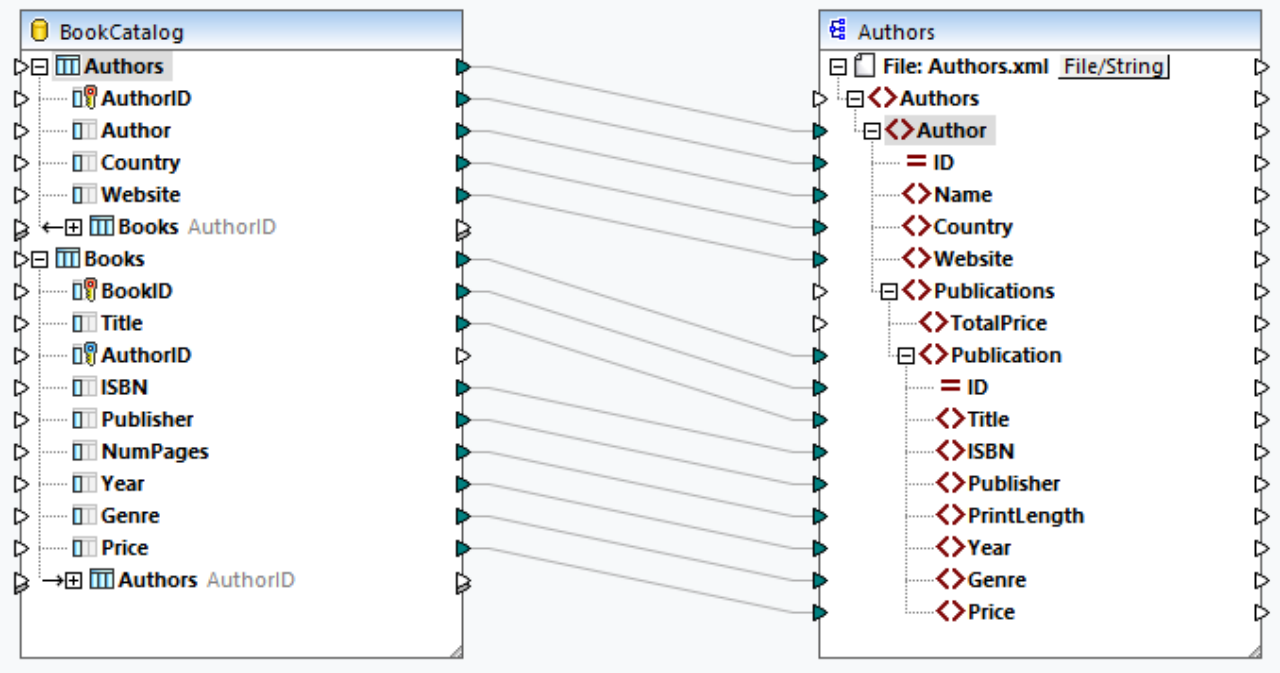

La liste de codes ci-dessous affiche un extrait de la sortie :

```
<Books>
      <Book ID="3">
            <Title>Blackout</Title>
            <Author>Ragnar Jonasson</Author>
            <AuthorID>2</AuthorID>
            <ISBN>1910633461</ISBN>
            <Publisher>Orenda Books</Publisher>
            <PrintLength>276</PrintLength>
            <Year>2016</Year>
            <Genre>Crime &amp; Mystery</Genre>
            <Price>8.49</Price>
      </Book>
      <Book ID="4">
            <Title>Outsider</Title>
            <Author>Stephen King</Author>
            <AuthorID>1</AuthorID>
            <ISBN>1501180983</ISBN>
            <Publisher>Scribner</Publisher>
            <PrintLength>576</PrintLength>
```

```
            <Year>2018</Year>
            <Genre>Horror</Genre>
            <Price>12.79</Price>
      </Book>
</Books>
```

## Scénario 3 : Mapper des données BD de différentes tables racine

Dans le troisième scénario, nous mappons des données de chaque table racine du composant de la base de données vers **`Authors.xsd`** (*voir la cpature d'écran ci-dessous*). Les tables connexes seront ignorées.




En conséquence, chaque livre individuel de la base de données sera recensé sous chaque auteur (*liste de code ci-dessous*).

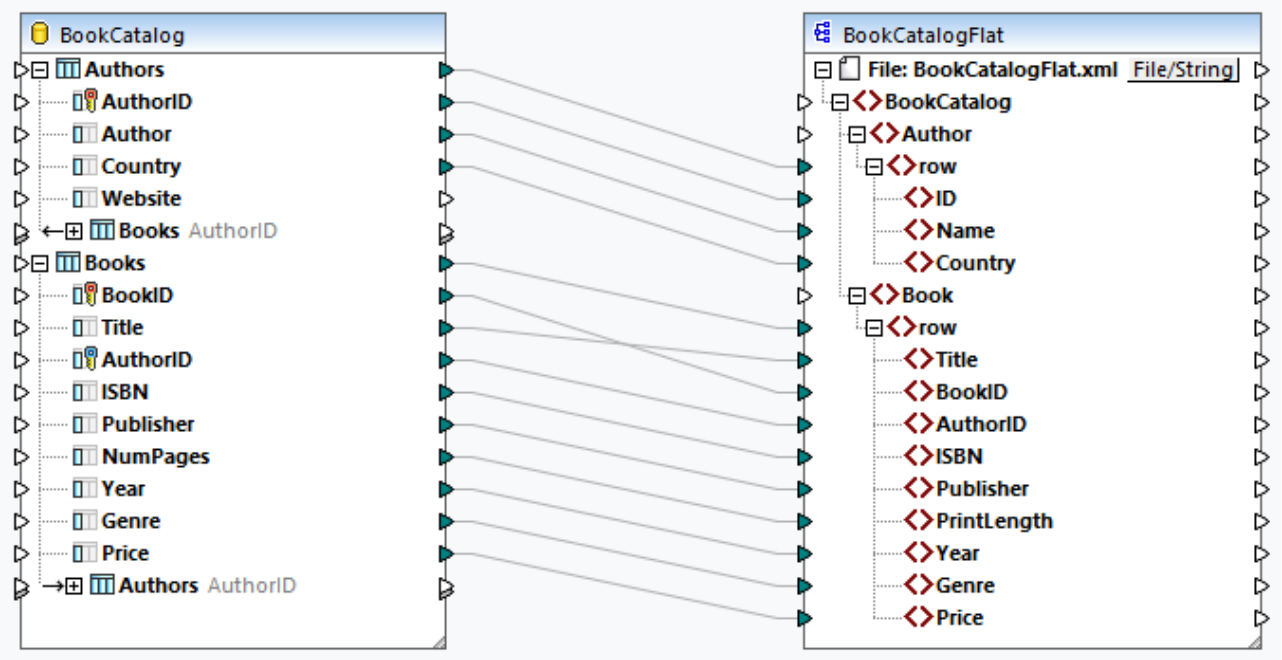
```
<Author ID="19">
      <Name>Sebastian Fitzek</Name>
      <Country>Germany</Country>
      <Website>www.sebastianfitzek.com</Website>
      <Publications>
            <Publication ID="1">
                  <Title>Misery</Title>
                  <ISBN>1501143107</ISBN>
                  <Publisher>Scribner</Publisher>
                  <PrintLength>368</PrintLength>
                  <Year>2016</Year>
                  <Genre>Horror</Genre>
                  <Price>11.99</Price>
            </Publication>
            <Publication ID="2">
                  <Title>Nightblind</Title>
                  <ISBN>9781910633113</ISBN>
```

```
                    <Publisher>Orenda Books</Publisher>
                    <PrintLength>231</PrintLength>
                    <Year>2016</Year>
                    <Genre>Crime &amp; Mystery</Genre>
                    <Price>9.99</Price>
                </Publication>
                <Publication ID="3">...</Publication>
                <Publication ID="4">...</Publication>
                <Publication ID="5">...</Publication>
                <Publication ID="6">...</Publication>
                <Publication ID="7">...</Publication>
                <Publication ID="8">...</Publication>
            </ Publications>
        </Author>
```

## Scénario 4 : Mapper des données BD à la structure SQL/XML

Dans le quatrième scénario, notre objectif est de mapper les données de base de données vers une structure de schéma (SQL/XML Standard). Le modèle de schéma est basé sur la spécification ISO-ANSI SQL/XML INCITS/ISO/IEC 9075-14-2008. La spécification SQL/XML définit comment mapper les bases de données en XML. Les relations sont définies dans les schémas en utilisant des contraintes d'identité ; il n'y a pas de références aux éléments. Ainsi, le schéma présente une structure plate qui ressemble à un aperçu arborescent de la base de données. La spécification peut être achetée dans la boutique ANSI. Pour plus d'information, voir www.iso.org.

Le mappage ci-dessous affiche que les données de la base de données sont mappées de tables racine différentes vers une structure SQL/XML plate. Les tables connexes seront ignorées. Il est également possible de mapper les données de la base de données depuias les tables connexes. Toutefois, si'il existe des enregistrements `Book` qui n'appartiennent pas à un `Author`, ces enregistrements `Book` ne seront pas mappés à la cible.



En conséquence, nous recevrons une liste de lignes `Author`et une liste séparée de lignes `Book` (*capture d'écran ci-dessous*).

```
<Author>
      <row>
            <ID>1</ID>
            <Name>Stephen King</Name>
            <Country> US</Country>
      </row>
      <row>
            <ID>2</ID>
            <Name>Ragnar Jonasson</Name>
            <Country>Iceland</Country>
      </row>
      <row>...</row>
      <row>...</row>
</Author>
<Book>
      <row>
            <Title>Misery</Title>
            <BookID>1</BookID>
            <AuthorID>1</AuthorID>
            <ISBN>1501143107</ISBN>
            <Publisher>Scribner</Publisher>
            <PrintLength>368</PrintLength>
            <Year>2016</Year>
            <Genre>Horror</Genre>
            <Price>11.99</Price>
      </row>
      <row>
            <Title>Nightblind</Title>
            <BookID>2</BookID>
            <AuthorID>2</AuthorID>
            <ISBN>9781910633113</ISBN>
            <Publisher>Orenda Books</Publisher>
            <PrintLength>231</PrintLength>
            <Year>2016</Year>
            <Genre>Crime &amp;amp; Mystery</Genre>
            <Price>9.99</Price>
      </row>
      <row>...</row>
      <row>...</row>
</Book>
```

Pour plus d'informations sur ce scénario, voir le mappage suivant :
**`MapForceExamples\DB_Altova_SQLXML.mfd`**.

## 4.2.2.4 Relations locales

Quand les tables de base de données n'ont pas de relations entre elles, vous pouvez créer des relations de clés primaire et étrangères entre les colonnes des différentes tables directement dans MapForce (par ex., relations locales). Toute colonne de base de données peut être utilisée en tant que clés primaire ou étrangère.

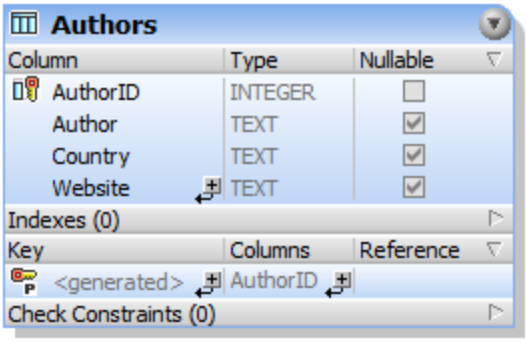
De plus, de nouvelles relations peuvent être créées, en plus de celles existant dans la base de données. Les relations définies localement sont enregistrées ensemble avec le mappage.

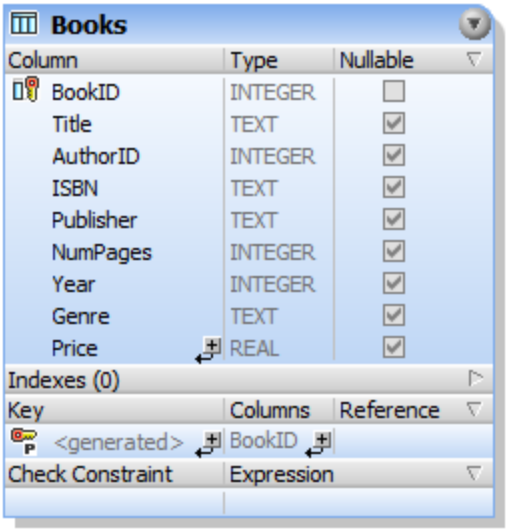
La table suivante recense tous les champs possibles entre lesquels vous pouvez définir des relations locales. Des relations mixtes sont possibles (par exemple, mapper la sortie d'une procédure stockée dans une colonne de base de données). Les champs participant à la relation doivent avoir les mêmes types de données ou des txpes de données compatibles.

| Clé primaire/unique | Clé étrangère |
|---|---|
| • La colonne d'une table de base de données ou d'un mode<br>• Paramètre de sortie ou valeur de retour d'une procédure stockée, voir aussi Procédures stockées 321<br>• La colonne d'un recordset retournée par une procédure stockée* * Applicable si la procédure stockée est appelée soit en tant que source de données (sans paramètres) ou en tant que fonction (avec paramètres d'entrée et de sortie). Pour que le recordset devienne disponible pour la sélection, vous devez exécuter la procédure stockée une fois, pour extraire le recordset.<br>• La colonne d'une instruction SELECT définie par l'utilisateur, voir aussi Instructions SELECT SQL en tant que tables virtuelles 264. | • La colonne d'une table de base de données ou d'un mode<br>• Paramètre d'entrée d'une procédure stockée<br>• Paramètre d'entrée d'une instruction SELECT définie par l'utilisateur |

## Exemple

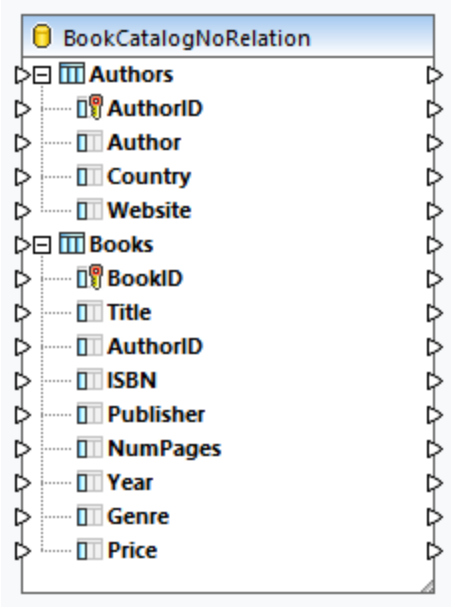
La base de données **`BookCatalogNoRelation.sqlite`** a deux tables : `Authors` et `Books` (*voir la capture d'écran ci-dessous*). À cette étape, aucune relation de clé étrangère n'existe entre les tables.




*Relations BD sans relations*

Lorsque nous insérons la base de données dans le mappage, les composants de base de données sont comme suit :

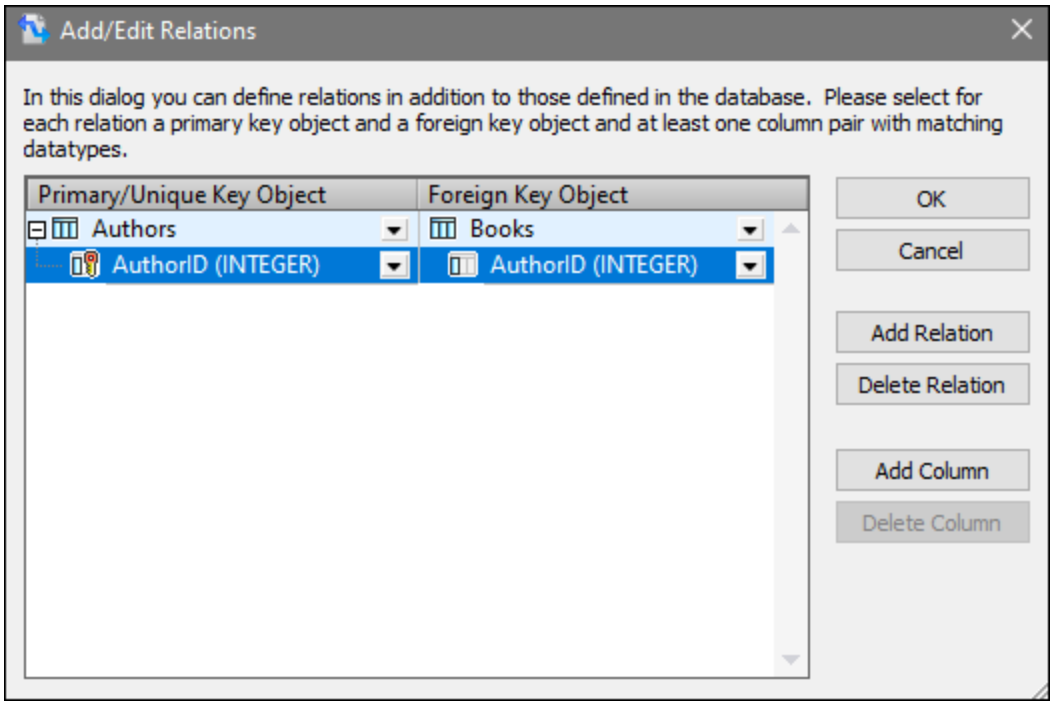



*Définition de relation locale*

Dans cet exemple, notre objectif est de référencer la table `Authors` dans la table `Books` table. Suivez les instructions ci-dessous :

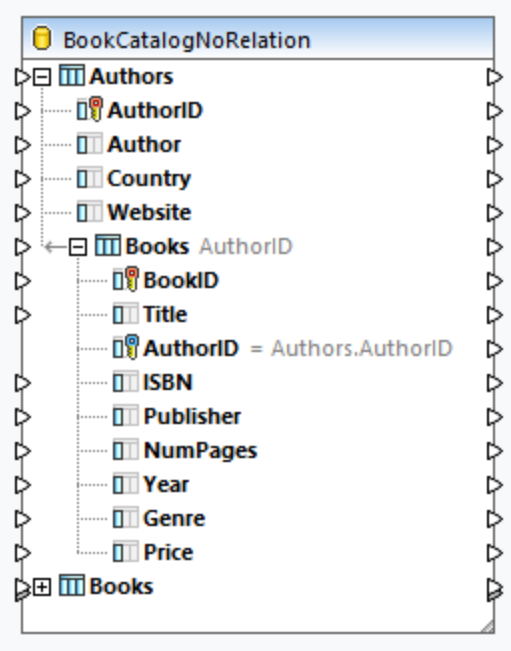
1. Cliquez avec la touche de droite à l'intérieur du composant et sélectionnez **Ajouter/Supprimer/ Éditer Filtre** depuis le menu contextuel.
2. Cliquer sur la touche **Ajouter/Éditer Relations** dans le dialogue **Ajouter/Supprimer/Éditer Objets de base de données**.

3. Cliquez **Ajouter relation** dans le dialogue **Ajouter/Éditer Relations** (*capture d'écran ci-dessous*)
4. Cliquez sur *[select object]* dans la colonne *Objet de Clé* et sélectionnez `Authors`. Puis, sélectionnez `AuthorID` dans la liste déroulante *[select column]*.
5. Cliquez sur *[select object]* dans la colonne *Objet de clé étrangère* et sélectionnez `Books`. Puis, sélectionnez `AuthorID` dans la liste déroulante *[select column]*.
6. Cliquez sur **OK** pour compléter la définition de relation locale.




*Composants BD avec relations*

Dès que vous avez terminé de définir les relations locales, le composant Base de données devient disponible dans la zone de mappage (*capture d'écran ci-dessous*). Le composant affiche les deux structures de base de données possibles. Dans chacune de ces structures, la table racine est différente. Par exemple, dans la structure élargie ci-dessous, `Authors` est la table racine. Dépendant de vos besoins, vous pouvez mapper les données des structures disponibles dans le composant. Vous pouvez aussi mixer et faire correspondre les tables des différentes structures dans le composant. Pour plus d'informations concernant ces scénarios, voir des relations de base de données 268.




## 4.2.2.5 Fonctions liées à la base de données

Quand vous travaillez avec des bases de données, vous allez éventuellement devoir utiliser diverses fonctions pour gérer des valeurs null, générer des valeurs séquentielles et uniques et remplacer des caractères spéciaux. Pour plus d'information, veuillez voir les sous-sections ci-dessous.

### Gérer des valeurs nulles

MapForce fournit les fonctions suivantes pour gérer les valeurs null :

- Pour contrôler pendant l'exécution d'un mappage si un champ de base de données est null, utiliser les fonctions **is-null**[643] et **is-not-null**[643]. Pour voir si une table a zéro champs, interrogez-la en utilisant le navigateur de base de données dans MapForce (voir Volet Requête BD[300]).
- Pour régler un champ de base de données sur zéro, utiliser la fonction **set-null**[644].
- Pour remplacer les valeurs de base de données nulles par un string, utiliser la fonction **substitute-null**[644].

Pour des informations sur la gestion des valeurs NULL dans la base de données, voir *Null Equal*[283]. Voir également *Valeurs Null dans les Composants de base de données*[131].

### Générer des valeurs séquentielles et uniques

Lorsque vous mettez à jour des enregistrements de base de données, vous devrez créer des valeurs séquentielles ou uniques on-the-fly pour les champs de base de données qui ne reçoivent pas d'entrée de depuis la source. Dans de tels cas, vous pouvez utiliser les fonctions suivantes :

- La fonction **auto-number**[576] peut être utilisée pour générer des valeurs de clé primaire.
- La fonction **create-guid**[671] crée un identifiant globalement unique (en tant que string encodé hex) pour un champ spécifique.

Veuillez noter que les valeurs pour les champs de base de données peuvent aussi être écrites en utilisant des valeurs générées par la base de données. Cette option est disponible dans le dialogue Actions de table de base de données[280] et est particulièrement utile lors de la génération de clés primaires.

### Remplacer des caractères spéciaux

Lorsque vous mettez à jour vos données de base de données, vous allez éventuellement devoir remplacer les caractères spéciaux (par ex., caractères retour de chariot / saut de ligne (CR/LF)). Pour ce faire, vous pouvez utiliser une des approches suivantes :

1. Vous pouvez définir une fonction de nœud pour un champ de base de données spécifique (ou plusieurs champs) que vous devez traiter. La fonction de nœud recevra la valeur du champ de base de données en tant qu'entrée, y appliquera des processus puis retournera le résultat au mappage. Pour plus d'informations concernant cette approche, voir Défauts et fonctions de nœud[476].
2. Traiter les valeurs de base de données à l'aide des fonctions intégrées de MapForce. Par exemple, pour identifier des caractères spécifiques, y compris les caractères de commande, vous pouvez utiliser la fonction **char-from-code**[626]. Pour remplacer les valeurs, utilisez la fonction **remplacer**[694].

## 4.2.3 Actions de Table de base de données

Quand vous utilisez une base de données en tant que composant cible, vous définissez les options de gestion de la transaction. Par exemple, vous pouvez insérer tous les enregistrements depuis le fichier source dans votre base de données. Vous pouvez décider quand vous souhaitez mettre à jour, supprimer et ignorer des enregistrements. Cette section fournit un aperçu de toutes les actions disponibles et affiche quelques-uns des scénarios possibles utilisant les actions de table.

### Instructions SQL dans la sortie

Quand vous mappez des données à la base de données et visualisez le résultat du mappage dans le volet de sortie, vous verrez un script SQL. Le script affiche des instructions pseudo-SQL à des fins d'information uniquement. Vous ne devez pas appliquer ce script SQL manuellement à la base de données, utilisant les outils SQL autres que les moteurs d'exécution suivants : MapForce, MapForce Server[864] (tous deux autonomes ou sous gestion de FlowForce Server[867]), ou l'environnement d'exécution du code généré pour C++, C# ou Java. Le script dans le volet de sortie peut contenir des valeurs qui ne sont pas comprises par des éditeurs SQL externes.

Si vous voulez appliquer les changements à la base de données directement depuis MapForce, ouvrez le volet de sortie et cliquez sur la commande **Exécuter Script SQL/NoSQL** dans la barre d'outils ou dans le menu **Sortie**. Cette action modifiera la base de données avec effet immédiat.

Lorsque le mappage est exécuté avec MapForce Server (aussi la version autonome que sous gestion FlowForce Server), les modifications de la base de données sont réalisées avec effet immédiatement. La même chose se passe dans le code généré : Les changements de la base de données sont réalisés quand vous compilez et exécutez le code (par ex., en cliquant sur la commande **Exécuter** dans Visual Studio).

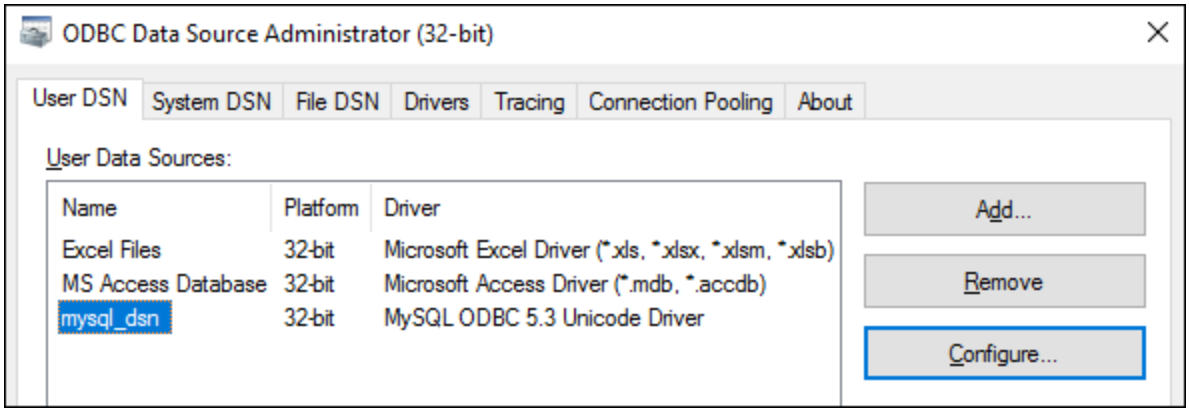
*Note importante*
Votre installation de MapForce inclut plusieurs exemples de base de données qui sont disponibles dans le dossier **`MapForceExamples`**. Il n'est pas recommandé de modifier les bases de données dans ce dossier, étant donné que ceci peut rendre plusieurs exemples inutilisables. Une manière simple d'éviter d'écraser des données originelles est de sauvegarder en back-up tout le dossier **`MapForceExamples`** avant de mettre à jour tout fichier dans celui-ci..

### Note sur MySQL/MariaDB ODBC

Si la base de données cible est MySQL ou MariaDB through ODBC, l'option *Retourner des lignes correspondantes au lieu des lignes affectées* doit être activée dans l'onglet Curseur/Résultats de MySQL ODBC Connector. En alternative, si vous saisissez le string de connexion manuellement dans l'assistant Connexion de base de données, ajoutez `Option=2` au string de connexion (par ex., `Dsn=mydsn;Option=2;`).

Pour activer cette option depuis le Connecteur ODBC MySQL, suivez les étapes ci-dessous :

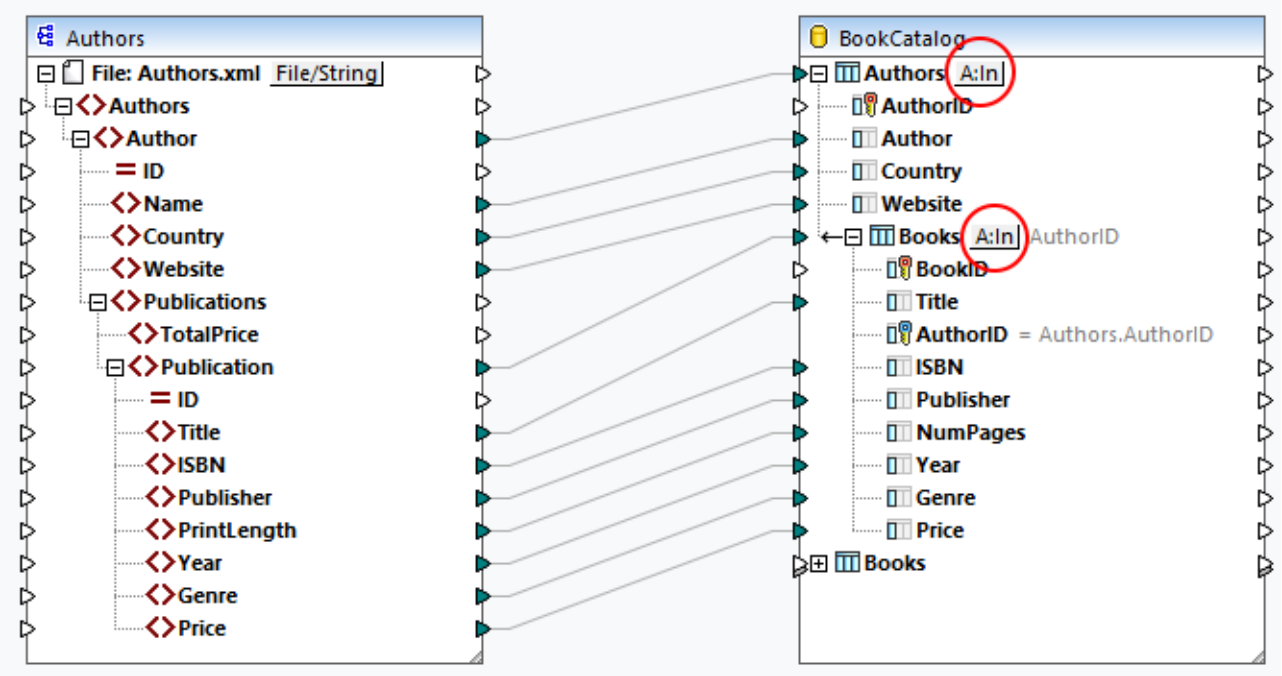
1. Cliquez sur la clé **Windows**, puis commencez à saisir *ODBC*.
2. Exécutez l'Administrateur de sources de données ODBC (32-bit soit 64-bit, selon la plateforme du Connecteur ODBC MySQL installé).
3. Cliquez sur Data Source Name (DSN) utilisé par le mappage MapForce, puis cliquez sur **Configurer** (*voir ci-dessous*).




4. Cliquez sur les **Détails >>** pour rendre disponible les options avancées.
5. Cliquez sur l'onglet *Curseurs/Résultats* et sélectionnez la case à cocher *Retourner lignes correspondantes au lieu des lignes touchées*.

## 4.2.3.1 Actions de table BD : Paramètres

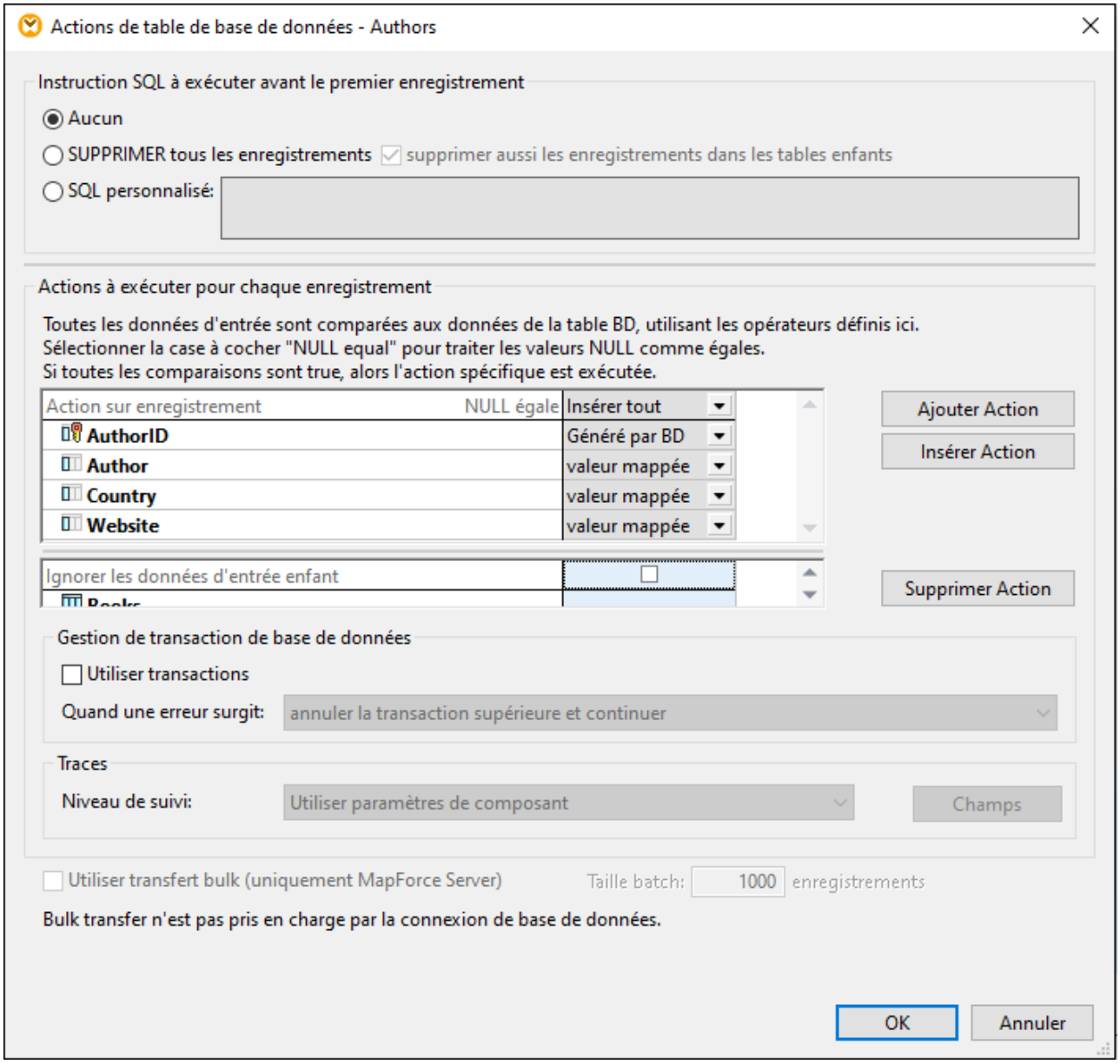
Quand vous mappez des données vers une table de données, cette table aura le bouton **Actions de base de données** à côté de celle-ci. La capture d'écran ci-dessous affiche que nous mappons les données XML de deux tables de base de données, et chaque table a son propre bouton **Actions de base de données** (*encerlcé en rouge ci-dessous*).




Cliquez sur le bouton **Actions de base de données** ouvre le dialogue **Actions de table de base de données** (*voir ci-dessous*), dans lequel vous pouvez configurer divers paramètres, actions et options. Le dialogue **Actions de Table de base de données** comprend cinq parties :

1. Actions à être réalisées avant toute action définie pour chaque enregistrement
2. Actions à définir pour chaque enregistrement
3. Paramètres de transaction de la base de données
4. Suivi et paramètres de journalisation d'erreur
5. Paramètres de transfert bulk

Pour en savoir plus sur chaque partie du dialogue, voir les sous-sections ci-dessous.




## Instruction SQL pour une exécution avant le premier enregistrement

Dans cette section, vous pouvez définir des instructions SQL qui seront exécutées *avant* toute action définie dans la section *Actions à exécuter pour chaque enregistrement*. Les options suivantes sont disponibles :

- L'option *Aucun* signifie qu'aucune action n'est réalisée. Il s'agit là du réglage par défaut.
- L'option *DELETE tous les enregistrements* supprimera tous les enregistrements de la table sélectionnée. De plus, vous pouvez choisir la suppression de tous les enregistrements dans les tables enfant (la case à cocher *aussi supprimer tous les enregistrements dans toutes les tables enfant*).
- Le paramètre *SQL Personnalisé* vous permet d'écrire une instruction SQL personnalisée qui affectera toute la table. Par exemple, vous pouvez ajouter une instruction qui fournira une information de suivi sur

le mappage. Veuillez noter que la prise en charge pour plusieurs instructions SQL dans une requête dépend de la base de données, de la méthode de connexion et du pilote utilisé.

Pour plus d'information à propos de ces actions, voir Actions de table BD[288]. Scénarios[288].

## Actions à exécuter pour chaque enregistrement

La section *Actions à exécuter pour chaque enregistrement* vous permet de définir les actions de base de données qui seront réalisées dans votre base de données. Pour gérer les actions de table, cliquer sur les touches **Apposer Action**, **Insérer Action**, ou **Supprimer Action**. De multiples actions peuvent être définies, le cas échéant. Toute action de table définie après les actions *Insérer tout* ou *Insérer Rest* ne sera jamais exécutée car aucune condition de colonne n'existe pour les actions d'insertion. Si vous avez ajouté une action de table après l'action *Insérer Tout* ou *Insérer Rest*, un dialogue vous informera que l'action de table suivante sera supprimée.

Pour chaque action, toutes les données d'entrée sont comparées aux données de la base de données. Si toutes les comparaisons sont true, une action spécifique sera réalisée. Les actions de table définies sont traitées de gauche à droite. Par exemple, si vous avez défini une condition *Update If*, puis une condition *Insérer Rest*, l'action *Update If* sera traitée d'abord. Si la condition *Update If* n'est pas satisfaite, alors l'action *Insert Rest* sera réalisée. Si aucune des conditions n'est satisfaite, aucune action n'aura lieu.

*Mapped value/DB-generated/max() + 1*

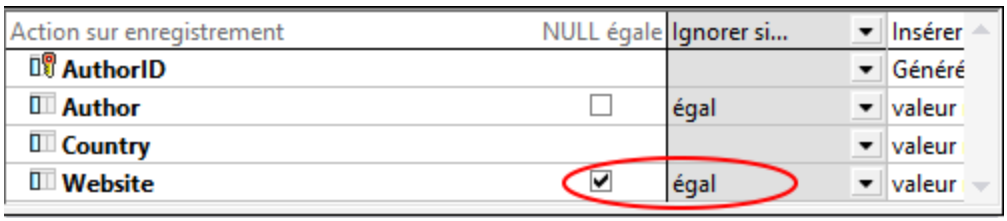
Quand vous définissez l'action *Insert All* ou *Insert Rest*, vous pouvez décider comment les valeurs doivent être générées. Les options suivantes sont disponibles : *mapped value*, *DB-generated*, et *max() + 1*. L'option *valeur mappée* signifie que les données source seront mappées vers le champ de base de données directement. L'option *valeur mappée* est le paramètre standard pour les champs de base de données.

Quand « auto-increment » est défini pour la/les clé/s primaire/s dans votre base de données, vous pouvez choisir entre une valeur mappée et la valeur générée par BD. Quand il n'y a pas d'auto-increment, vous pouvez choisir entre une valeur mappée et l'option *max() + 1*. L'option *max() + 1* génère les valeurs numériques basées sur les valeurs existantes dans la base de données. Par exemple, si la table comporte trois enregistrements, avec des clés primaires 1, 2 et 3, alors la clé primaire d'un nouvel enregistrement sera 4.

*NULL equal*

Lorsque vous sélectionnez la case à cocher *NULL equal* à côté de l'enregistrement (*capture d'écran ci-dessous*), les valeurs null dans l'enregistrement source et les valeurs null dans l'enregistrement cible seront considérées égales. Ne pas sélectionner cette case à cocher peut entrîner des résultats incorrects. Vous pouvez utiliser la case à cocher *NULL equal* si *toutes* les conditions suivantes sont true :

- Le champ pour lequel vous voulez permettre l'option *NULL equal* est nullable. Ceci signifie que ce champ a été configuré de telle manière qu'il peut accepter les valeurs NULL.
- Une ou plusieurs des actions suivantes a été configurée : *Ignore If*, *Update If*, ou *Delete If*.
- Une ou plusieurs actions de table (par ex., *Ignore If*) ont au moins une condition *equal* ou *equal (ignorer la casse)* (voir *Champ Author dans la capture d'écran ci-dessous*).



L'image ci-dessus affiche que deux actions ont été définies pour la table `Authors` : *Ignore If* et *Insert Rest*. La condition *Ignore If* comparera les valeurs `Author` et `Website` dans la source avec les valeurs `Author` et `Website` dans la base de données. Si ces valeurs sont les mêmes, ces enregistrements seront ignorés dans la base de données, et les enregistrements de la source qui n'ont pas de contreparties dans la base de données seront simplement insérés dans la base de données.

L'option *NULL equal* a été activée pour la colonne `Website`. Par exemple, un des enregistrements de `Author` a une nouvelle valeur dans le champ `Website` dans la source et la cible. Avec l'option *NULL equal* activée, cet enregistrement `Author` sera ignoré et ne sera pas inséré dans la base de données. Toutefois, si la case à cocher *NULL equal* n'est pas sélectionnée, l'enregistrement ne satisfera plus la condition *Ignore If* et sera inséré dans la base de données.

*Tables enfant*
Si une relation foreign-key existe dans votre base de données, vous serez à même de voir les noms de tables enfant dans le dialogue **Actions de table de base de données** (*Books dans le dialogue des Actions de Table de base de données ci-dessus*). Les options suivantes seront disponibles :

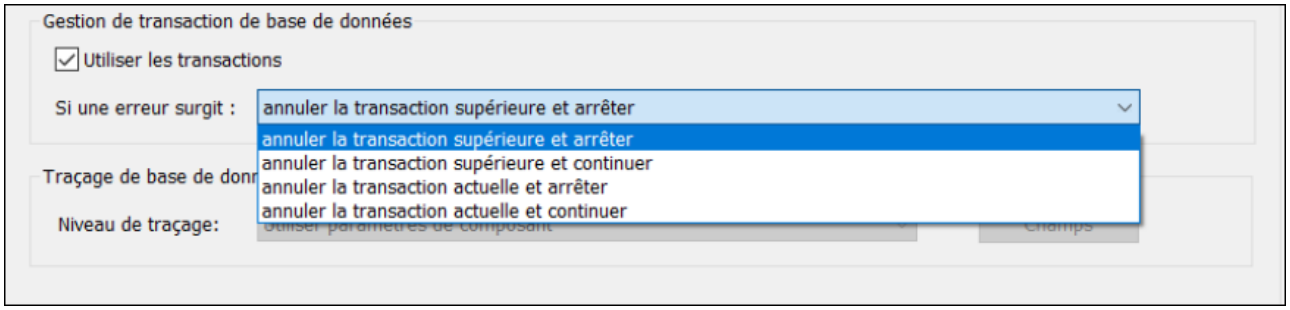
- *Supprimer des données dans les tables enfants :* Cette option peut être particulièrement utile quand vous configurez, par exemple, l'action *Update If* pour une table parent et l'action *Insert All* pour sa table enfant. La condition *Update If* mettra à jour uniquement les enregistrements parent qui existent dans la source et la base de données. La condition *Update If* vous préviendra d'avoir des enregistrements doubles dans la table parent.

  L'option *Supprimer données des tables enfant* supprimera uniquement ces enregistrements `Book` dont les enregistrements parent (`Authors`) satisfont la condition *Update If* condition (c'est-à-dire existent dans la source et la base de données). Supprimer des enregistrements enfant vous empêchera d'avoir des doublons ou des données orphelines dans la table enfant. L'action *Insert All* définie pour la table enfant insérera seuls ces enregistrements enfant source dont les enregistrements parent satisfont la condition *Update If* .

- *Ignorer les données enfant d'entrée :* Cette option est utile quand vous voulez mettre à jour uniquement une table parent et laisser les enregistrements enfant inchangés.

## Gestion de la transaction de base de données

La section *Database transaction handling* vous permet de restaurer une opération ou une séquence d'opérations (transaction) en cas d'erreur (par ex., valeurs NULL insérées dans les colonnes non-NULL). Pour permettre les transactions au niveau de l'action de table, sélectionnez la case à cocher *Utiliser transactions* dans le dialogue **Actions de Table de base de données** (*capture d'écran ci-dessous*).




Les options de gestion de la transation sont disponibles :

- *Restaurer transaction du haut et arrêter :* Puisque la gestion de la transaction peut être configurée aux différents niveaux de la hiérarchie de la base de données, la *transaction du haut* peut se référer à (i) la transaction au niveau du composant de base de données[256] si vous avez activé les transactions à ce niveau ou (ii) la transaction définie dans la table du niveau supérieur. Les changements effectués dans la base de données seront restaurés pour chaque niveau de la hiérarchie jusqu'au niveau supérieur, puis l'exécution s'arrêtera.
- *Annuler cette transaction et continuer :* Comme ci-dessus, mais le mappage continue d'être exécuté après la restauration (par ex., pour traiter un autre composant cible si un tel composant existe).
- *Restaurer transaction actuelle et arrêter :* Quand une erreur associée à la base de données apparaît, seuls les changements dans la transaction actuelle seront restaurés, et le traitement d'enregistrements ultérieurs s'arrêtera. Les changements rendus avant en dehors de la transaction actuelle seront validés.
- *Restaurer transaction actuelle et continuer :* Comme ci-dessus, mais le mappage continue à être exécuté après l'annulation.

Pour plus d'information sur les différents scénarios de gestion de la transaction, voir Transaction Rollback: Scénarios[293].

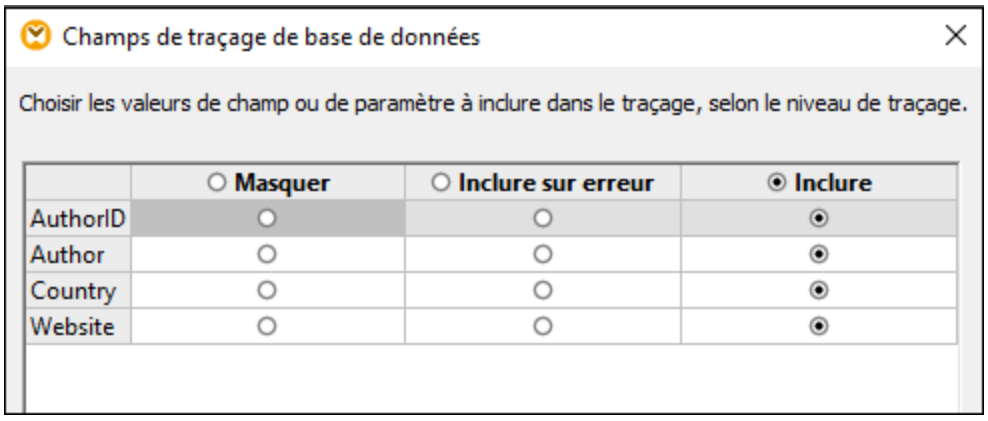
## Traçages

Lorsqu'un mappage écrit des données dans une base de données, vous pouvez activer le traçage et la journalisation d'erreur de la base de données. Le traçage est utile si vous souhaitez suivre tous les changements devenir dans la base de données lors de l'exécution du mappage. Les changements effectués dans la base de données sont journalisés dans un rapport de trace. S'il existe des erreurs pendant l'exécution, ces erreurs seront également journalisées.

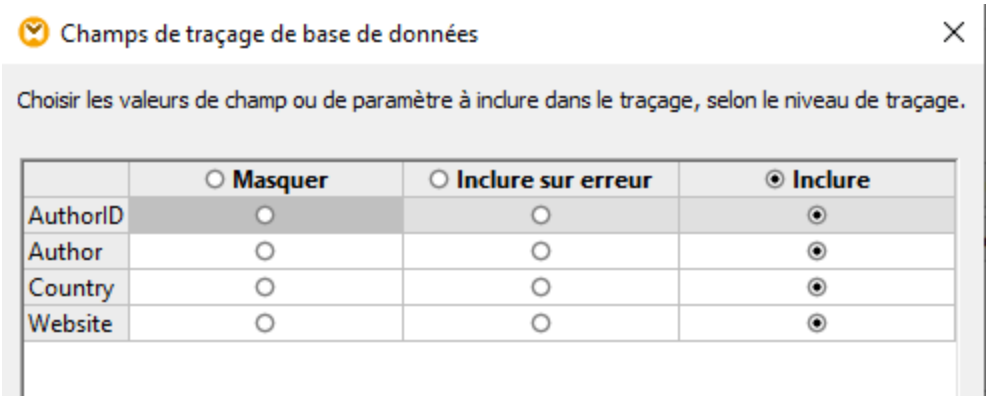
Pour plus d'information sur la structure du rapport de trace, voir Fichier de traçage[260]. Pour un exemple de traçage activé, voir le *Scénario 1* dans Restauration de transaction : Scénarios[298].

Pour activer le traçage au niveau de la table, suivez les étapes ci-dessous :

1. Assurez-vous que le niveau de traçage au niveau du composant de base de données[260] est défini soit comme *Toujours* ou *Erreur*.
2. Cliquez sur la touche de l'**Action de table** située à côté de la table dans laquelle vous souhaitez activer le traçage.
3. Sélectionnez un des niveaux de trace suivants dans la section *Traces* des **Actions de table de la base de données** (*capture d'écran ci-dessous*) : (i) l'option *Toujours désactivé* signifie qu'aucun traçage n'apparaîtra pour cette table ; (ii) l'option *Limiter aux erreurs* restreint le traçage uniquement aux événements d'erreur ; (iii) l'option *Utiliser des paramètres de composant* hérite les paramètres qui ont été définis au niveau du composant.




*Traçage au niveau du champ de base de données*
Lorsque vous activez le traçage au niveau du-composant-base de données et au niveau de la table, tous les champs (colonnes de base de données) sont inclus par défaut dans le rapport de traçage.(*capture d'écran ci-dessous*). Si vous voulez tracer le rapport pour inclure l'information de traçage sur les champs de base de données, cliquez sur le bouton **Champs** dans le dialogue **Actions de table de base de données**. Vous pouvez choisir de masquer des champs, les inclure dans tout cas ou les inclure uniquement dans le cas d'une erreur.




## Utiliser le Transfert Bulk

L'option *Utiliser le Transfert Bulk* signifie que de multiples instructions INSERT sont exécutées comme requête. Le recours à cette option accélère considérablement le processus Insérer, étant donné qu'il suffit d'exécuter une seule instruction au lieu de plusieurs. L'option *Utiliser le Transfert Bulk* est configurée dans MapForce, mais le transfert bulk actuel de données apparaît quand le mappage est exécuté par MapForce Server.

*Prérequis*
L'option transfert bulk est prise en charge lorsque les conditions suivantes sont vraies :

- Le langage de transformation de mappage est configuré sur Built-In.
- Le mappage est exécuté par MapForce Server[864] (autonome ou sous gestion FlowForce Server[867]).
- La licence de MapForce Server ne se limite pas à une exécution single-thread sur un appareil multi-core. C'est-à-dire que l'option *Limiter à l'exécution de thread unique* dans l'onglet "Gestion de serveur" d'Altova LicenseServer doit être inactive.

- L'action *Insérer Tout* est définie pour la table de base de données pertinente.
- La table dans laquelle vous voulez avoir les données bulk-insert ne doivent pas avoir de tables associées, des aperçus, ou des procédures stockées référençant la table dans le mappage.
- Le pilote de base de données prend en charge l'opération bulk-insert dans les conditions WHERE.

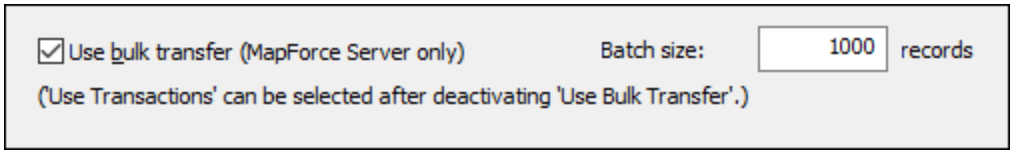
Prendre en charge l'opération bulk-insert dépend du type de base de données et du pilote utilisé (*table ci-dessous*).

| Type BD | ADO | ODBC | JDBC | ADO.NET | Natif |
|---|---|---|---|---|---|
| *Access* | Non | Non | n/a | n/a | n/a |
| *DB2* | Non | Oui | Oui | Oui | n/a |
| *Firebird* | n/a | Oui | Oui | Non | n/a |
| *Informix* | Non | Oui | Oui | Oui | n/a |
| *iSeries* | Non | Oui | Oui | Oui | n/a |
| *MariaDB* | Non | Oui | Oui | Non | n/a |
| *MySQL* | n/a | Oui *(MySQL Version 5 ou ultérieure si requis)* | Oui | Non | n/a |
| *Oracle* | Non | Oui | Oui | Oui | n/a |
| *PostgreSQL* | n/a | Oui | Oui | n/a | Oui |
| *Progress* | n/a | Oui | Oui | n/a | n/a |
| *Serveur SQL* | Oui | Oui | Oui | Oui | n/a |
| *SQLite* | n/a | n/a | n/a | n/a | Non |
| *Sybase* | Non | Oui | Oui | n/a | n/a |
| *Teradata* | n/a | Oui | Oui | n/a | n/a |

**Note :** Pour permettre une prise en charge bulk-insert de MySQL et MariaDB via JDBC, utilisez l'option de connexion `rewriteBatchedStatements=true`.

*BD échantillon*

Pour tester l'opération bulk-insert, vous pouvez utiliser le Serveur SQL et la base de données AdventureWorks. Lorsque vous chargez cette base de données dans votre mappage et ouvrez le dialogue **Actions de Table de base de données**, vous serez à même de spécifier une taille batch (1000 enregistrements dans notre exemple). La taille batch définit le nombre d'enregistrements à insérer à la fois. Il est important de noter que l'option *Utiliser le Transfert Bulk* et l'option *Utiliser des transactions* sont exclusives mutuellement : Quand une des options est activée, l'autre option devient inactive.




*Étape suivante*
Maintenant que l'option bulk-insert est activée, la prochaine étape est d'exécuter le mappage dans MapForce Server[864] (autonome ou sous la Gestion de FlowForce Server[867]).

## 4.2.3.2 Actions de table BD : Scénarios

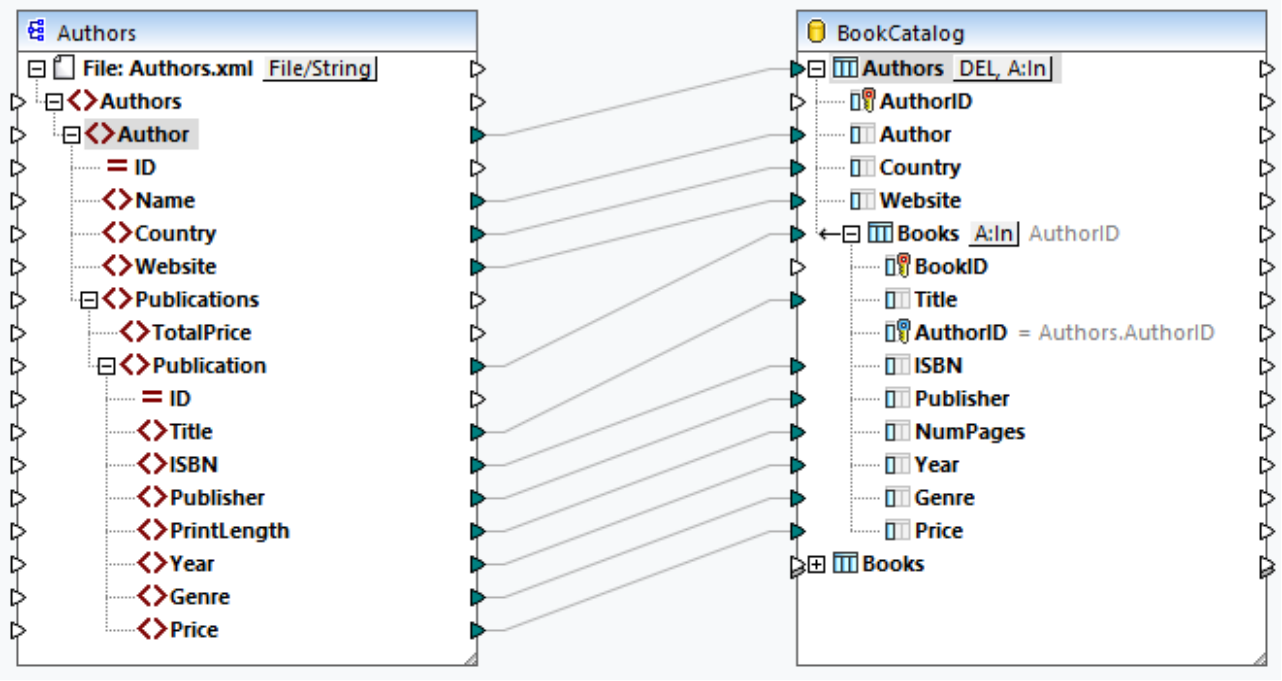
**Site web d'Altova :** MapForce, les vidéos démo

Ce chapitre discute certains des scénarios possibles pour utiliser les actions de Table de base de données[280]. Tous les scénarios utilisent une base de données hiérarchique appelée **`BookCatalog.sqlite`**. Cette base de données a trois tables : `Authors` (parent), `Books` (child), `TrackingInfo` (non connecté à toute autre table). Les tables `Authors` et `Books` ont une relation de clé étrangère. Il est important de noter qu'il y a configuration autoincrement pour les clés primaires dans les tables `Authors` et `Books`. La structure de la base de données est décrite dans le script ci-dessous :

```
CREATE TABLE
       "main"."Authors" (
  "AuthorID" INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
       "Author" TEXT,
       "Country" TEXT,
       "Website" TEXT
);

CREATE TABLE
       "main"."Books" (
       "BookID" INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
       "Title" TEXT,
       "AuthorID" INTEGER,
       "ISBN" TEXT,
       "Publisher" TEXT,
       "NumPages" INTEGER,
       "Year" INTEGER,
       "Genre" TEXT,
       "Price" DECIMAL,
       FOREIGN KEY ("AuthorID") REFERENCES "Authors" ("AuthorID") ON DELETE CASCADE
ON UPDATE NO ACTION
);

CREATE TABLE
       "main"."TrackingInfo" (
       "MappedOn" DATETIME
);
```

## Scénario 1 : Supprime toutes les données BD et insère toutes les données source

Dans le premier scénario, nous voulons supprimer *toutes* les données depuis la base de données **`BookCatalog.sqlite`** et peupler la base de données avec *toutes* les données depuis le fichier source. Notre mappage a l'air de ceci :




Même si la base de données a trois tables, seules `Authors` et `Books` sont incluses dans le composant de base de données. Puisque rien n'est mappé dans la table `TrackingInfo`, cette table est absente depuis le composant. Puisque « autoincrement » est définie pour les tables `Authors` et `Books`, nous n'avons pas besoin de connecter quoique ce soit dans les colonnes `AuthorID` et `BookID` : ces ID seront générées automatiquement par la base de données.

*Actions de table BD pour les auteurs*

Les actions de table pour la table `Authors` (*capture d'écran ci-dessous*) ont été configurées de la manière suivante :

- Dans la section *instruction SQL à exécuter avant le premier enregistrement*, nous avons défini l'action SUPPRIMER qui supprimera tous les enregistrements de la base de données, y compris tous les enregistrements enfant.
- Dans la section *Actions à exécuter pour chaque enregistrement*, nous avons défini l'action *Insérer tout*.
- Les ID Author seront générées automatiquement par la base de données (l'option *BD générée* dans l'action *Insérer tout*).
- Les autres valeurs seront mappées depuis le fichier source.

| Action on record | NULL equal | Insert All |
|---|---|---|
| AuthorID | | DB-generated |
| Author | | mapped value |
| Country | | mapped value |
| Website | | mapped value |

*Actions de table BD pour les Books*
Pour la table `Books`, nous avons défini la même action *Insérer tout* (*capture d'écran ci-dessous*). Les ID Book seront générées par la base de données. La colonne `AuthorID` référence la clé primaire dans la table `Authors`. Les valeurs pour cette colonne seront fournies automatiquement. Toutes les autres valeurs seront mappées depuis le fichier source.

| Action on record | NULL equal | Insert All |
|---|---|---|
| BookID | | DB-generated |
| Title | | mapped value |
| AuthorID | | foreign key |
| ISBN | | mapped value |
| Publisher | | mapped value |
| NumPages | | mapped value |
| Year | | mapped value |
| Genre | | mapped value |
| Price | | mapped value |

*Sortie*
La liste de codes ci-dessous affiche un extrait de la sortie :

```
DELETE FROM "Books"

DELETE FROM Authors

INSERT INTO "Authors" ("Author", "Country", "Website") VALUES ('Stephen King', 'US',
'www.stephenking.com')

SELECT "AuthorID" FROM "Authors" WHERE "AuthorID" = last_insert_rowid()
-- >>> %AuthorID1%

INSERT INTO "Books" ("AuthorID", "Title", "ISBN", "Publisher", "NumPages", "Year",
"Genre", "Price") VALUES ('%AuthorID1%', 'Misery', '1501143107', 'Scribner', 368,
2016, 'Horror', 11.99)

INSERT INTO "Books" ("AuthorID", "Title", "ISBN", "Publisher", "NumPages", "Year",
"Genre", "Price") VALUES ('%AuthorID1%', 'Outsider', '1501180983', 'Scribner', 576,
2018, 'Horror', 12.79)
```

Notez que les instructions SQL dans la sortie sont à des fins d'information uniquement. Pour exécuter les instructions SQL, ouvrez le volet Sortie et exécutez la commande de barre d'outils **Exécuter Script SQL/NoSQL**. Pour plus d'information, voir *Instructions SQL dans la Sortie* 279.

## Scénario 2 : Mettre à jour les auteurs et livres, insérer rest, insère l'info de traçage

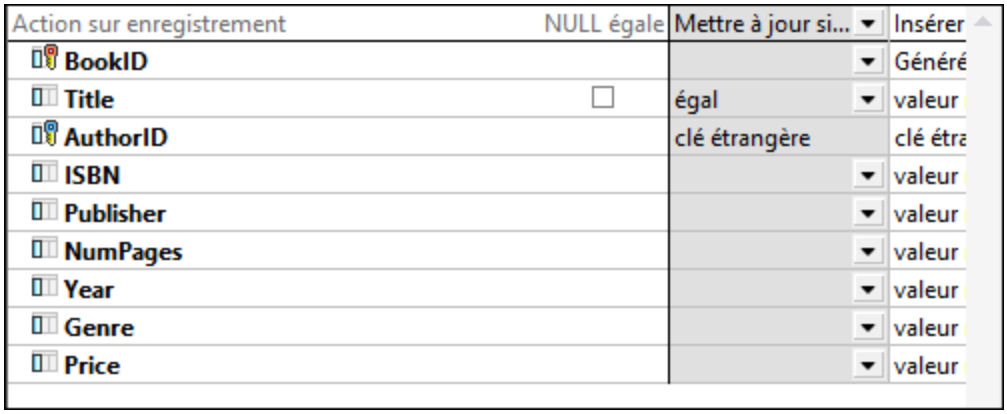
Dans des situations réelles, la base de données est constamment en train de changer, et plusieurs personnes pourraient travailler dans la même base de données. Pour cette raison, supprimer tous les enregistrements de base de données peut être indésirable. Dans le deuxième scénario, nous avons l'objectif suivant :

- Pour mettre à jour les auteurs et livres qui existent dans le fichier source et la base de données
- Pour insérer les enregistrements source qui n'existent pas dans la base de données
- Pour fournir une information de traçage sur la date et l'heure du mappage

Notre mappage a les mêmes composants et connexions comme dans le premier scénario. Toutefois, les actions de table de base de données ont été configurées différemment (*voir ci-dessous*)

*Actions de table BD pour les auteurs*
La capture d'écran ci-dessous affiche les actions définies pour la table des `Authors`. Dans la condition *Update If*, nous avons défini la valeur *equal* dans la colonne `Author`. Ceci signifie que 'il y a des auteurs avec le même nom dans la source et dans la base de données, uniquement les enregistrements d'auteur seront mis à jour dans la base de données. Les auteurs qui existent uniquement dans le fichier source seront simplement insérés dans la base de données. Les auteurs qui existent uniquement dans la base de données resteront inchangés.


*Actions de table BD pour les Books*
Nous avons la même combinaison d'actions pour la table `Books` (*voir ci-dessous*). Cette fois, la valeur égale est définie dans la colonne `Titre`. Ceci signifie qu'il y a des mêmes livres dans la source et la base de données, les enregistrements book seront mis à jour dans la base de données. Les Livres qui existent uniquement dans la source seront insérés dans la base de données. Les Livres qui existent uniquement dans la base de données resteront inchangés.



*Aucun enregistrement dupliqué*
L'avantage majeur de la condition *Update If* est qu'elle prévient d'obtenir des enregistrements doubles dans la base de données.

*Sortie*
La liste de codes ci-dessous affiche un extrait de la sortie :

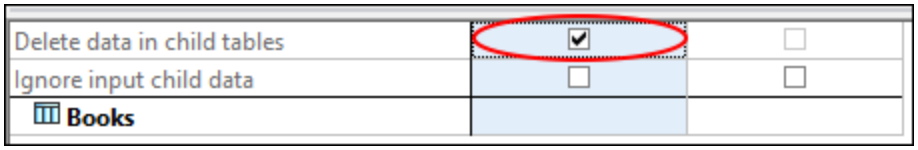
```
UPDATE "Authors" SET "Country" = 'US', "Website" = 'www.stephenking.com' WHERE
("Authors"."Author" = 'Stephen King')

SELECT "AuthorID" FROM "Authors" WHERE ("Author" = 'Stephen King')
-- >>> %AuthorID1%

UPDATE "Books" SET "ISBN" = '1501143107', "Publisher" = 'Scribner', "NumPages" =
368, "Year" = 2016, "Genre" = 'Horror', "Price" = 11.99 WHERE ("Books"."AuthorID" =
'%AuthorID1%') AND ("Books"."Title" = 'Misery')

UPDATE "Books" SET "ISBN" = '1501180983', "Publisher" = 'Scribner', "NumPages" =
576, "Year" = 2018, "Genre" = 'Horror', "Price" = 12.79 WHERE ("Books"."AuthorID" =
'%AuthorID1%') AND ("Books"."Title" = 'Outsider')
```

*Solution alternative*
Pour éviter les enregistrements doubles dans la table `Books`, vous pouvez aussi choisir de supprimer les enregistrements enfant[284] (*voir ci-dessous*) et définir la condition *Insert All* dans la table `Books`. Les actions pour la table `Authors` restent inchangées. Dans cette configuration, tous les enregistrements `Book` dont les auteurs existent dans la source et la base de données seront supprimés de la base de données. Avec la condition *Insérer Tout* définie dans la table `Books`, les types suivants d'enregistrements source `Book` seront insérés :

- Les enregistrements `Book` dont les auteurs existent dans la source et la base de données
- Les enregistrements `Book` des nouveaux auteurs qui existent uniquement dans la source

S'il existe un enregistrement source `Book` sans un parent, cet enregistrement `Book` sera aussi mappé dans la base de données et son enregistrement parent sera créé dans la table `Authors`. Le nouvel enregistrement `Author` recevra une nouvelle ID (clé primaire), et tous les autres champs recevront des valeurs NULL. Ceci est possible *uniquement* si tous les champs dans la table `Authors`, à l'exception de la clé primaire, sont nullables. Si les champs ne sont pas nullables, vous recevrez un message d'erreur disant que la contrainte NOT NULL a échoué.

De manière importante, les résultats du scénario principal et la solution alternative peuvent diverger. Par exemple, si un auteur dans la base de données a cinq enregistrements book, et si le même auteur a uniquement trois enregistrements dans la source, tous les cinq enregistrements de base de données seront supprimés et seront remplacés par trois enregistrements de la source.




*Insérer une info de traçage*
Quand plusieurs personnes travaillent dans la même base de données, il pourrait être une bonne idée de savoir quand le mappage a été exécuté. Il pourrait y avoir différentes possibilités : par ex., vous pouvez utiliser différentes fonctions `datetime`[646] ; vous pouvez aussi fournir une instruction SQL personnalisée dans le

dialogue Actions de Table de base de données[282]. Dans notre exemple, nous ajouterons une instruction SQL suivante aux *Actions pour exécuter chaque section d'enregistrement* (table`Authors`) :

```
INSERT INTO TrackingInfo VALUES (DATETIME())
```

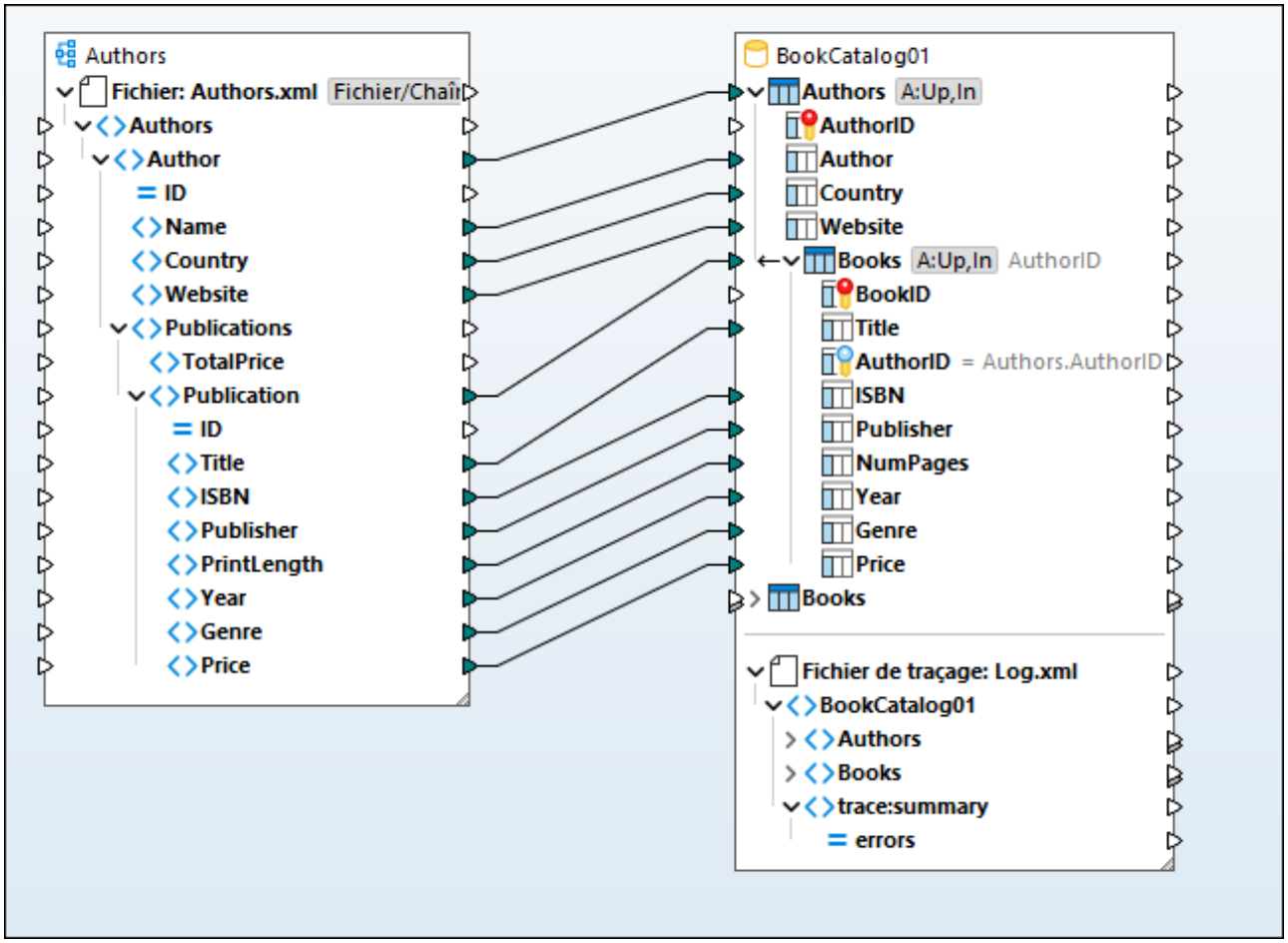
Quand vous exécutez le script SQL[279], cette instruction SQL sera exécutée d'abord, avant toute instruction pour les enregistrements de base de données.

*Quelques autres scénarios possibles*
À la place de la condition *Update If*, vous pouvez aussi définir l'action *Delete If*. Dans ce cas, les enregistrements `Author` et `Book` qui existent dans la source et la base de données seront supprimés, et les nouveaux enregistrements seront insérés dans l'action de base de données (*Insert Rest*). Vous pouvez aussi choisir d'ignorer les mêmes enregistrements (condition *Ignore If*) et d'insérer de nouveaux enregistrements dans la base de données (action *Insert Rest*).

### 4.2.3.3 Rollback de transaction: Scénarios

Quand vous mappez les données dans une base de données, vous pouvez rencontrer diverses erreurs liées à la base de données (par ex., le compte de base de données ne peut pas avoir assez de privilèges pour réaliser une action de base de données spécifique). Pour prévenir de telles erreurs d'annulation de l'exécution du mappage, vous pouvez configurer des paramètres de restauration de la transaction, qui vous permettrons de restaurer les changements. Vous pouvez activer une restauration de transaction au niveau du composant de base de données[259] au niveau de l'action de table[284], et au niveau de la procédure stockée[334].

Ce chapitre décrit quelques-uns des scénarios possibles des rollbacks de transaction. Tous les scénarios de ce chapitre caractérisent un fichier source XML appelé **`Authors.xml`** et une base de données cible appelée **`BookCatalog.sqlite`** (*mappage ci-dessous*). Les tables `Authors` et `Books` ont une relation de clé étrangère. La table `Author` est le parent de la table `Books`.




Dans tous les scénarios, les mêmes actions de table de base de données[280] ont été configurées :

- La combinaison des actions *Update If* et *Insert Rest* dans la table `Authors`. La condition *Update If* est définie dans la colonne `Author`.
- La combinaison des actions *Update If* et *Insert Rest* dans la table `Books`. La condition *Update If* est définie dans la colonne `Titre`.

De plus, le traçage[285] a été activé au niveau du composant (défini à *Toujours*), dans la table `Authors` (*Utiliser les paramètres de composant*), et la table `Books` (*Utiliser les paramètres de composant*). Les paramètres de rollback de transaction rollback varieront dépendant du scénario.

*Présentation sommaire des scénarios*
Les scénarios décrits dans ce chapitre sont décrits ci-dessous :

- *Scénario 1 :* Si un enregistrement `Author` est erroné, il sera restauré avec tous ses enregistrements enfant ; si seul un enregistrement `Book` est erroné, alors seul cet enregistrement sera renvoyé.
- *Scénario 2 :* Si un enregistrement `Author` est erroné, il sera retourné avec tous ses enrgistrements enfant.
- *Scénario 3 :* Si un enregistrement `Book` est erroné, seul cet enregistrement sera restauré.
- *Scénario 4 :* Si un enregistrement `Book` ne peut pas être inséré, ses enregistrements parent seront restaurés (de manière semblable au *Scénario 2*).

- *Scénario 5 :* S'il existe une erreur associée à la base de données, tous les changements effectués dans la base de données seront restaurés.

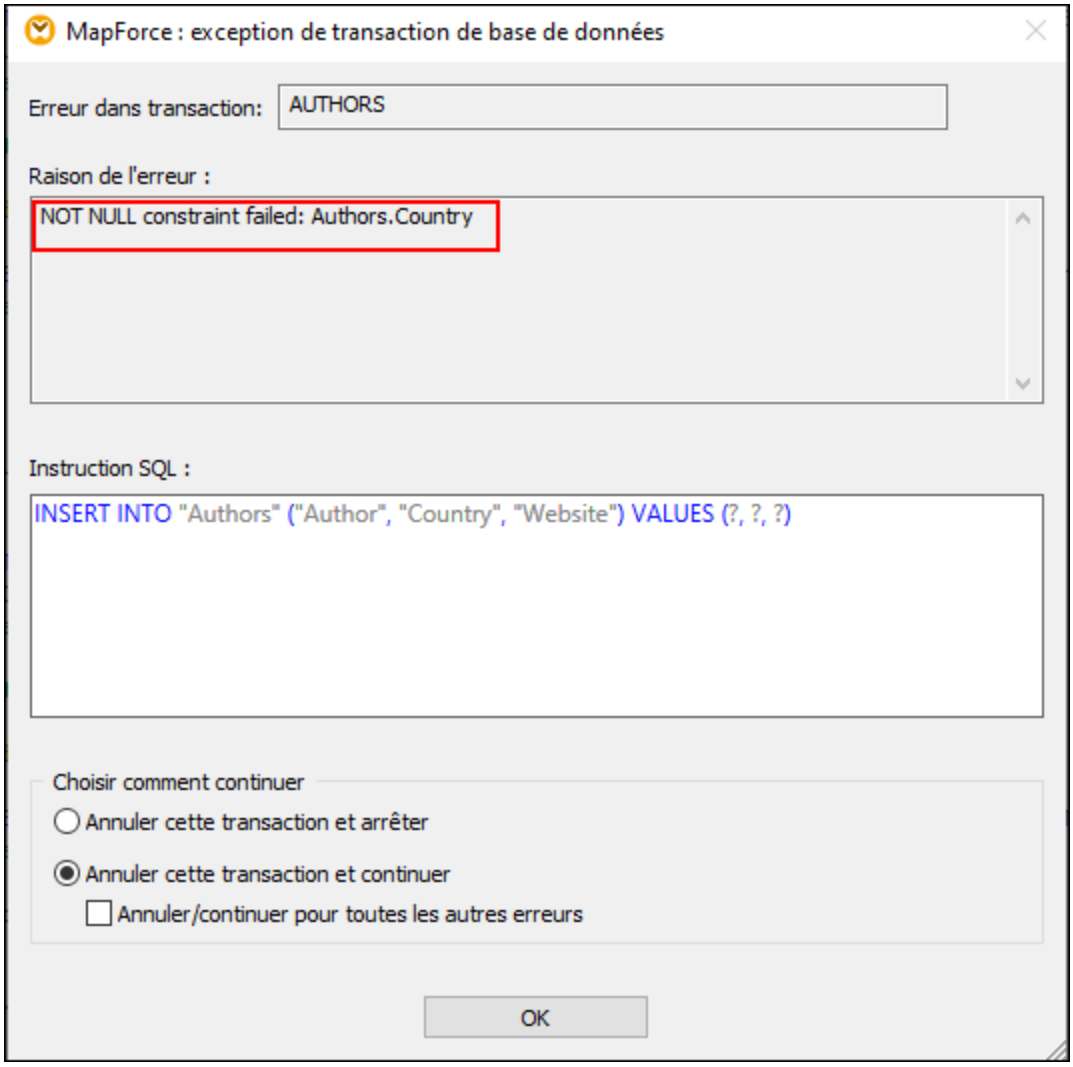
## Scénario 1 : Rollback Author et Book actuels

Dans le premier scénario, nous permettons la gestion de transaction au niveau de l'action de table : L'option *Restaurer transaction actuelle et continuer* est définie dans les deux tables de la base de données. La combinaison de ces paramètres signifient que (i) s'il y a un enregistrement `Author` défectueux, ni cet enregistrement, ni ses enregistrements enfant `Book` ne seront insérés dans la base de données, puis le traitement du prochain enregistrement démarrera ; (ii) s'il existe un enregistrement `Book` défectueux mais que son enregistrement parent `Author` est valide, uniquement l'enregistrement `Book` défectueux sera annulé, et le traitement du prochain enregistrement se démarrer.

*Authors.xml*
Le fichier **`Authors.xml`** contient des informations sur les auteurs et leurs livres. Un des enregistrements `Author` manque d'information sur le pays de l'auteur. Le fichier source a également un enregistrement `Book` qui ne contient pas d'information sur l'éditeur. :

*BookCatalog.sqlite*
Nous avons défini une contrainte `NOT NULL` dans la colonne `Country` dans la table `Authors` et la colonne `Publisher` dans la table `Books`. Mapper les valeurs null à ces colonnes causera une erreur.

*Sortie*
Après avoir exécuté le script SQL [279] dans le volet Sortie, nous obtenons le dialogue suivant :



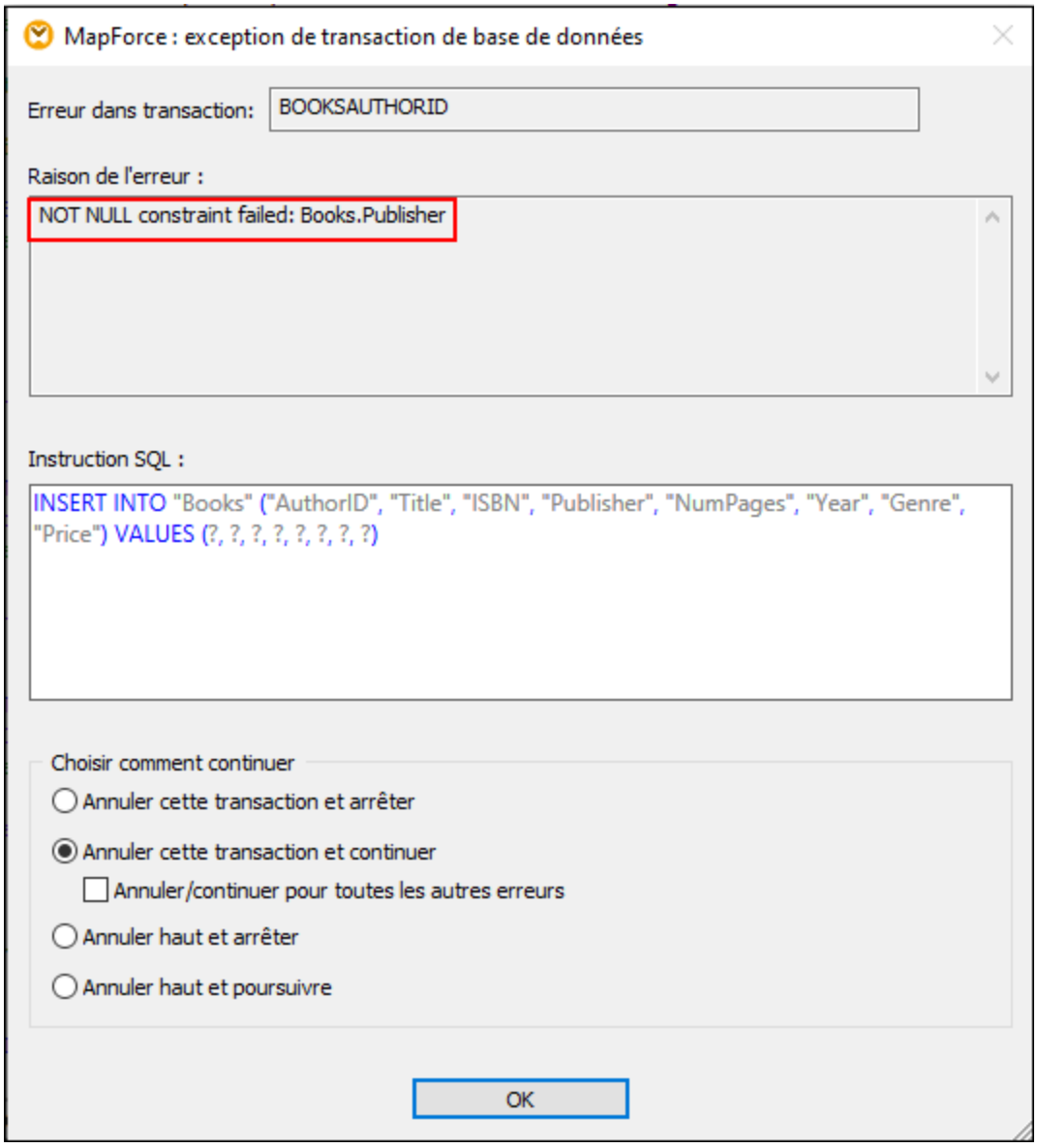

Le dialogue **Exception de transaction de base de données** nous informe sur l'erreur liée à la base de données et la raison de cette erreur. Dans cet exemple, une valeur null n'a pas pu être insérée dans la colonne `Country` de la table `Authors` (*rectangle rouge ci-dessus*). Cette boîte de dialogue permet aussi de choisir la prochaine action. Par défaut, l'action configurée dans le dialogue Actions de Table de base de données 280 est sélectionnée. Il est possible d'appliquer le même paramètre à toutes les erreurs suivantes ou d'annuler la transaction défectueuse et d'arrêter.

Après avoir confirmé la sélection, nous recevons une autre notification sur une autre erreur qui est apparue dans la table `Books` (*capture d'écran ci-dessous*). Cette erreur est apparue car le mappage a tenté d'insérer l'information absente `Publisher` d'un de nos enregistrements `Book` dans le champ `Publisher` de la table `Books`. Puisque l'erreur est apparue dans la table enfant, il est possible de restaurer la transaction actuelle (*Restaurer cette transaction et arrêter/continuer*) ou les changements de la transaction supérieure (*Restaurer le haut et arrêter/continuer*). La transaction du niveau supérieur dans ce contexte signifie l'enregistrement parent (`Author`) de l'enregistrement enfant erroné (`Book`).

S'il existe plusieurs niveaux dans la hiérarchie, une restauration aura lieu pour chaque niveau jusqu'à ce que la restauration atteigne le niveau le plus élevé auquel la transaction a été activée. Selon vos besoins, vous pouvez

choisir de continuer ou arrêter l'exécution de mappage. L'option *Restaurer le haut et continuer/arrêter* est disponible uniquement lorsqu'il existe des transactions imbriquées. Pour plus d'information sur la restauration des transactions du haut, voir le *Scénario 4* ci-dessous.




Dans cet exemple, nous laissons l'option de restauration de la transaction affichée dans la capture d'écran ci-dessus inchangés. Après la fin du traitement, de tous les enregistrements, nous pouvons voir les instructions SQL générées dans le volet Sortie, le rapport de trace[260] sous le format XML, et les données qui sont désormais disponibles dans la base de données.

*Instructions SQL générées*

Un extrait du script SQL généré est illustré ci-dessous. L'opération insert échouée a été restaurée dans `SAVEPOINT`. La commande `SAVEPOINT` représente un point de transaction auquel la transaction sera restaurée. La commande `SAVEPOINT` permet de défaire les changements effectués après `SAVEPOINT` et de restaurer la transaction à l'état au moment du `SAVEPOINT`.

```
SAVEPOINT "BOOKSAUTHORID"
-- >>> OK. One or more rows.

UPDATE "Books" SET "ISBN" = '0099590328', "Publisher" = NULL, "NumPages" = 464, "Year" = 2016, "Genre" =
'Crime & Mystery', "Price" = 8.6 WHERE ("Books"."AuthorID" = 41) AND ("Books"."Title" = 'Cockroaches')
-- >>> OK. 0 row(s).

INSERT INTO "Books" ("AuthorID", "Title", "ISBN", "Publisher", "NumPages", "Year", "Genre", "Price")
VALUES (41, 'Cockroaches', '0099590328', NULL, 464, 2016, 'Crime & Mystery', 8.6)

>>> FAILED!
-->>> NOT NULL constraint failed: Books.Publisher


ROLLBACK TRANSACTION TO SAVEPOINT "BOOKSAUTHORID"
-- >>> OK. One or more rows.
```

*Rapport de trace*

Un extrait du rapport de trace est fourni ci-dessous : Le rapport de trace affiche les champs et leurs valeurs qui ont été tracées (par ex., `<Price>8.6</Price>`), fournit des informations sur les actions réalisées (par ex., `insert`) et l'erreur (voir l'élément `trace:error`).

```
<Books>
      <trace:values>
            <Title>Cockroaches</Title>
            <AuthorID>41</AuthorID>
            <ISBN>0099590328</ISBN>
            <Publisher xsi:nil="true"/>
            <NumPages>464</NumPages>
            <Year>2016</Year>
            <Genre>Crime &amp;amp; Mystery</Genre>
            <Price>8.6</Price>
      </ trace:values>
      <trace:actions>
            <trace:update rows-affected="0"/>
            <trace:insert>
                  <trace:error code="19" state="1299">contrainte NOT NULL échouée :
                  Books.Publisher</trace:error>
            </ trace:insert>
      </ trace:actions>
</Books>
```

*Mettre à jour les données dans la BD*

Pour voir quelles données sont disponibles dans la base de données, vous pouvez utiliser le volet Requête BD[300]. L'extrait de la table `Books` ci-dessous affiche que plusieurs enregistrements ont été insérés avec succès. Puisqu'insérer un des livres de Jo Nesbo a échoué (*rapport de trace ci-dessus*), uniquement deux de ces trois livres ont été mappés (`The Bat` et `The Redbreast`).

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 52 | 52 | Night | 35 | 1473678161 | Mulholland Books UK | 384 | 2020 | Crime & Mystery | 13.11 |
| 53 | 53 | Don Quixote | 36 | 0142437239 | Penguin Classics | 1072 | 2003 | Classics | 13.99 |
| 54 | 54 | The Count of Monte Cristo | 37 | 0140449264 | Penguin Classics | 1276 | 2003 | Classics | 13.99 |
| 55 | 55 | The Odyssey | 38 | 0141192445 | Penguin Classics | 416 | 2010 | Classics | 18.49 |
| 56 | 56 | The Divine Comedy | 39 | 1435162064 | Barnes & Noble | 492 | 2001 | Classics | 25 |
| 57 | 57 | Miss Peregrine's Home for Peculiar Children | 40 | 9781594746031 | Random House UK Ltd | 384 | 2013 | Fantasy | 2.42 |
| 58 | 58 | The Bat | 41 | 9780099581871 | Random House UK Ltd | 432 | 2013 | Crime & Mystery | 8.93 |
| 59 | 59 | The Redbreast | 41 | 0099546779 | Vintage | 624 | 2009 | Crime & Mystery | 9.03 |

## Autres scénarios possibles

D'autres scénarios possibles sont décrits ci-dessous.

*Scénario 2 : Restaurer Author actuel et tous les livres de l'auteur*

Dans ce scénario, la *transaction de restauration actuelle et l'option continuer* est définie uniquement dans la table `Authors`. Au sein de cette configuration, chaque enregistrement `Author` et les enregistrements correspondants `Book` sont perçus comme opération atomique. S'il existe un enregistrement `Author` erroné ou au moins un des livres d'auteur qui est erroné, cet enregistrement de l'auteur et ses enregistrements enfant associés seront restaurés. Puis, le traitement continue avec le prochain enregistrement `Author`.

*Scénario 3 : Restaurer uniquement Book actuel*

Dans ce scénario, la *transaction de restauration actuelle et l'option continuer* est définie uniquement dans la table `Livres`. S'il existe un enregistrement erroné `Book`, seul l'enregistrement `Book` sera restauré, et le traitement continuera. Si vous définissez la *transaction de restauration du niveau supérieur et l'option continuer* dans la table `Books` et aucune autre option de gestion de la transaction au niveau parent, ce ne sera que l'enregistrement `Book` erroné qui sera restauré. Dans ce cas, il n'y a pas de différence entre une restauration de la transaction du niveau supérieur et actuel, car il n'existe qu'un niveau de transaction et pas d'imbrication de transaction.

*Scénario 4 : Restauration de Author si Book ne peut pas être inséré*

Dans ce scénario, l'action de transaction pour la table `Authors` est définie pour *Restaurer la transaction actuelle et continuer*, et l'action de transaction pour la table `Books` est *Restaurer la transaction du niveau supérieur et poursuivre*. Avec cette configuration, si insérer un enregistrement enfant échoue, l'opération de restauration sera effectuée pour chaque niveau de la hiérarchie, jusqu'au niveau le plus élevé auquel la gestion de la transaction a été activée. Par exemple, nous voulons insérer un nouvel enregistrement `Author` (`Jo Nesbo`) et ses trois enregistrements enfant. Dans un des enregistrements `Book` (`The Cockroaches`), il n'existe aucune information sur l'éditeur. Puisque la table `Books` n'a pas de contrainte NOT NULL dans la colonne `Publisher`, mapper la valeur null dans ce champ causera une erreur. Dans ce scénario, lorsqu'une erreur apparaît, l'enregistrement `Book` erroné (`The Cockroaches`) sera restauré ensemble avec les deux autres enregistrements `Book` et l'enregistrement parent. Puis, le traitement continuera avec le prochain enregistrement `Author`.

La combinaison des options de gestion de transaction décrite dans ce scénario est particulièrement pertinente pour les structures complexes et imbriquées (par ex., une table parent avec deux ou plusieurs tables enfant)

*Scénario 5 : Restaurer toutes les transactions de BD et continuer*

Dans ce scénario, la *transaction de restauration actuelle et l'option continuer* sont définies uniquement dans la table Livres 256. Ce paramètre peut être particulièrement utile quand vous avez un mappage en chaîne (par ex., XML-DB-JSON). S'il existe une erreur associée à la base de données, tous les changements réalisés dans les composants de base de données seront restaurés et le traitement continuera avec le composant JSON.

### 4.2.3.4 Instructions MERGE

Pour certains mappages, MapForce génère des instructions MERGE (*capture d'écran ci-dessous*) qui seront exécutés par rapport à la base de données au moment de la marche du mappage. L'avantage des instructions MERGE réduisent le nombre d'appels de serveur de base de données, puisque ces instructions combinent les instructions INSERT et UPDATE en une instruction. De même, en cas d'instructions MERGE, la vérification de cohérence est effectuée par la base de données. Les instructions MERGE sont appuyées si :

- la base de données est l'une des suivantes : SQL Server 2008 et plus récent, Oracle, DB2, Firebird ;
- la base de données cible a une combinaison des actions de table *Insert If* et *Insert Rest* actions de table[283]

Si les instructions MERGE ne sont pas prises en charge par le type de base de données, le script SQL généré contient uniquement des instructions UPDATE. Aucune instruction INSERT n'est visible pour l'affichage, puisque celle-ci est exécutée uniquement si la condition *Update If* n'est pas satisfaite.

```
SET QUOTED_IDENTIFIER ON

UPDATE [dbo].[Users] SET [FirstName] = (CAST('Despine' AS nvarchar(50))), [LastName] =
(CAST('Buttler' AS nvarchar(50))) WHERE ([dbo].[Users].[UserID] = 1)

SELECT [UserID] FROM [dbo].[Users] WHERE ([UserID] = 1)
-->>> %UserID1%

DELETE FROM [dbo].[Addresses] WHERE EXISTS(SELECT * FROM [dbo].[Users] WHERE [dbo].[Users].[UserID] =
[dbo].[Addresses].[UserID] AND ([dbo].[Users].[UserID] = '%UserID1%'))

MERGE INTO [dbo].[Addresses] AS T USING ( VALUES ( '%UserID1%', 1, 1, (CAST('Home' AS nvarchar(20))),
(CAST('Louisville' AS nvarchar(50))), (CAST('Elm Street' AS nvarchar(50))), (CAST('12' AS
nvarchar(20))) ) ) AS S ( [UserID], [IsShipping], [IsBilling], [AddressType], [City], [Street],
[Number] ) ON ( (S.[UserID] = T.[UserID]) ) WHEN MATCHED THEN UPDATE SET [IsShipping] = S.
[IsShipping], [IsBilling] = S.[IsBilling], [AddressType] = S.[AddressType], [City] = S.[City],
[Street] = S.[Street], [Number] = S.[Number] WHEN NOT MATCHED THEN INSERT ( [UserID], [IsShipping],
[IsBilling], [AddressType], [City], [Street], [Number] ) VALUES ( S.[UserID], S.[IsShipping], S.
[IsBilling], S.[AddressType], S.[City], S.[Street], S.[Number] );
```

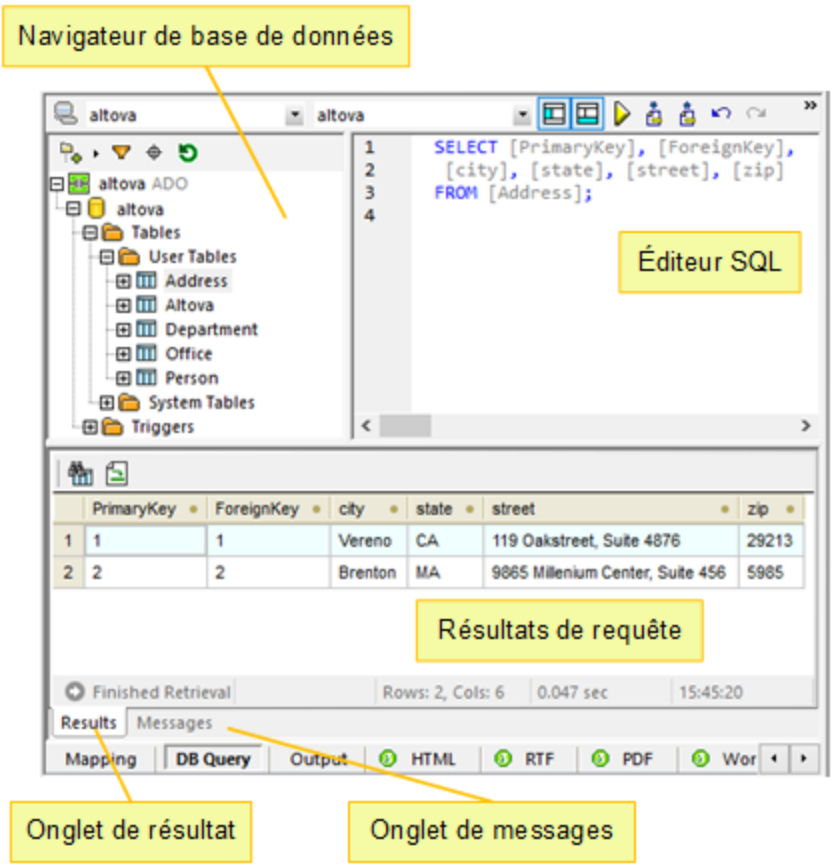
Si vous mettez à jour de multiples tables ayant des relations parent-enfant, les fusions sont créées uniquement pour les tables vers lesquelles les données sont mappées. Par exemple, si une base de données a une table parent appelée `Authors` et une table enfant appelée `Books`, l'instruction MERGE sera générée uniquement pour la table `Books`. Pour la table `Authors`, les instructions UPDATE seront générées à la place.

Avec des instructions MERGE, l'option Bulk Transfer[286] est prise en charge uniquement pour des connexions de base de données ODBC et JDBC.

## 4.2.4 Requête de bases de données

MapForce a un volet requête BD (*illustré ci-dessous*) qui vous permet d'interroger ou de modifier votre base de données indépendamment du processus de mappage. De telles requêtes directes ne sont pas enregistrées ensemble avec le mappage. Un volet Requête BD séparé existe pour chaque mappage actif actuellement. Dans chaque volet Requête BD, vous pouvez vous connecter à plusieurs bases de données différentes. Veuillez noter que les connexions créées avant le volet Requête BD ne font pas partie du mappage et ne sont

pas préservés une fois que vous fermez MapForce à moins que vous les définissiez ces connexions tant que Ressources globales[881].




Le volet Requête de BD de données consiste en les parties suivantes :

- Le *Navigateur de base de données*[302], qui affiche une information de connexion et des tables de base de données ;
- *Éditeur SQL*[304], dans lequel vous rédigez vos requêtes SQL ;
- *Onglet Résultats*[307], qui affiche les résultats de recherche dans une format tabulaire ;
- *Onglet Messages*[309], qui affiche des avertissements et des messages d'erreurs.

Pour configurer des paramètres de requête de base de données, utilisez la section *Base de données* dans la boîte de dialogue **Options**. Pour plus d'information, voir Paramètres de requête de base de données[1092].

*Se connecter à une base de données*

Avant d'interroger une base de données[304], vous devez vous connectez à la base de données. Si votre mappage inclut déjà un composant de base de données, vous pouvez sélectionner une connexion de base de données existante depuis une partie supérieure du volet Requête BD et commencez à interroger votre base de données.

Si votre mappage ne contient pas de composant de base de données, ou si vous souhaitez vous connecter à une nouvelle base de données, cliquer sur le bouton, cliquez sur le bouton (**Connexion rapide**) et suivez les étapes de l'assistant (voir Exemples[193]). Vous pouvez aussi sélectionner une connexion de base de données existante depuis les Ressources globales[881].

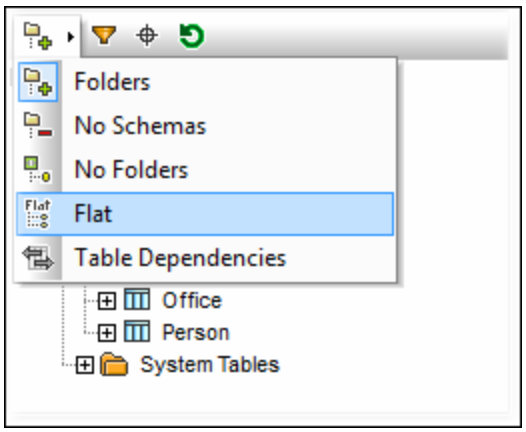
## 4.2.4.1 Navigateur de base de données

Ce chapitre explique comment personnaliser l'affichage de l'arborescence de la base de données, filtrer et rechercher des objets de base de données. Il décrit également les options du menu contextuel qui sont disponibles pour différents objets de base de données.

Lorsque vous êtes connecté à une ou plusieurs bases de données, le Navigateur de base de données donne un aperçu complet des objets contenus dans chaque base de données, y compris des tables, des vues, des procédures, etc. Pour des bases de données prenant XML en charge, le Navigateur de base de données montre également des schémas XML enregistrés dans un dossier séparé.

### Mise en page de l'arborescence de la BD de données

Le navigateur de base de données vous permet de personnaliser l'affichage de l'arborescence de la base de données. Les mises en page prédéfinies sont disponibles dans la partie supérieure du Navigateur de base de données (*capture d'écran ci-dessous*).




Pour sélectionner une mise en page, cliquez sur le bouton  (**Mise en page des dossiers**) et sélectionnez l'option requise depuis la liste. Notez que le bouton change avec la mise en page sélectionnée. Les options suivantes sont disponibles :

- La mise en page *Dossiers* organise des objets de base de données dans des dossiers sur la base du type d'objet (*paramètre par défaut*).
- La mise en page *Pas de schémas* est semblable à la mise en page *Dossiers*, excepté qu'il n'y a pas de dossiers de schéma de la base de données.
- La mise en page *Aucuns dossiers* affiche les objets de base de données de mise en page dans une hiérarchie sans utiliser de dossiers.
- La mise en page *Flat* affiche les objets de base de données par le type (par ex., toutes les colonnes sont affichées dans un dossier séparé `Colonnes`).
- La mise en page *Dépendances de table* catégorise les tables basées sur leurs relations avec d'autres tables (par ex., les tables avec clés étrangères, les tables référencées).

Outre la navigation dans la mise en page, vous pouvez utiliser le Navigateur de base de données pour les tâches suivantes :

- Créer des instructions SQL 305
- Filtrer et rechercher des objets de base de données (*voir la sous-sections ci-dessous*)
- Trier les tables dans les tables *System* et *User* (*voir ci-dessous*)
- Réinitialiser l'objet racine de la source de données active (le bouton )

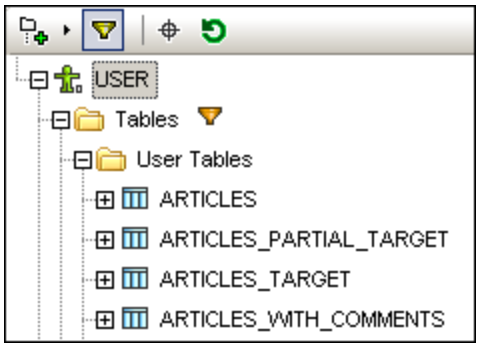
Pour trier des tables dans les tables *User* et *System*, cliquez avec la touche de droite dans le dossier `Tables` dans le Navigateur de base de données et sélectionnez **Trier dans les Tables User et System** de le menu contextuel. Note : cette fonction est disponible lorsqu'une des mises en page suivantes est sélectionnée : *Dossiers*, *Pas de schémas* ou *Plate*.

## Filtrer des objets de base de données

Vous pouvez filtrer tout objet de base de données (schémas, tables, vues, etc.) de données par le nom ou une partie du nom. Les objets sont filtrés pendant que vous saisissez les caractères. Le filtrage est insensible à la casse par défaut. Filtrer n'est pas pris en charge si vous avez choisi la mise en page Pas de Dossiers.

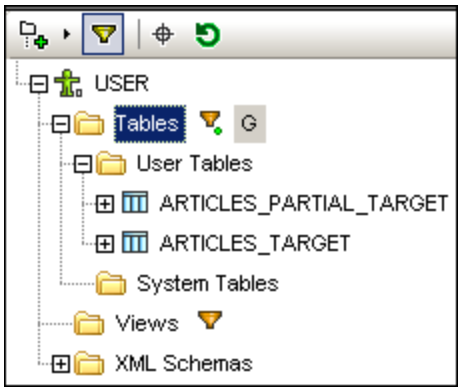
Pour filtrer des objets de base de données, suivez les étapes suivantes :

1. Cliquez sur le bouton en haut du Navigateur de base de données. Les icônes du filtre apparaissent à côté de tous les dossiers dans la mise en page sélectionnée actuellement (*voir la capture d'écran ci-dessous*).




2. Cliquez sur l'icône du filtre à côté du dossier que vous voulez filtrer et sélectionnez l'option de filtrage du menu contextuel (ex., **Contient**).


3. Saisissez le texte de recherche (par ex., *G*) dans le champ vide qui apparaît à côté de l'icône du filtre. Les résultats sont ajustés pendant la saisie (*capture d'écran ci-dessous*)..



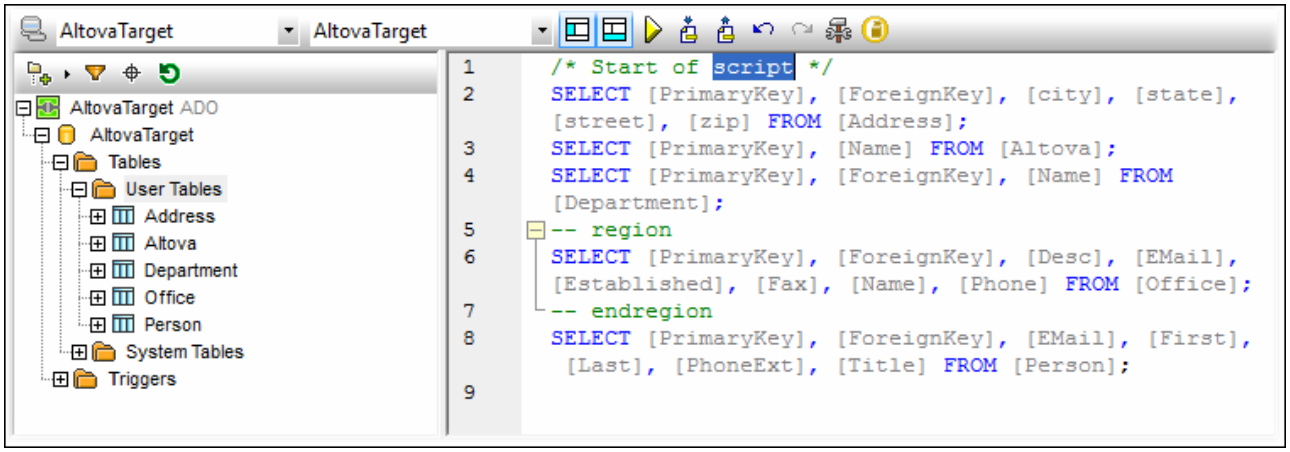

### Recherche des objets de base de données

Pour trouver un objet de base de données spécifique, vous pouvez utiliser la fonctionnalité de filtre (*voir ci-dessus*) ou le **localisateur d'objet**. Pour trouver des objets de base de données en utilisant le localisateur d'objet, suivez les instructions ci-dessous :

1. Cliquez sur le bouton  en haut du Navigateur de base de données.
2. Saisissez du texte de recherche dans la barre de texte.
3. Appuyez sur **Saisir** ou cliquez sur un objet dans la liste pour le sélectionner.

## 4.2.4.2 Éditeur SQL

Une fois que vous êtes connectés à votre base de données, vous pouvez utiliser un Éditeur SQL pour écrire et exécuter des instructions SQL. L'Éditeur SQL affiche toute instruction SQL que vous pouvez avoir généré automatiquement, chargé depuis des scripts SQL existants, ou écrit manuellement. L'éditeur SQL prend en charge le remplissage automatique (voir Remplissage automatique 256, régions, commentaires de ligne/ bloc.



## Boutons de la barre d'outils de l'Éditeur SQL

La barre d'outils de l'Éditeur SQL dispose des touches suivantes :

| | |
|---|---|
|  | Le bouton **Toggle Browser** rend le volet du navigateur visible/invisible. |
|  | Le bouton **Toggle Résultat** rend le Résultat visible/invisible. |
|  | Le bouton **Exécute requête** exécute les instructions SQL qui ont été sélectionnées actuellement. Si de multiples instructions existent et qu'aucune n'est sélectionnée, seuls les résultats de la dernière instruction seront affichés dans l'onglet Résultats. Pour de multiples résultats, MapForce suggérera ouvrir les résultats dans Altova DatabaseSpy. |
|  | Le bouton **Annuler** vous permet d'annuler un nombre illimté d'éditions dans la fenêtre SQL. |
|  | Le bouton **Rétablir** vous permet de rétablir des commandes annulées précédemment. |
|  | La commande **Importer fichier SQL** ouvre un script SQL externe, qui peut être exécuté. |
|  | La commande **Exporter fichier SQL** enregistre un script SQL à un emplacement désiré. |
|  | La commande **Ouvrir script SQL dans DatabaseSpy** ouvre le script SQL actuel dans Altova DatabaseSpy (doit être installé). |
|  | Le bouton **Options** ouvre le dialogue Options 1092 permettant de définir des paramètres de requête de base de données ainsi que des paramètres d'Éditeur SQL. |

## Créer des instructions SQL

Vous pouvez interroger votre base de données utilisant les méthodes suivantes :

- Vous pouvez importer un script SQL ou copier quelques instructions SQL et les coller dans l'Éditeur SQL (la commande de barre d'outils **Importer fichier SQL**).
- Vous pouvez écrire des instructions SQL dans l'Éditeur SQL manuellement.
- Cliquez avec la touche de droite sur un objet dans le Navigateur de base de données 302 et générer une requête (généralement, SELECT).

Pour générer une instruction SQL SELECT avant le Navigateur de base de données, procédez comme suit :

- Appuyez et gardez appuyé un objet de base de données (par ex., une table) ou un dossier dans le Navigateur de base de données, glissez cet objet du dossier dans l'Éditeur SQL.
- Cliquez avec la touche de droite de la souris sur l'objet de base de données dans le Navigateur en ligne et sélectionnez **Afficher dans le Éditeur SQL| SELECT**.

## Exécutez les instructions SQL

Les instructions SQL dans l'Éditeur SQL peuvent être exécutées avec un effet immédiat. Pour exécuter une instruction SQL, cliquez sur le bouton  . Le résultat de la requête SQL et le nombre de lignes affectées est affiché dans le volet Messages du volet Requête BD.

Quand de multiples instructions SQL apparaissent dans l'Éditeur SQL, seul le résultat de l'instruction sélectionnée ou le résultat de la dernière instruction (si aucune instruction n'a été sélectionnée) sera affiché dans l'onglet Résultats. Dans le cas de multiples résultats, MapForce suggérera de les ouvrir dans DatabseSpy.

## Importer/Exporter des scripts SQL

Vous pouvez importer et exporter les scripts SQL. Pour importer un fichier SQL externe, cliquez sur le bouton de la barre d'outils  et sélectionnez le fichier SQL que vous souhaitez ouvrir. Pour exporter les contenus du volet de l'Éditeur SQL vers un fichier SQL, cliquez sur le bouton de la barre d'outils  et saisissez le nom du script SQL.

## Ajouter/Supprimer les commentaires en ligne/bloc SQL

L'Éditeur SQL vous permet de commenter des instructions, des parties d'instructions, ou des groupes d'instructions. De telles instructions sont ignorées quand le script SQL est exécuté. Vous pouvez insérer une ligne ou un commentaire en bloc. Le commentaire en ligne indique que la ligne actuelle ou la partie restante de la ligne est commentée. Le commentaire en bloc peut s'étendre sur plusieurs lignes. Le commentaire en bloc peut aussi être utilisé pour commenter les mots individuels.

Pour insérer un commentaire en ligne/bloc, suivez les étapes suivantes :

1. Sélectionner une instruction ou une partie d'instruction.
2. Cliquez avec la touche de droite sur l'instruction sélectionnée et choisissez **Insérer/Supprimer ligne/commentaire en bloc** depuis le menu contextuel.

La capture d'écran ci-dessous illustre un commentaire en bloc (*texte vert*).

```
2   /*
3   SELECT [PrimaryKey], [ForeignKey], [city], [state],
    [street], [zip] FROM [Address];
4   SELECT [PrimaryKey], [Name] FROM [Altova];
5   SELECT [PrimaryKey], [ForeignKey], [Name] FROM
    [Department];*/
6
7   SELECT [PrimaryKey], [ForeignKey], [Desc], [EMail],
    [Established], [Fax], [Name], [Phone] FROM [Office];
8   SELECT [PrimaryKey], [ForeignKey], [EMail], [First],
    [Last], [PhoneExt], [Title] FROM [Person];
9
```

Pour supprimer un commentaire en ligne/bloc, suivez les étapes suivantes :

1. Sélectionnez la partie de l'instruction qui est commentée.
2. Cliquez avec la touche de droite sur la partie sélectionnée et sélectionnez **Insérer/Supprimer Commentaire en bloc (ou ligne)** de le menu contextuel.

Vous pouvez également supprimer les caractères de commentaire manuellement.

### Ajouter signets

Les signets sont utilisés pour marquer des items d'intérêt dans de scripts. Pour ajouter un signet, cliquez avec la touche de droite sur la ligne que vous souhaitez marquer et choisir **Insérer/Supprimer signet** depuis le menu contextuel. L'icône de signet est affichée dans la marge au début de la ligne ajoutée aux favoris.

Pour supprimer un signet, cliquez avec la touche de droite sur la ligne dont vous souhaitez ôter le signet et choisir **Insérer/Supprimer signet** depuis le menu contextuel. Pour supprimer tous les signets, cliquez avec la touche de droite de la souris n'importe où dans l'Éditeur SQL et sélectionnez **Supprimer tous les Signets** du menu contextuel.

Pour sauter vers le signet suivant/précédent, cliquez avec la touche de droite sur la ligne pertinente et choisissez **Se rendre au signet prochain/précédent**.

### Insérez les régions SQL

Les Régions sont des sections de texte que vous marquez et déclarez comme unité afin de structurer vos scripts SQL. Les régions peuvent être agrandies et réduites pour afficher ou dissimuler des parties de scripts SQL. Il est également possible d'imbriquer des régions à l'intérieur d'autres régions. Lorsque vous insérez une région, une icône d'agrandissement/réduction et un commentaire `--region` sont insérés au-dessus du texte sélectionné. Vous pouvez modifier le nom d'une région en ajoutant un texte descriptif au commentaire `--region`. Le terme *région* ne doit pas être supprimé.

Pour ajouter une région, suivez les instructions ci-dessous :

1. Sélectionnez les instructions que vous souhaitez transformer en région.
2. Cliquez avec la touche de droite sur l'instruction sélectionnée et sélectionnez **Insert Region** depuis le menu contextuel. Un exemple d'une région est illustré ci-dessous.

```
1
2     SELECT [PrimaryKey], [ForeignKey], [city], [state],
      [street], [zip] FROM [Address];
3     -- region
4     SELECT [PrimaryKey], [Name] FROM [Altova];
5     SELECT [PrimaryKey], [ForeignKey], [Name] FROM
      [Department];
6     -- endregion
7     SELECT [PrimaryKey], [ForeignKey], [Desc], [EMail],
      [Established], [Fax], [Name], [Phone] FROM [Office];
8     SELECT [PrimaryKey], [ForeignKey], [EMail], [First],
      [Last], [PhoneExt], [Title] FROM [Person];
9
```

Pour retirer une région, supprimez les deux délimiteurs `-- region` et `-- endregion`.

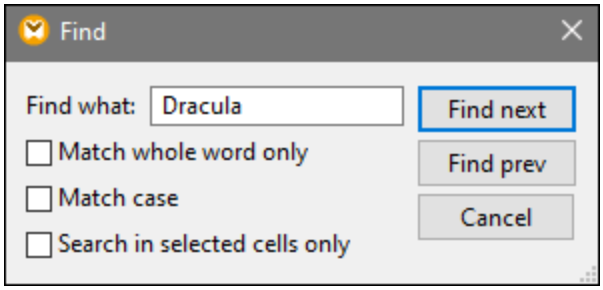
## 4.2.4.3 Onglet de Résultats

L'onglet Résultats du volet Requête BD affiche le « recordset » extrait comme résultat de la requête de base de données (*capture d'écran ci-dessous*).

| | BookID | Title | AuthorID | ISBN | Publisher | NumPages | Year | Genre | Price |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | Misery | 1 | 1501143107 | Scribner | 368 | 2016 | Horror | 11.99 |
| 2 | 2 | Nightblind | 2 | 9781910633113 | Orenda Books | 231 | 2016 | Crime & Mystery | 9.99 |
| 3 | 3 | Blackout | 2 | 1910633461 | Orenda Books | 276 | 2016 | Crime & Mystery | 8.49 |
| 4 | 4 | Outsider | 1 | 1501180983 | Scribner | 576 | 2018 | Horror | 12.79 |
| 5 | 5 | Dracula | 3 | 9781435142817 | Barnes & Noble | 512 | 2013 | Classics | 13.69 |
| 6 | 6 | The Mystery of Edwin Drood | 4 | 9781400043286 | Everyman's Library | 336 | 2004 | Classics | 19.79 |
| 7 | 7 | Crime and Punishment | 5 | 0679420290 | Everyman's Library | 608 | 1993 | Classics | 14.99 |
| 8 | 8 | Wuthering Heights | 6 | 979-8469527794 | Independently published | 200 | 2021 | Classics | 22.99 |
| 9 | 9 | Dune (Dune Chronicles, Book 1) | 7 | 0441013597 | Penguin Publishing Group | 704 | 2005 | Sci-Fi | 14.99 |

*Boutons de barre d'outils pour la navigation*

Au niveau de l'onglet Résultats, il existe deux boutons de barre d'outils qui vous permettent de naviguer les résultats de requête. Le bouton de la barre d'outils  (**Rechercher**) vous permet d'exécuter une recherche dans les résultats extraits. Quand vous appuyez sur ce bouton, le dialogue **Rechercher** apparaît (*capture d'écran ci-dessous*). La boîte de dialogue vous permet de configurer divers paramètres de recherche (ex., correspondre au mot entier). Utilisez les boutons **Recherche suivant** et **Recherche préc.** Pour basculer entre les occurrences du terme de recherche.




La commande de la barre d'outils  (**Aller à l'instruction**) passe à l'Éditeur SQL et marque l'instruction SQL qui produit le résultat actuel. Cela peut être particulièrement utile lorsque l'Éditeur SQL contient plusieurs instructions.

*Sélectionner les données*

Pour sélectionner les données dans les résultats de recherche, vous pouvez utiliser les méthodes suivantes :

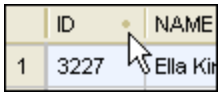
- Cliquez sur une des en-têtes de colonne pour sélectionner toute la colonne.
- Vous pouvez cliquer sur un nombre de lignes pour sélectionner toute la ligne.
- Vous pouvez appuyer sur et maintenir appuyé la clé **Shift** pour sélectionner une portée de cellules.
- Vous pouvez cliquer avec la touche de droite sur une cellule d'intérêt et sélectionnez diverses options depuis le menu contextuel : **Sélection | [Ligne | Colonne | Tout]**.
- Vous pouvez aussi sélectionner toutes les cellules en cliquant quelque part à l'intérieur de la table et en appuyant **Ctrl + A**.

*Copier des données*

Pour copier des cellules sélectionnées, appuyez sur **Ctrl + C**. En alternative, cliquez avec la touche de droite sur les cellules sélectionnées et choisissez **Copier cellules sélectionnées** ou **Copier cellules sélectionnées avec l'en-tête** depuis le menu contextuel.

*Trier les données*
Pour trier les données, vous pouvez choisir une des options suivantes :

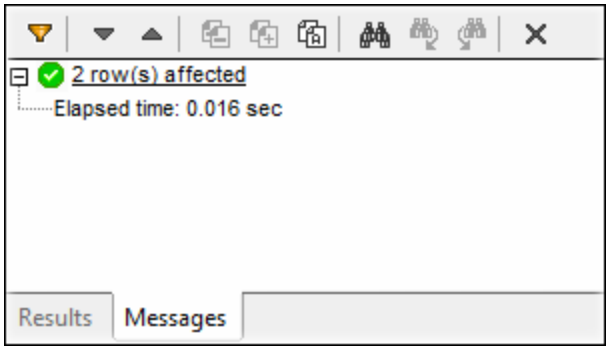
- Cliquez avec la touche de droite n'importe où dans la colonne d'intérêt et sélectionnez **Tri | Restaurer | Descendant** du menu contextuel.
- Cliquez l'icône point (*capture d'écran ci-dessous*) dans l'en-tête de colonne jusqu'à ce que vous ayez sélectionné l'option pertinente : la flèche pointant vers le haut trie les données par ordre ascendant, la flèche pointant vers le bas trie les données par ordre descendant.




Pour restaurer l'ordre de tri par défaut, cliquez avec la touche de droite n'importe où dans la table et choisir **Tri | Restaurer défaut** depuis le menu contextuel.

## 4.2.4.4 Onglet de message

L'onglet Messages de la Requête BD fournit une information sur la/les dernière/s instruction/s SQL exécutée/s et peut afficher les erreurs et avertissements.




*Boutons de barre d'outils.*
Vous pouvez utiliser les différents filtres pour personnaliser l'affichage de l'onglet Messages Les touches situées dans la partie supérieure de l'onglet Messages (*capture d'écran ci-dessus*) sont utilisées pour parcourir les messages, copier du texte dans le presse-papiers et dissimuler certaines parties du message. Quand vous cliquez avec la touche de droite quelque part à l'intérieur de l'onglet Messages, vous serez à même de voir ces options dans le menu contextuel.

| | |
|---|---|
|  | Le bouton **Filtrer** ouvre un menu dans lequel vous pouvez sélectionner les types de messages que vous souhaitez<br>voir dans l'onglet Messages. Les options suivantes sont possibles : *Check All*, *Uncheck All*, *Summary*, *Success*, *Warning*, *Error*, *Autoinsertion* et *Progress*.<br><br>L'option *Auto-insertion* se réfère à des messages qui apparaissent quand les instructions SQL ou les constructions SQL sont insérées automatiquement dans l'Éditeur SQL. L'option *Progress* vous informe sur le résultat de la connexion de base de données ainsi que sur le résultat du parsing SQL et du chargement de la structure des données. |

| | |
|---|---|
|  | Le bouton **Suivant** passe au message suivant et le marque. |
|  | Le bouton **Précédent** passe au message précédent et le marque. |
|  | **Copier le message choisi dans le presse-papiers** |
|  | **Copier le message choisi y compris ses enfants dans le presse-papiers** |
|  | **Copier tous les messages dans le presse-papiers** |
|  | La commande **Recherche** ouvre le dialogue **Trouver** qui vous permet de configurer le critère de recherche. |
|  | La commande **Trouver précédent** passe à l'occurrence précédente du string de recherche spécifié dans le dialogue **Trouver**. |
|  | La commande **Trouver suivant** passe à l'occurrence suivante du string de recherche spécifié dans le dialogue **Trouver**. |
|  | Le bouton **Supprime** supprime tous les messages depuis l'onglet Message de la fenêtre de l'Éditeur SQL. |

## 4.2.5 Mapper des données XML de/vers des champs BD

MapForce vous permet de mapper les données vers ou depuis des champs de base de données (colonnes) qui stockent du contenu XML. Ceci signifie que les données XML stockées par le champ de base de données (colonne) peuvent être extraites et écrites dans toute autre structure de appuyée par MapForce, et vice versa. Vous pouvez mapper des données comme suit :

1. Depuis ou vers les champs d'un type de XML dédié (par exemple, `Xml` dans SQL Server, `XMLType` dans Oracle). Lire ou écrire XML vers/de des champs XML dédiés est applicable aux bases de données qui ont une prise en charge native pour XML (tel que IBM DB2, Oracle et SQL Server).
2. Vers ou depuis les champs de texte stockant du contenu XML (par exemple, `Texte, Varchar`). Dans tous les cas, un schéma XML valide doit exister pour chaque colonne de base de données de/vers laquelle vous souhaitez mapper des données.

Dans les deux cas, un schéma XML valide doit exister pour chaque colonne vers/depuis vous voulez mapper des données. Lorsque du XML est stocké en tant que champ de string dans une base de données, l'encodage de caractère du document XML est celui du champ de string sous-jacent. Vous pouvez assigner un schéma XML par colonne de base de données. Si le schéma contient de mutliples éléments racine, vous pouvez sélectionner un élément de racine de ce schéma.

Quand XML est stocké comme champ de string dans la base de données, l'encodage de caractère du document XML est celui du champ de string sous-jacent. S'il est pris en charge par la base de données, il est convenu que la déclaration d'encodage du document XML sera celle déclarée dans le champ XML.

Pour plus d'informations concernant la prise en charge d'encodage XML fournie par diverses bases de données, veuillez vous référer à leur documentation. Si pris en charge par la base de données, la déclaration d'encodage du document XML est assumée être celle qui est déclarée dans le champ XML. Pour toute information sur la

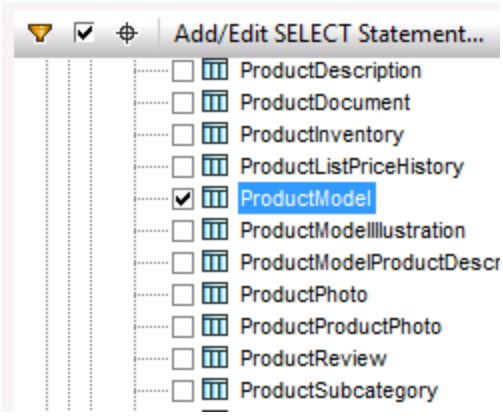
prise en charge de l'encodage XML fournie par les différentes bases de données, référez-vous à leur documentation.

### 4.2.5.1 Attribuer un Schéma XML dans un champ de base de données

Cette rubrique illustre comment attribuer un schéma dans un champ défini nativement en tant que type XML dans la base de données. Les instructions ci-dessous utilisent SQL Server 2014 et l'Adventure Works 2014 de base de données. Ce dernier peut être télécharger depuis la page des échantillons AdventureWorks sur GitHub (https://github.com/Microsoft/sql-server-samples/releases/tag/adventureworks). Veuillez noter que le mappage des données de ou vers les champs XML fonctionne de la même manière qu'avec d'autres types de base de données qui prennent en charge les champs XML.

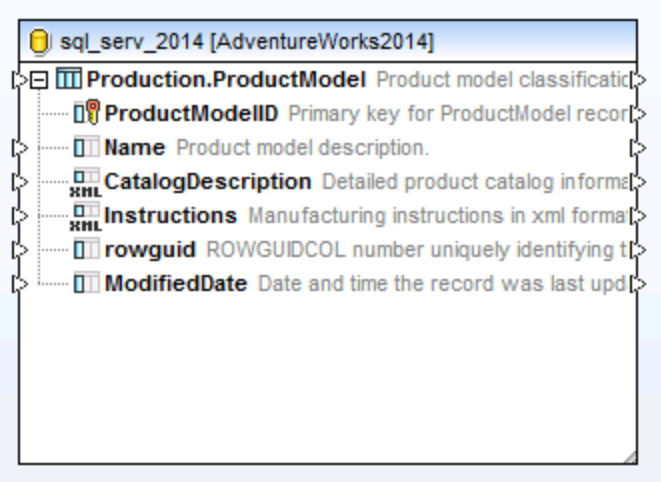
**Pour ajouter la base de données Adventure Works 2014 en tant que composant de mappage :**

1. Dans le menu **Insérer**, cliquer sur la **Base de données**, et suivre l'assistant pour se connecter à la base de données en utilisant votre méthode préférée (ADO ou ODBC). Pour plus d'informations, voir Connexion à Microsoft SQL Server (ADO) 168 et Connexion à Microsoft SQL Server (ODBC) 181. NOTE : si vous utilisez le pilote **SQL Server Native Client**, vous pouvez souhaitez définir la propriété **Integrated Security** dans un espace de caractère (voir Configurer les propriétés de liaison de données SQL Server 172 ).
2. Dans le dialogue **Insérer l'objet de base de données**, agrandir le schéma **Production**, puis sélectionner la table **ProductModel**.

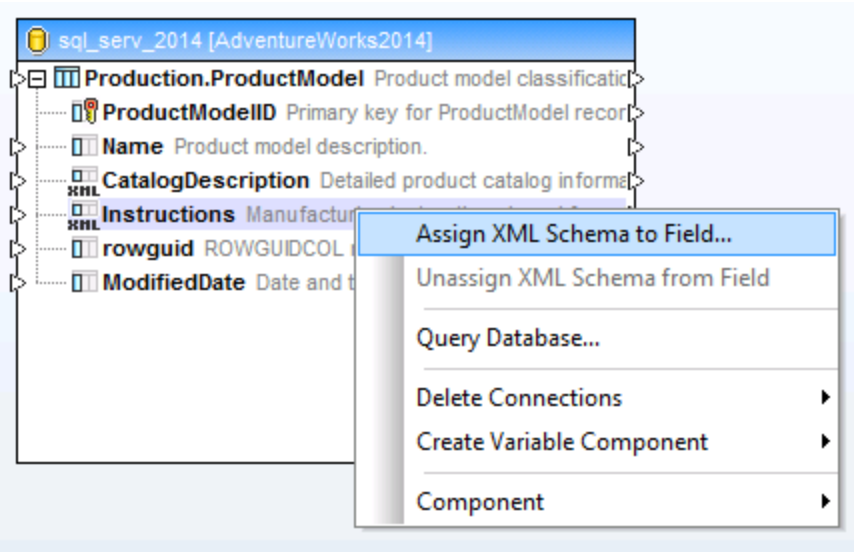



3. Cliquer sur OK.

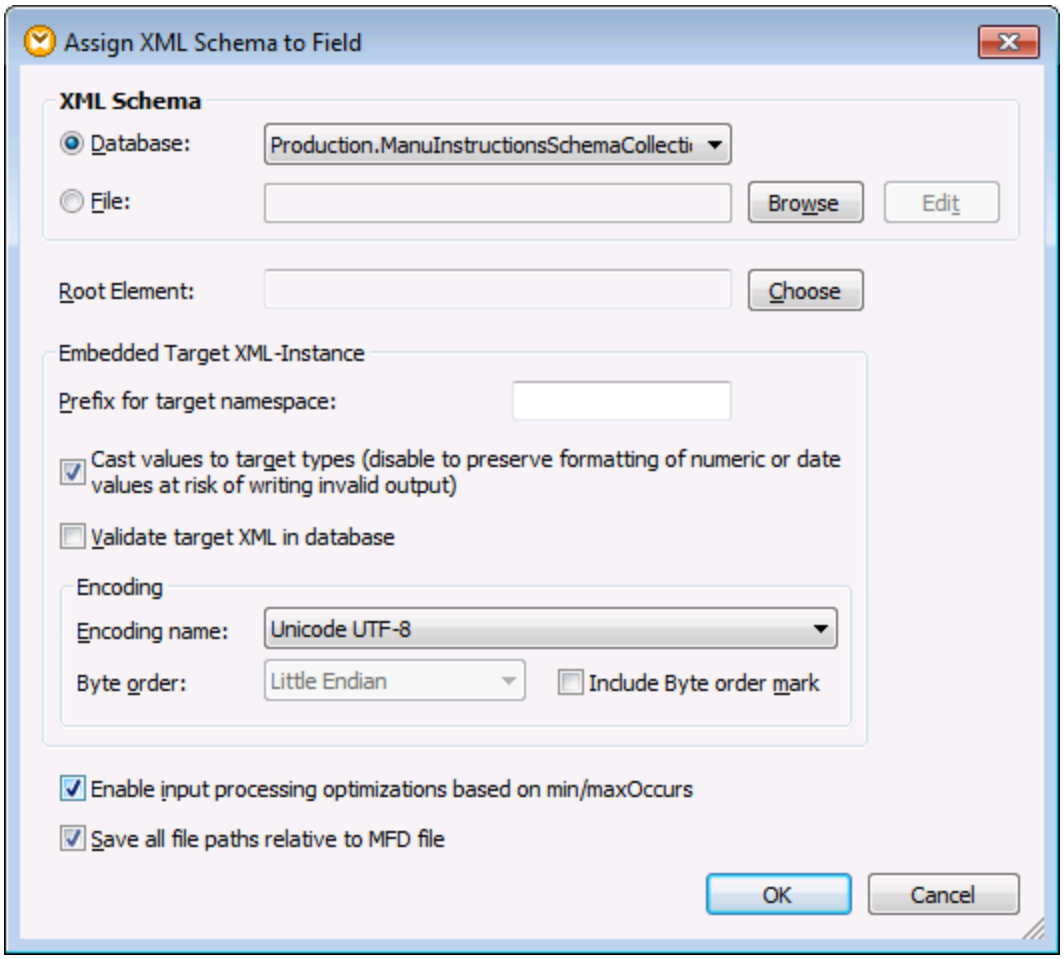
La table de base de données a maintenant été ajoutée à la surface de mappage. Veuillez noter que cette table a deux champs de type XML : **CatalogDescription** et **Instructions**:



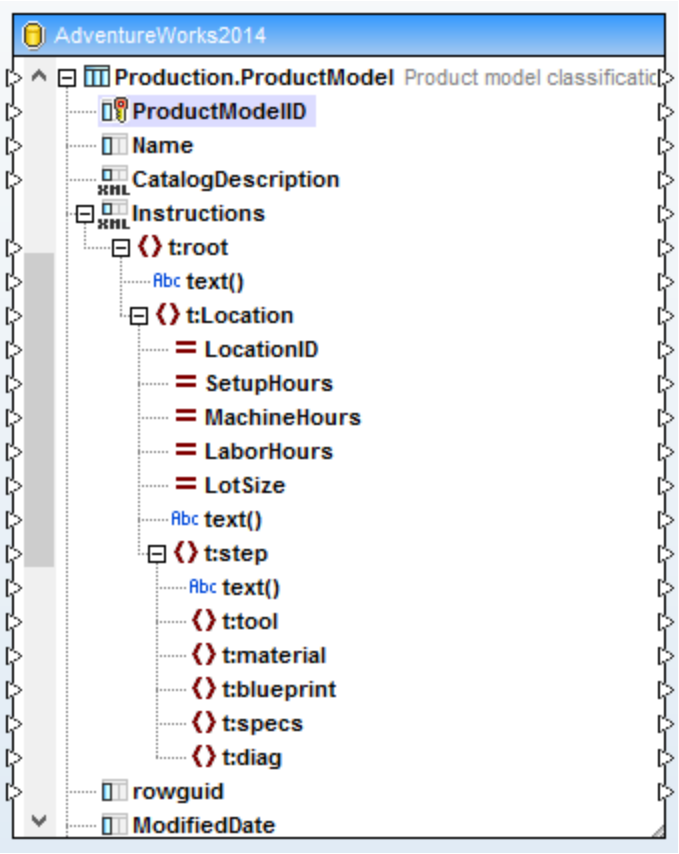

Pour que la structure des champs XML apparaissent dans le mappage, le schéma XML du contenu de champ est requis. Cliquer avec la touche de droite sur le champ **Instructions** et choisir **Attribuer Schéma XML dans le champ** depuis le menu contextuel.




Dans cet exemple particulier, vous allez attribuer un schéma au champ **Instructions** directement depuis la base de données. Pour ce faire, choisir l'item **Production.ManuInstructionsSchemaCollection** à côté de l'option **Base de données**, puis cliquer sur OK.




La structure du champ XML apparaît maintenant dans le composant. Vous pouvez maintenant établir des connexions (et mapper des données) de ou vers ce champ.




## 4.2.5.2 Exemple : Écrire des données XML dans un champ SQLite

Cet exemple vous guide à travers les étapes nécessaires pour créer un mappage de MapForce qui lit des données provenant de plusieurs fichiers XML et qui les rédige dans une base de données SQLite. L'objectif du mappage est de créer, pour chaque fichier de source XML, un nouvel enregistrement de base de données dans la base de données SQLite. Chaque enregistrement stockera le document XML en tant qu'un champ TEXTE.

Tous les fichiers utilisés dans cet exemple sont disponibles dans le chemin suivant : **<Documents>\Altova\MapForce2024\MapForceExamples\Tutorial**. Les noms de fichier sont comme suit :

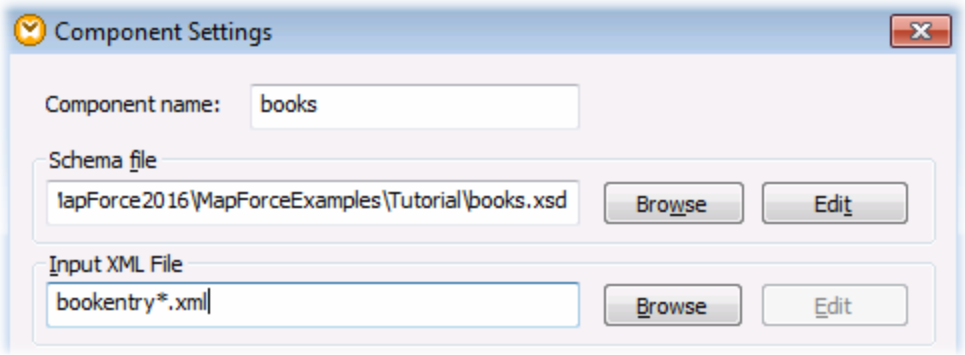
| | |
|---|---|
| Le fichier de design de mappage | • **XmlToSqliteField.mfd** |
| Les fichiers XML de source | • **bookentry1.xml**<br>• **bookentry2.xml**<br>• **bookentry3.xml** |
| Le schéma XML utilisé pour validation | • **books.xsd** |
| La base de données SQLite cible | • **Library.sqlite** |

Pour atteindre l'objectif du mappage, les mesures suivantes seront prises :

1. Ajouter le composant XML et le configurer pour lire depuis plusieurs fichiers.
2. Ajouter le composant de la base de données SQLite et attribuer un schéma XML dans le champ TEXTE cible.
3. Créer les connexions de mappage et configure l'action INSERT de la base de données.

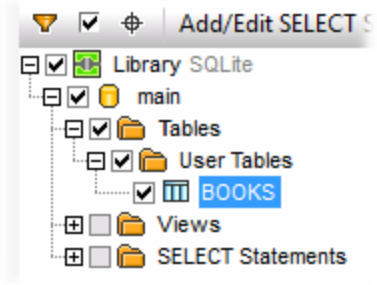
## Étape 1: Ajouter le composant XML

1. Dans le menu **Insérer**, cliquer sur **Schéma XML/Fichier** et chercher le schéma **books.xsd** situé dans le répertoire **<Documents>\Altova\MapForce2024\MapForceExamples\Tutorial**. Lorsque vous êtes invité à fournir un fichier XML échantillon, cliquer sur **Sauter**. Lorsque vous êtes invité à choisir un élément racine, choisir **Books**.
2. Double-cliquer sur l'en-tête du composant et saisir **bookentry*.xml** dans le champ **Fichier XML d'entrée**. Cela indique à MapForce de lire tous les fichiers XML dont le nom commence avec "bookentry-" dans le répertoire de source. Pour plus d'informations concernant cette technique, voir Traiter plusieurs fichiers d'entrée ou de sortie dynamiquement 787.

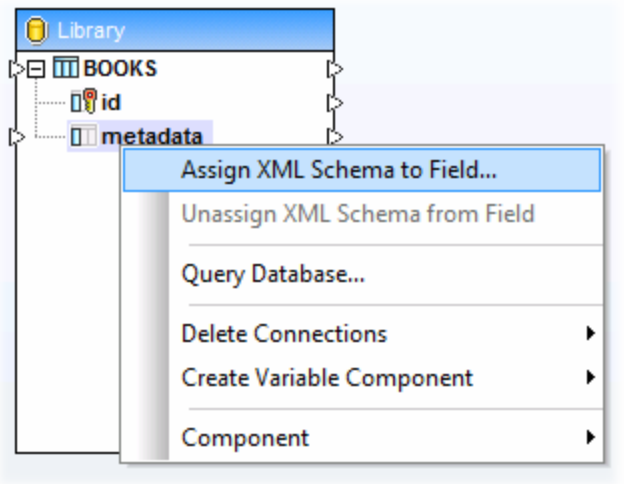



## Étape 2: Ajouter le composant SQLite

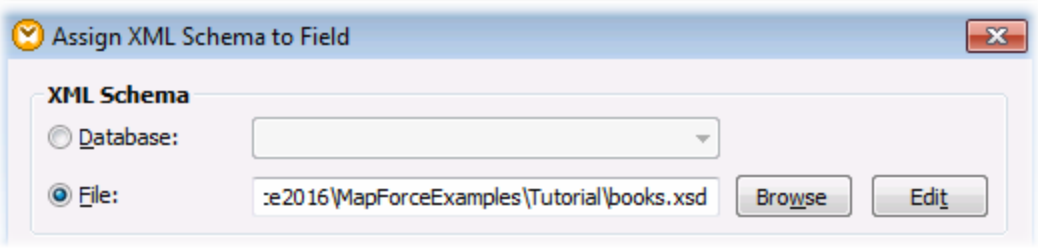
Dans le menu **Insérer**, cliquer sur **Base de données**, et suivre l'assistant pour se connecter au fichier de base de données **Library.sqlite** provenant du répertoire **<Documents>\Altova\MapForce2024\MapForceExamples\Tutorial** (voir aussi Connexion à une base de données SQLite existante 191 ). Lorsque vous êtes invité à choisir les objets de base de données, choisir la table BOOKS.



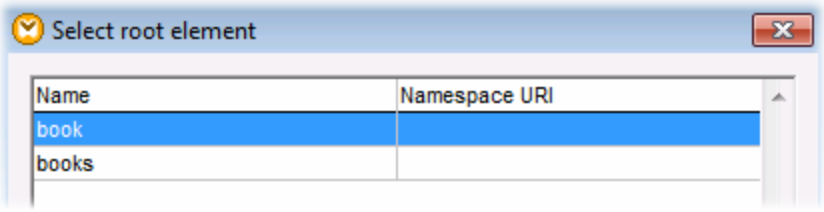

Le champ de base de données où le contenu XML sera écrit s'appelle `metadata`. Pour attribuer un schéma XML à ce champ, cliquer avec la touche de droite et sélectionner **Attribuer Schéma XML dans le champ** depuis le menu contextuel.




Dans ce tutoriel, le schéma attribué au champ `metadata` est le même que celui utilisé pour valider les fichiers XML de source. Cliquer sur **Chercher** et choisir le schéma **books.xsd** depuis le répertoire **<Documents>\Altova\MapForce2024\MapForceExamples\Tutorial** :

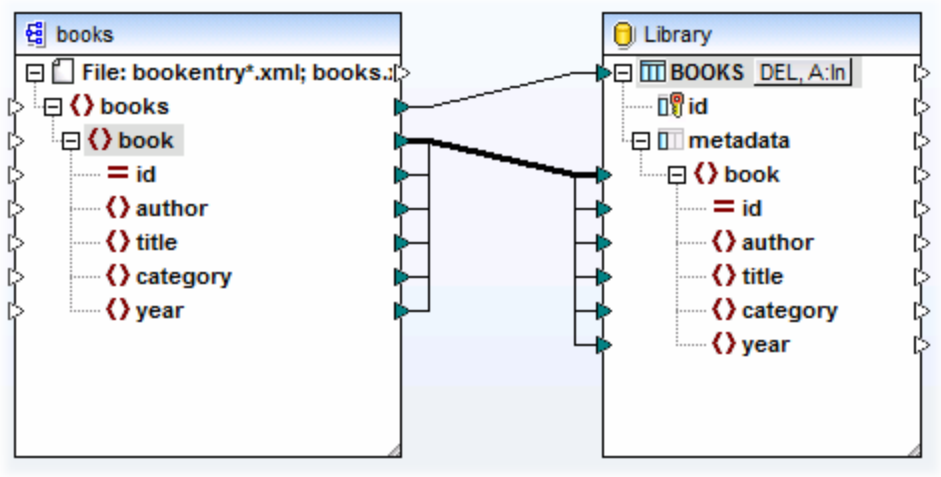



Le schéma **books.xsd** possède deux éléments avec une déclaration globale : `book` et `books`. Dans cet exemple, nous allons définir `book` en tant que l'élément racine du XML écrit dans le champ de base de données. Cliquer sur **Choisir**, et sélectionner `book` en tant qu'élément racine :



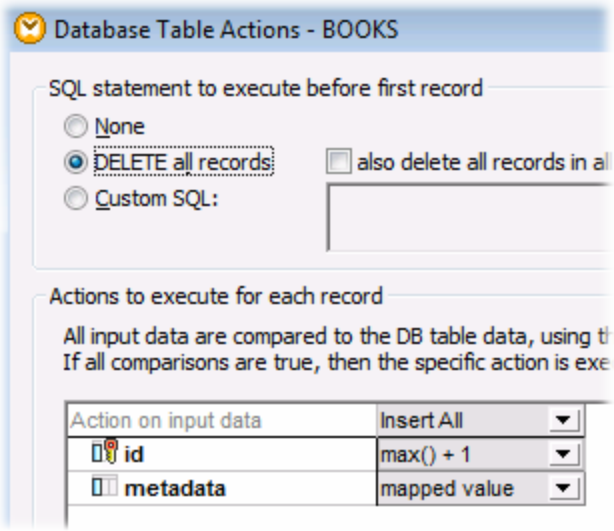

## Étape 3: Créer les connexions de mappage et configurer l'action de base de données INSERT

Créer les connexions de mappage comme suit :




Comme indiqué ci-dessus, la connexion provenant de `book` vers `book` est une connexion "Copy-All", étant donné qu'aussi bien la source et la cible utilisent le même schéma et que les noms des éléments enfants sont les mêmes. Pour plus d'informations concernant ce type de connexions, voir connexions Copy-all 60.

La connexion supérieure (`books` vers `BOOKS`) itère par chaque élément de livre dans la source et écrit un nouvel enregistrement dans la table BOOKS. Cliquer sur la touche `A:In` sur le composant de base de données et définit les paramètres de mise à jour de la base de données comme affiché ci-dessous :




L'option **SUPPRIMER tous les enregistrements** instruit MapForce à supprimer les contenus de la table `BOOKS` avant d'insérer des enregistrements.

Les actions **Insérer toutes** spécifient qu'une requête INSERT de base de données aura lieu. Le champ `id` est généré depuis la base de données elle-même, alors que le champ `metadata` sera rempli avec la valeur fournie par le mappage.

Veiller à enregistrer le mappage avant de l'exécuter.

Pour exécuter le mappage et consulter l'entrée générée, cliquer sur l'onglet **Sortie**. Veuillez noter que cette action ne met pas à jour la base de données immédiatement. Lorsque vous êtes prêt à exécuter le script de base de données généré, choisir la commande de menu **Sortie | Exécuter Script SQL** (ou cliquer sur la touche  dans la barre d'outils).

## 4.2.5.3 Exemple: Extraire des données depuis les colonnes de type XML IBM DB2

Cet exemple montre comment extraire des données depuis des colonnes de base de données IBM DB2 de type XML et les écrire dans un fichier CSV cible. Il illustre également comment utiliser des instructions XQuery intégrées dans SQL afin d'extraire du contenu XML de manière conditionnelle. L'exemple requiert un accès à une base de données IBM DB2 pour laquelle vous avez la permission de créer et de remplir des tables.

Premièrement, il s'agit de préparer la base de données afin qu'elle contienne réellement des données XML. Pour ce faire, soit utiliser un outils d'administration de base de données spécifique à votre base de données, soit directement dans MapForce. Pour ce faire, suivre les étapes suivantes :

1. Créer un nouveau mappage et cliquer sur l'onglet **DBQuery**.
2. Cliquer sur **Connexion rapide** ( ) et suivre les instructions de l'assistant pour créer une nouvelle connexion de base de données (voir aussi Database Connection Examples 193).
3. Coller le texte suivant dans l'éditeur SQL. Cette requête SQL crée une table de base de données appelée `ARTICLES` et la remplit avec des données.

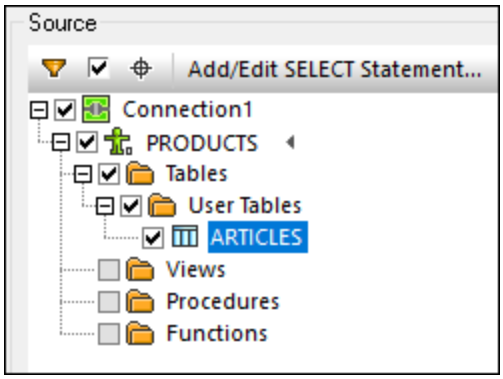
```
-- Create the table
CREATE TABLE
    ARTICLES (
        id INTEGER NOT NULL,
        article XML ) ;
-- Populate the table
INSERT INTO ARTICLES VALUES
   (1, '<Article>
      <Number>1</Number>
      <Name>T-Shirt</Name>
      <SinglePrice>25</SinglePrice>
   </Article>'),
(2, '<Article>
      <Number>2</Number>
      <Name>Socks</Name>
      <SinglePrice>230</SinglePrice>
   </Article>'),
(3, '<Article>
      <Number>3</Number>
      <Name>Pants</Name>
      <SinglePrice>34</SinglePrice>
   </Article>'),
 (4, '<Article>
      <Number>4</Number>
      <Name>Jacket</Name>
      <SinglePrice>5750</SinglePrice>
   </Article>');
```

4. Cliquer sur la touche **Exécuter** ( ). Le résultat d'exécution de la requête est affiché dans la fenêtre Résultats de requête. Si la requête est exécutée correctement, quatre lignes sont ajoutées à la table nouvellement créée.

Ensuite, nous allons créer un mappage qui extrait des données XML depuis la table `ARTICLES` créées ci-dessus de manière conditionnelle. L'objectif est d'extraire depuis la colonne `ARTICLES` uniquement les articles dont le prix est supérieur à 100.

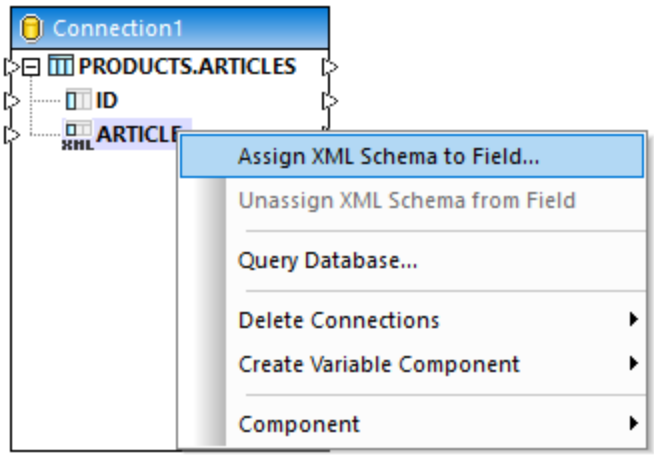
## Étape 1: Ajouter la base de données

1. Cliquer sur l'onglet **Mappage** pour retourner au volet de mappage.
2. Dans le menu **Insérer**, cliquer sur **base de données**, et suivre les instructions de l'assistant pour se connecter à la base de données.
3. Lorsque vous êtes invité à choisir les objets de la base de données, choisir la table `ARTICLES` créée précédemment.




## Étape 2: Attribuer le schéma au champ de type XML

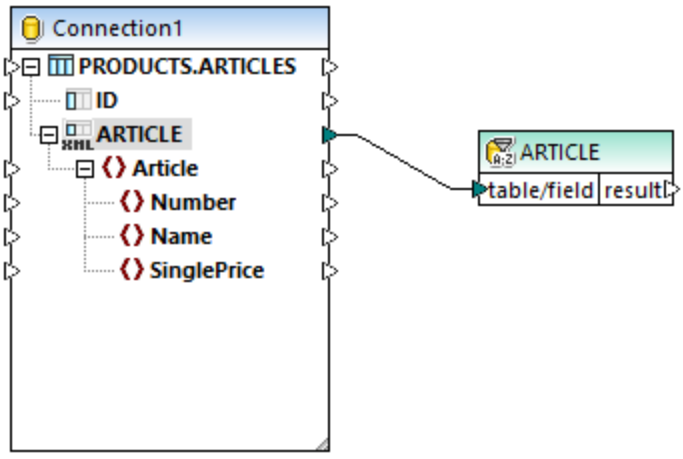
1. Cliquer avec la touche de droite sur l'item `ARTICLE` du composant, et choisir **Attribuer Schéma XML au champ** depuis le menu contextuel.




2. Choisir **Fichier**, et chercher le schéma suivant : **<Documents>\Altova\MapForce2024\MapForceExamples\Tutorial\DB2xsd.xsd**.

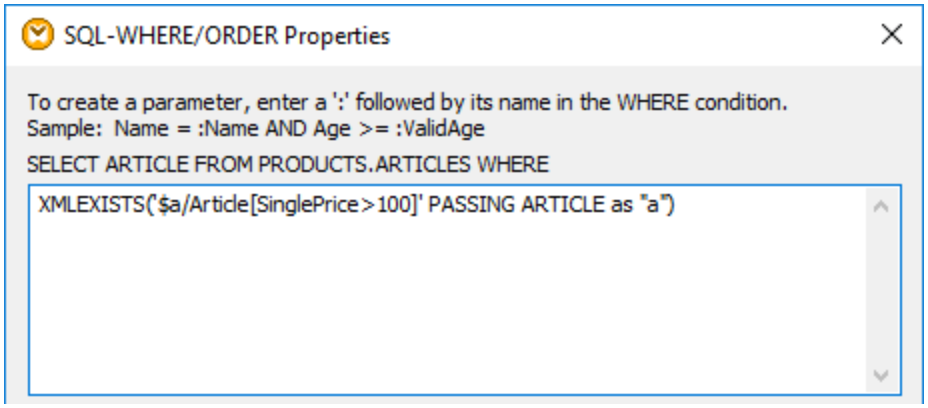
## Étape 3: Ajouter le composant WHERE/ORDER SQL

1. Dans le menu **Insérer**, cliquer sur **WHERE/ORDER SQL**.
2. Connecter la colonne `ARTICLE` de type XML à l'entrée du WHERE/ORDER SQL.




3. Dans le dialogue Propriétés WHERE/ORDER SQL, saisie le texte suivant :

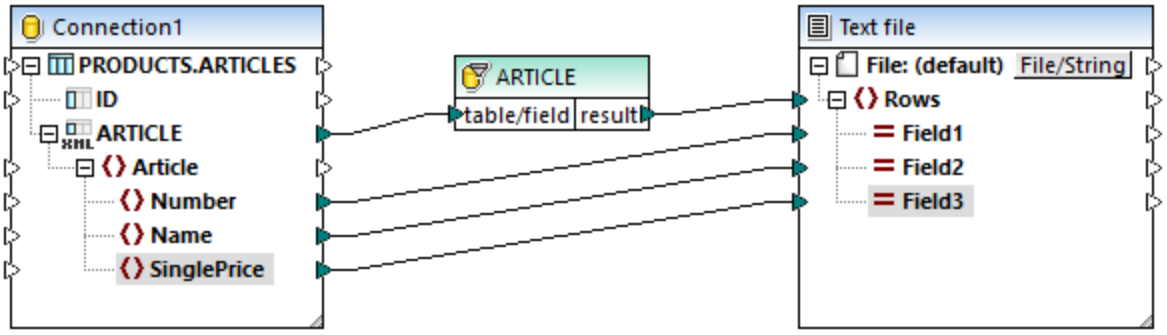
```
XMLEXISTS('$a/Article[SinglePrice>100]' PASSING ARTICLE as "a")
```




Le texte ci-dessus représente la partie "WHERE" de la requête SQL. Lors de l'exécution du mappage, cette partie sera combinées avec la partie "SELECT" affichée dans le dialogue. Cette instruction utilise la fonction `XMLEXISTS` et la syntaxe spécifiques aux bases de données IBM DB2.

## Étape 4: Ajouter le fichier CSV cible

1. Dans le menu **Insérer**, cliquer sur **Fichier texte**.
2. Sur invitation, choisir **Utiliser traitement simple pour CSV standard...** , et cliquer sur **Continuer**.
3. Cliquer sur **Apposer champ** trois fois pour ajouter trois champs qui stockeront le numéro, le nom et le prix de l'article. Ne pas toucher aux autres paramètres.
4. Tirer les connexions de mappage comme indiqué ci-dessous.




Vous pouvez maintenant consulter un aperçu du mappage en cliquant sur l'onglet **Sortie**. Comme prévu, seuls des articles dont le prix est supérieur à 100 sont affichés dans la sortie.

```
1    2,Socks,230
2    4,Jacket,5750
```

## 4.2.6 Procédures stockées

Les procédures stockées sont des programmes qui sont hébergés et exécutés sur un serveur de base de données. Les procédures stockées peuvent être appelées par des applications client et elles sont souvent écrites en un dialecte élargi de SQL. Certaines bases de données prennent aussi en charge des mises en place dans Java, .NET CLR, ou d'autres langages de programmation.

Des utilisations typiques de procédures stockées incluent l'interrogation d'une base de données et le renvoi de données au client appelant, ou la réalisation de modifications à la base de données après une validation supplémentaire des paramètres d'entrée. Les procédures stockées peuvent également réalisées d'autres actions à l'extérieur de la base de données, telle qu'envoyer des e-mails.

Une procédure stockée peut avoir zéro ou plusieurs paramètres d'entrée et de sortie, et peut retourner en option zéro ou plusieurs recordsets, en plus de la valeur retournée par défaut. Par conséquent, dans MapForce, vous pouvez appeler une procédure stockée de plusieurs manières :

- Appeler une procédure stockée pour obtenir des données, comme s'il s'agissait d'un composant de source dans le mappage. Ceci s'applique pour des procédures qui ne prennent pas des paramètres d'entrée. Lorsque le mappage est exécuté, la procédure est appelée, et elle retourne certains recordset ou paramètres de sortie. Vous pouvez mapper recordset, ou les paramètres de sortie, ou les deux, vers tout autre type de données pris en charge par MapForce. Pour consulter un exemple, voir Procédures stockées en tant que source de données 327.
- Appeler une procédure stockée comme appel function-like, avec des paramètres. Dans ce cas, vous apportez tous les paramètres d'entrée requis depuis le mappage, et vous pouvez aussi mapper le recordset retourné, ou les paramètres de sortie, ou les deux, vers une autre cible prise en charge par MapForce. Pour consulter un exemple, voir Procédures stockées avec Entrée et Sortie 330.
- Appeler une procédure stockée comme s'il s'agissait d'un composant cible sur le mappage. Le cas d'utilisation typique est d'appeler une procédure stockée avec des paramètres afin de modifier la base de données (par exemple, insérer un enregistrement). Cette approche est convenable si vous ne nécessitez pas de sortie provenant de la procédure stockée. De même, dans cette approche, vous pouvez exécuter la procédure stockée dans le cadre d'une transaction de base de données qui peut

être renversée en cas d'erreur. Pour un exemple, voir Procédures stockées dans les Composants cibles 334.

Il existe également des cas où vous devez appeler des procédures stockées ou réaliser des actions dans des tables de base de données dans un ordre spécifique (d'abord insérer, puis mettre à jour, etc.). Par exemple, si vous souhaitez passer le paramètre de sortie d'une procédure stockée dans une autre procédure stockée. Ou bien vous pouvez souhaiter combiner des données retournées par une procédure stockée avec des données provenant d'une table. Ce genre d'actions est possible avec l'aide des relations locales définies dans MapForce, même si la base de données sous-jacente n'impose pas de relations clés primaire/étrangère entre les tables. Pour voir un exemple, voir Procédures stockées et Relations locales 338.

**Note :** Pour illustrer comment MapForce met en place des procédures de stockage, ce chapitre utilise Microsoft SQL Server 2016 et la base de données "AdventureWorks2016". Cette dernière peut être téléchargée depuis le https://github.com/Microsoft/sql-server-samples/releases/tag/adventureworks.

## Notes de prise en charge

- Les procédures stockées peuvent être utilisées uniquement dans le moteur d'exécution BUILT-IN. La génération de code dans C++, C#, ou Java n'est pas prise en charge.
- Les types définis par l'utilisateur, les types de curseur, les types de variante et de nombreux autres types de données "exotiques" spécifiques à des bases de données (comme des tableaux, géométrie, types CLR) ne sont généralement pas pris en charge en tant que types de paramètres d'entrée ou de sortie.
- La surcharge de procédure et de fonction (plusieurs définitions de routines portant le même nom et des paramètres différents) n'est pas prise en charge.
- Certaines valeurs par défaut de prise en charge de bases de données dans les paramètres d'entrée, cela n'est pas pris en charge actuellement. Vous ne pouvez pas omettre des paramètres d'entrée dans le mappage pour utiliser la valeur par défaut.
- Les procédures stockées retournant des enregistrements multiples sont pris en charge selon la combinaison du pilote et de l'API de base de données (ODBC/ADO/ADO.NET/JDBC). Seules des procédures qui retournent le même nombre d'enregistrements avec une structure de colonne fixe sont prises en charge.
- Dès que possible, utiliser la dernière version du pilote natif de base de données maintenu par le fournisseur de base de données. Éviter d'utiliser des pilote de pont, comme ODBC vers ADO Bridge, ou ODBC vers JDBC Bridge.
- En option, vous pouvez activer des transactions de base de données pour les procédures stockées appelées en tant que source de données, voir Procédures stockées dans les Composants cibles 334. Les transactions ne sont pas prises en charge pour les procédures stockées appelées en tant que source de données (sans les paramètres d'entrée), ou celles qui sont appelées en tant qu'une fonction (avec l'entrée et la sortie).

La table suivante liste les notes de prises en charge spécifique à la base de données.

| Base de données | Notes de prise en charge |
|---|---|
| **Accès** | • Les procédures stockées dans les bases de données Microsoft Access disposent de fonctions très limitées et ne sont pas prises en charge dans MapForce. |
| **DB2** | • Prise en charge dans MapForce: procédures stockées, fonctions scalaires, fonctions table. |

| Base de données | Notes de prise en charge |
|---|---|
| | • Les valeurs de renvoi provenant des procédures de stockage de DB2 ne sont pas prises en charge parce qu'elles ne peuvent pas être lues par le biais des API de base de données utilisées dans MapForce.<br>• Les fonctions Row-valued (RETURNS ROW) ne sont pas prises en charge.<br>• Il est recommandé d'installer au moins « IBM_DB2 9.7 Fix Pack 3a » pour éviter un problème de pilote JDBC confirmé lors de la lecture des erreurs/avertissements après l'exécution. Cela permet aussi de régler un problème avec le fournisseur ADO qui entraîne la perte d'une ligne de résultat. |
| **Firebird** | • Prise en charge dans MapForce: procédures stockées, fonctions table-valued |
| **Informix** | • Prise en charge dans MapForce: procédures stockées, fonctions table-valued. |
| **MariaDB** | • Prise en charge dans MapForce: procédures stockées, fonctions scalaires |
| **MySQL** | • Prise en charge dans MapForce: procédures stockées, fonctions scalaires<br>• MySQL contient une prise en charge complète pour des procédures stockées et des fonction en commençant avec la version 5.5. Si vous utilisez une version précédente, les fonctions dans MapForce sont limitées. |
| **Oracle** | • Prise en charge dans MapForce: procédures stockées, fonctions scalaires, fonctions table. Cela contient des procédures stockées et des fonctions autonomes ainsi que celles définies dans un package Oracle.<br>• Il est recommandé d'utiliser un pilote Oracle natif au lieu de **Microsoft OLE DB Provider for Oracle**.<br>• Oracle a un moyen spécial de retourner les ensembles de résultat dans le client en utilisant des paramètres de sortie de type REF CURSOR. Cela est pris en charge par MapForce pour des procédures stockées, mais pas pour des fonctions. Les noms et le nombre des enregistrements sont donc toujours fixées pour des procédures stockées Oracle. |
| **PostgreSQL** | • Prise en charge dans MapForce: fonctions scalaires, fonctions row-valued, fonctions table-valued.<br>• Dans PostgreSQL, tous les paramètres de sortie définis dans une fonction décrivent les colonnes de l'ensemble de résultat. Cette information est utilisée automatiquement par MapForce - aucune détection par exécution ou entrée des enregistrements n'est nécessaire. Les paramètres de type `refcurseur` ne sont pas pris en charge. |
| **Progress OpenEdge** | • Prise en charge dans MapForce: procédures stockées. |
| **Serveur SQL** | • Prise en charge dans MapForce: procédures stockées, fonctions scalaires, fonctions table-valued.<br>• Il est recommandé d'utiliser le dernier pilote **SQL Server Native Client** au lieu de **Microsoft OLE DB Provider for SQL Server**.<br>• L'API d'ADO a une prise en charge limitée pour certains types de données introduits avec SQL Server 2008 (`datetime2`, `datetimeoffset`). Si vous rencontrez des problèmes de raccourcissement de données avec ces types temporels lorsque vous utilisez ADO avec le SQL Server Native Client, vous |

| Base de données | Notes de prise en charge |
|---|---|
| | pouvez définir l'argument de string de connexion `DataTypeCompatibility=80` ou utiliser ODBC.<br>• Les SQL Server Procedures ont un paramètre de retour implicite de type `int null`, qui est disponible pour le mappage. Si la procédure omet une instruction RETURN, la valeur résultante est 0. |
| **SQLite** | • SQLite n'utilise pas de procédures stockées. |
| **Teradata** | • Prise en charge dans MapForce: procédures stockées, macros.<br>• Fonctions scalaires, fonctions d'agrégat et fonctions de table ne sont pas prises en charge<br>• Problème connu : Le pilote Teradata ODBC refuse de remplir les valeurs de paramètre de sortie après un appel de procédure. |

### 4.2.6.1 Ajouter des procédures stockées dans le mappage

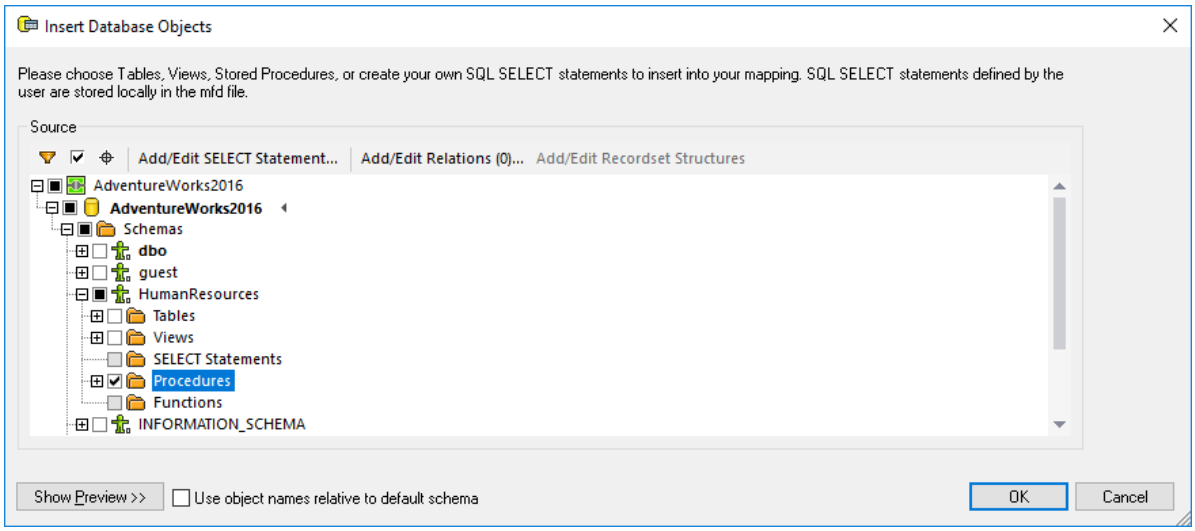
Dans la zone de mappage, les procédures stockées sont affichées en tant que partie intégrante du composant de données auquel elles appartiennent. Pour que les procédures stockées soient visibles sur le composant de la base de données, vous devez les choisir explicitement lorsque vous ajoutez le composant de base de données au mappage, comme indiqué ci-dessous. Dans cet exemple, nous nous connectons à la base de données "AdventureWorks" exécutée sur SQL Server. Les instructions sont semblables pour les autres types de base de données.

Dans le cas de base de données Oracle, des procédures stockées ou des fonctions peuvent être autonomes ou faire partie de packages Oracle. Vous pouvez ajouter les deux catégories au mappage. Les procédures stockées ou les fonctions appartenant à un package apparaissent sous le nom du package respectif dans le dialogue "Insérer des objets de base de données" illustré ci-dessous.

**Pour ajouter des procédures stockées au mappage :**

1. Choisir parmi les deux possibilités suivantes :
   - Dans le menu **Insérer**, cliquer sur **Base de données**.
   - Cliquer sur la touche the la barre d'outils **Insérer base de données** ( ).
2. Suivre les instructions de l'assistant de la base de données jusqu'à ce que vous parveniez au dialogue "Insérer Objets base de données". Pour consulter des instructions plus détaillées applicables aux différents types de base de données, voir Exemples de connexion à des bases de données 193.

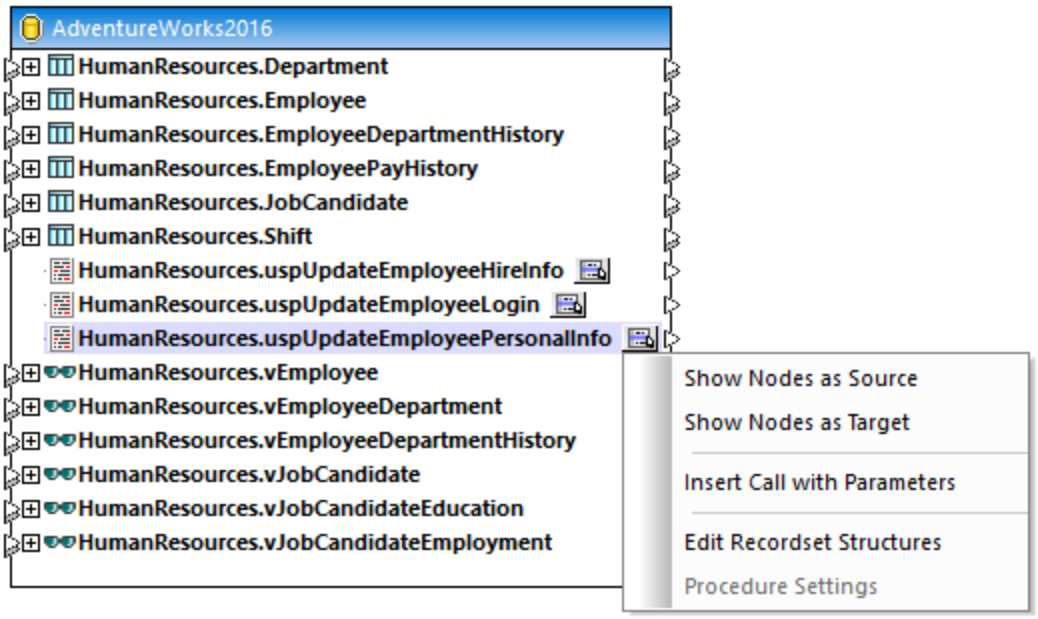



3. Choisir les cases à cocher situées à côté des objets de base de données que vous souhaitez voir affichées sur le mappage et cliquer sur **OK**. Dans cet exemple, nous avons choisi toutes les tables, modes et procédures stockées disponibles dans le schéma "HumanResources".

**Notes**

- Vous pouvez modifier les objets sélectionnés ultérieurement à tout moment, en cliquant avec la touche de droite sur la barre de titre d'un composant de base de données, et en choisissant **Ajouter/Supprimer/Éditer des objets de base de données** depuis le menu contextuel.
- Votre compte d'utilisateur de base de données doit disposer des droits pour consulter et exécuter des procédures stockées dans la base de données.

Le composant de base de données est maintenant ajouté au mappage. Veuillez noter que les procédures stockées sont identifiées par l'icône . De plus, les tables, modes et les procédures sont triées alphabétiquement dans le composant de base de données.




La touche **Afficher menu contextuel** située à côté de chaque procédure stockée vous permet de configurer la manière dont vous souhaitez appeler la procédure stockée, vous trouverez également d'autres paramètres liés à la procédure, comme indiqué ci-dessous :

| Option | Utilisation |
|---|---|
| **Afficher nœuds en tant que Source** | Choisir cette option si vous souhaitez appeler une procédure stockée *sans paramètres* afin d'extraire des données depuis une base de données et la mapper vers un autre composant pris en charge par MapForce (XML, texte, EDI, etc.). Pour consulter un exemple, voir Procédures stockées en tant que source de données 327. |
| **Afficher nœuds en tant que cible** | Choisir cette option si vous souhaitez appeler une procédure stockée pour modifier la base de données ou effectuer une autre action spécifique dans laquelle vous n'avez pas besoin de la sortie de la procédure stockée. Pour consulter un exemple, voir Procédures stockées dans composants de cible 334. |
| **Insérer appel avec Paramètres** | Choisir cette option si vous souhaitez appeler une procédure stockée *avec des paramètres* et que vous souhaitez mapper les données retournées dans un autre composant pris en charge par MapForce. Pour consulter un exemple, voir Procédures stockées avec paramètres d'entrée et de sortie 330. |
| **Éditer Structures de Recordset** | Applicable pour les procédures stockées qui retournent des recordsets. Choisir cette option pour exécuter la procédure stockée une fois, afin que MapForce puisse déterminer la structure du recordset retourné et l'afficher dans le mappage. En alternative, si vous ne souhaitez pas exécuter la procédure stockée au moment du design time, you can define the recordset structure manually. |

| | |
|---|---|
| **Paramètres de procédure** | Applicable uniquement pour les procédures stockées qui ont été configurées en tant que "target" (c'est à dire, celles qui mettent à jour la base de données). Choisir cette option pour configurer des paramètres supplémentaires liés à la procédure, comme exécuter un requête SQL personnalisée avant d'appeler la procédure, ou activer des transactions de base de données. |

## 4.2.6.2 Procédures stockées en tant que source de données

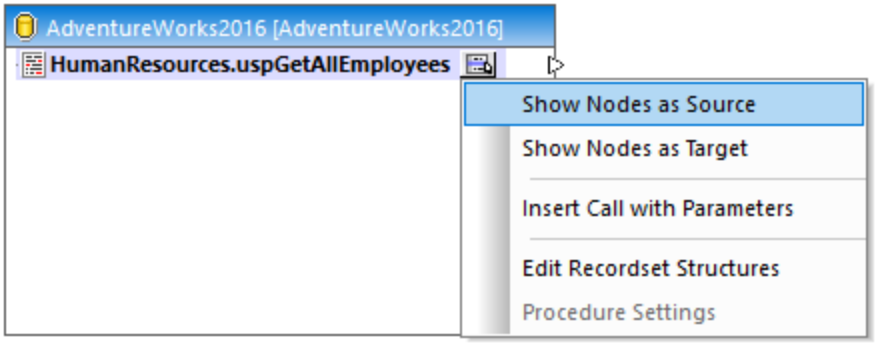
Cet exemple vous montre comment appeler une procédure qui ne prend pas de paramètres d'entrée et se limite à extraire des données de la base de données. Dans ce scénario, la procédure stockée se comporte comme un composant de source dans le mappage, et vous pouvez mapper des données extraites à tout autre composant cible pris en charge par MapForce. Si vous souhaitez appeler une procédure stockée avec des paramètres d'entrée, voir Procédures stockées avec Entrée et Sortie 330.

Créons tout d'abord la procédure stockée de démonstration dans la base de données "AdventureWorks". Pour ce faire, exécuter le script ci-dessous par rapport à la base de données. Vous pouvez ce faire à partir d'une fenêtre de requête de **Microsoft SQL Server Management Studio**, ou directement depuis l'onglet **Requête BD** de MapForce, voir Parcourir et interroger des bases de données 300. Dans tous les cas, s'assurer que votre compte d'utilisateur de base de données a la permission de créer des procédures stockées.

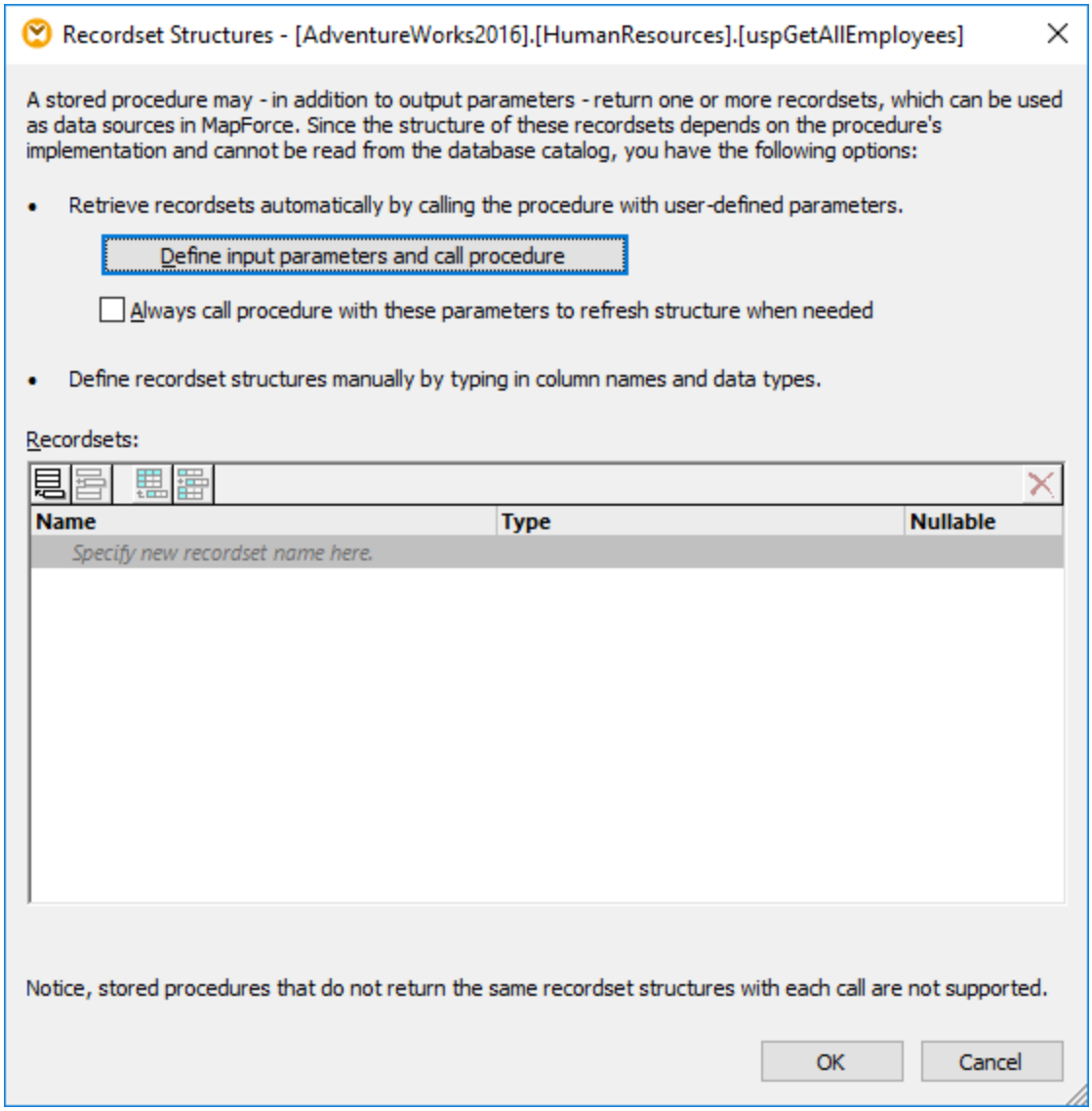
```
CREATE PROCEDURE HumanResources.uspGetAllEmployees
AS
    SELECT LastName, FirstName, JobTitle, Department
    FROM HumanResources.vEmployeeDepartment
```

La procédure stockée ci-dessus retourne des informations d'employés depuis le mode **vEmployeeDepartment**. Les étapes suivantes vous montrent comment créer un mappage qui consomme des données retournées par cette procédure.

1. Se connecter à la base de données "AdventureWorks" depuis MapForce, et ajouter la procédure stockée au mappage comme décrit dans la section Ajouter des procédures stockées dans le mappage 324. Veuillez vous assurer que votre compte utilisateur de base de données détienne la permission de consulter et d'exécuter des procédures stockées.
2. Cliquer sur la touche **Afficher menu contextuel**  située à côté de la procédure stockée et choisir **Afficher nœuds en tant que source**.

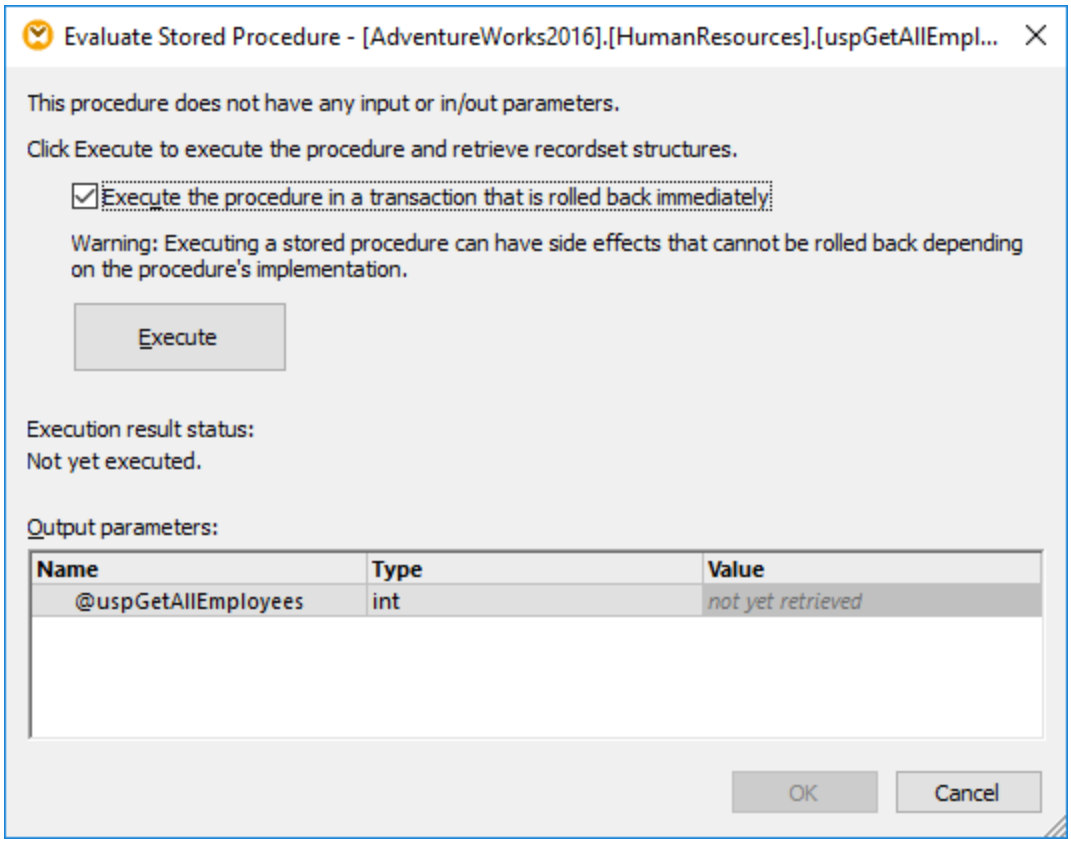


3. Cliquer à nouveau sur la touche **Afficher menu contextuel**, choisir **Éditer les structures de Recordset**. Le dialogue "Structures de Recordset" apparaît.

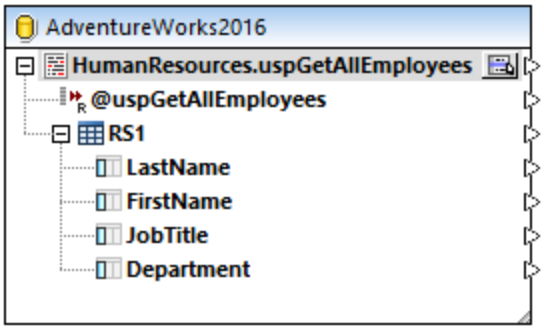



Le fait d'appeler une procédure stockée au moment de la conception peut présenter des effets secondaires (selon la mise en place de la procédure). Si vous ne souhaitez pas exécuter la procédure stockée au moment de la conception, ne pas cliquer sur **Exécuter**, comme décrit dans les étapes suivantes. Au lieu, définir le recordset attendu dans le dialogue "Structures de recordset", en ajoutant des recordsets et leurs colonnes associées manuellement. Utiliser les touches **Ajouter recordset** ou **Ajouter colonne** dans le dialogue "Structures de recordset".

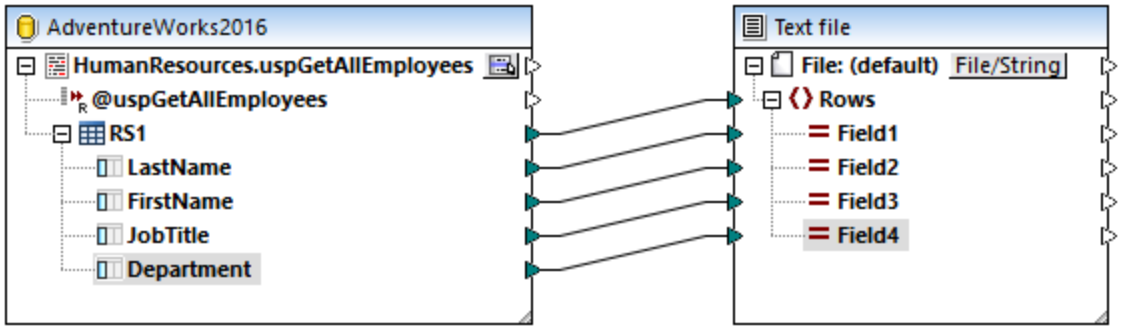
4. Cliquer sur **Définir les paramètres d'entrée et appeler la procédure**. Le dialogue "Évaluer procédure stockée"apparaît.



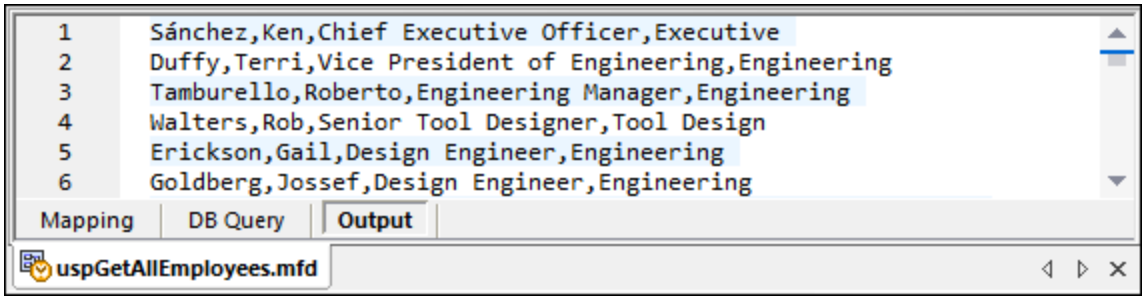

5. Cliquer sur **Exécuter**, puis sur **OK**. La structure de recordset ("RS1") est maintenant visible aussi bien dans le dialogue "Structures Recordset" que dans le mappage.




6. À ce niveau, vous pouvez ajouter un composant cible dans lequel les données extraites seront rédigées. Dans cet exemple, les données seront rédigées dans un fichier CSV. Dans le menu **Insérer**, cliquer sur **Fichier texte**, et ajouter un composant CSV au mappage. Pour plus d'informations, voir Fichiers CSV et Texte[373].




Vous pouvez maintenant consulter le mappage. Cliquer sur la touche **Sortie** et observez le résultat de mappage dans le volet **Sortie**, par exemple :




## 4.2.6.3 Procédures stockées avec Entrée et Sortie

Cet exemple vous montre comment appeler une procédure qui prend des paramètres d'entrée et extrait des données de sortie de la base de données. Dans ce scénario, la procédure stockée est appelée comme pour un service Web ou une fonction et vous pouvez mapper des données extraites à tout autre composant cible pris en charge par MapForce.

Créons tout d'abord la procédure stockée de démonstration dans la base de données "AdventureWorks". Pour ce faire, exécuter le script ci-dessous par rapport à la base de données. Vous pouvez ce faire à partir d'une fenêtre de requête de **Microsoft SQL Server Management Studio**, ou directement depuis l'onglet **Requête BD** de MapForce, voir Parcourir et interroger des bases de données 300. Dans tous les cas, s'assurer que votre compte d'utilisateur de base de données a la permission de créer des procédures stockées.

```
CREATE PROCEDURE Production.uspSearchProducts
   @SearchString nvarchar(50)
   ,@MaxPrice money
   ,@ComparePrice money OUTPUT
AS
BEGIN
   SET NOCOUNT ON
   SELECT pr.[Name], pr.ListPrice FROM [Production].[Product] pr
   WHERE pr.[Name] like @SearchString AND  pr.ListPrice < @MaxPrice

   SET @ComparePrice = @MaxPrice
```

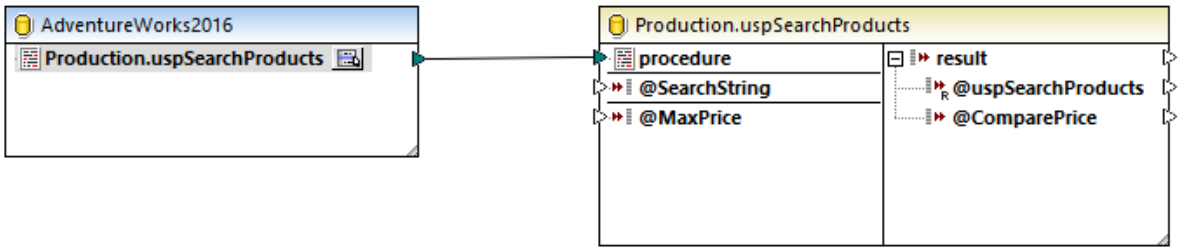
 

```
   RETURN @ComparePrice
END
```

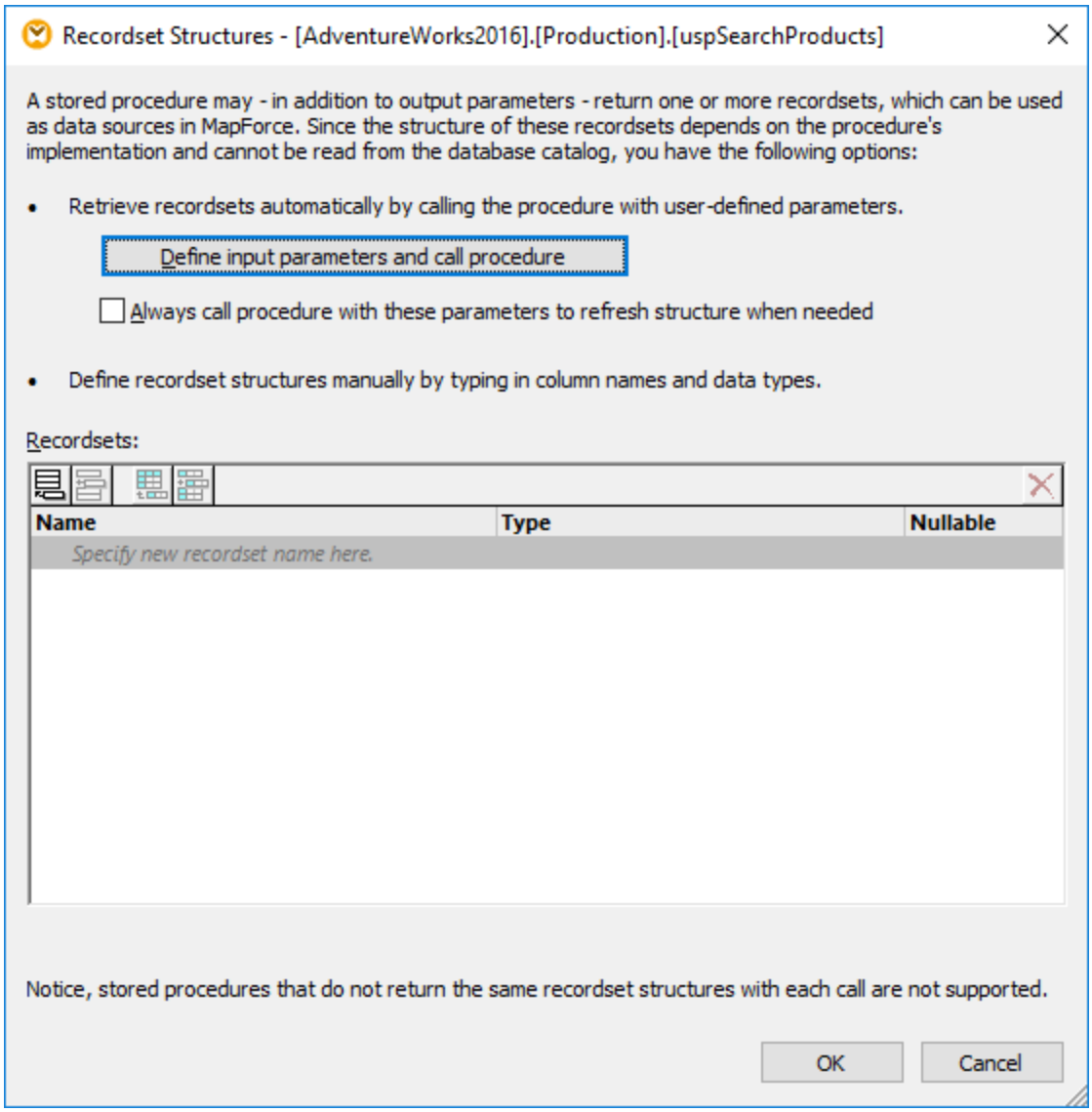
La procédure stockée ci-dessus extrait un recordset contenant des informations de produit. Elle prend deux paramètres en tant qu'entrée : un string avec le nom du produit (`@SearchString`) et le prix maximum du produit (`@MaxPrice`). Outre le recordset et le paramètre de retour par défaut, elle extrait aussi un paramètre de sortie (`@ComparePrice`).

Les étapes suivantes vous montrent comment créer un mappage qui consomme des données retournées par cette procédure.

1. Se connecter à la base de données "AdventureWorks" depuis MapForce, et ajouter la procédure stockée au mappage comme décrit dans la section Ajouter des procédures stockées dans le mappage 324. Veuillez vous assurer que votre compte utilisateur de base de données détienne la permission de consulter et d'exécuter des procédures stockées.
2. Cliquer sur la touche **Afficher menu contextuel** située à côté de la procédure stockée et choisir **Insérer appel avec paramètres**. La procédure stockée apparaît maintenant en tant que composant séparé dans le mappage, à l'endroit où le côté gauche recense les paramètres d'entrée, et le côté droit contient les paramètres de retour et de sortie.

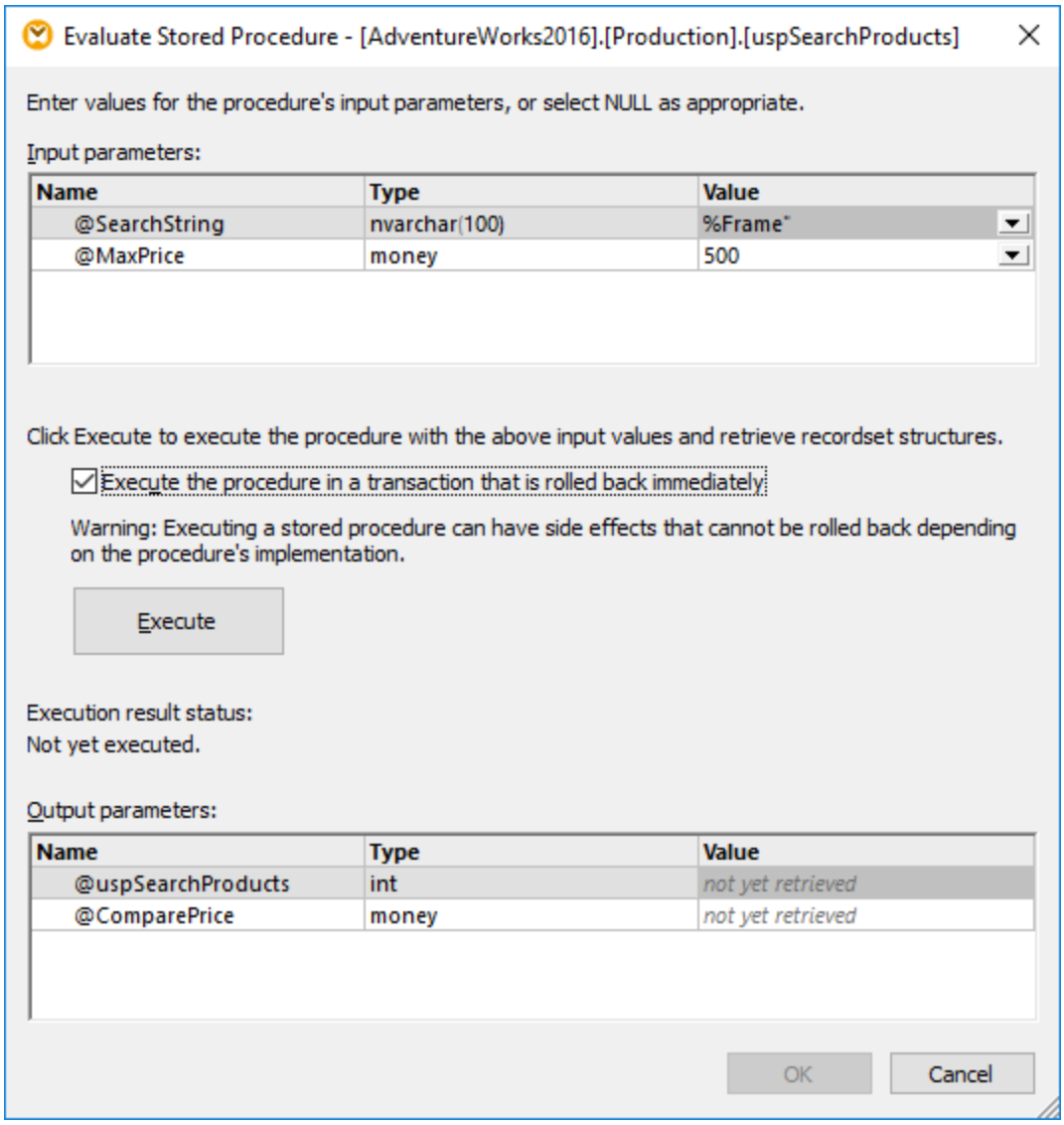



3. Cliquer à nouveau sur la touche **Afficher menu contextuel**, choisir **Éditer les structures de Recordset**. Cela est nécessaire pour fournir à MapForce des informations concernant la structure du recordset retourné par la procédure. Le dialogue "Structures de Recordset" apparaît.

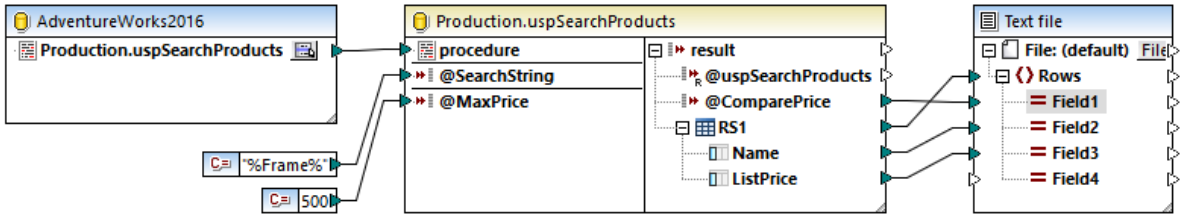



Le fait d'appeler une procédure stockée au moment de la conception peut présenter des effets secondaires (selon la mise en place de la procédure). Si vous ne souhaitez pas exécuter la procédure stockée au moment de la conception, ne pas cliquer sur **Exécuter**, comme décrit dans les étapes suivantes. Au lieu, définir le recordset attendu dans le dialogue "Structures de recordset", en ajoutant des recordsets et leurs colonnes associées manuellement. Utiliser les touches **Ajouter recordset** ou **Ajouter colonne** dans le dialogue "Structures de recordset".

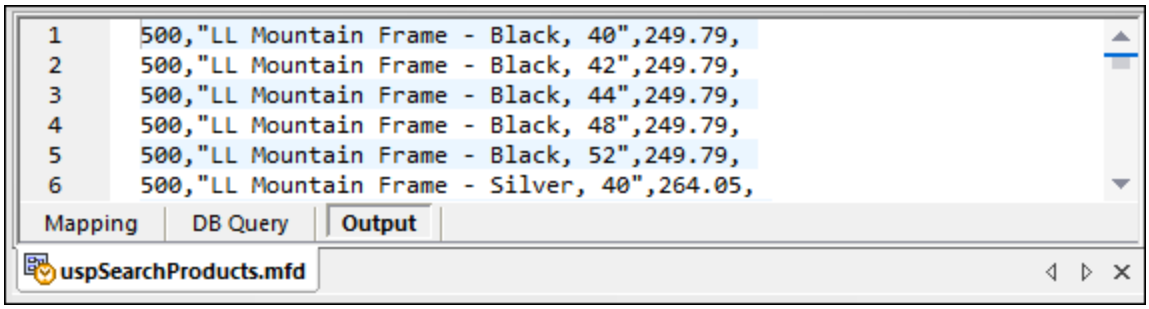
4. Cliquer sur **Définir les paramètres d'entrée et appeler la procédure**. Le dialogue "Évaluer procédure stockée"apparaît.




5. Remplir les valeurs de paramètre comme indiqué ci-dessus et cliquer sur **Exécuter**.
6. Cliquer sur **OK**. La structure de recordset ("RS1") est maintenant visible aussi bien dans le dialogue "Structures Recordset" que dans le mappage.
7. À ce niveau, vous pouvez ajouter un composant cible dans lequel les données extraites seront rédigées. Dans cet exemple, les données seront rédigées dans un fichier CSV. Dans le menu **Insérer**, cliquer sur **Fichier texte**, et ajouter un composant CSV au mappage, voir Fichiers CSV et Texte 373. Ensuite tracer les connexions de mappage tel qu'illutré ci-dessous. Veuillez noter que les paramètres d'entrée de la procédure sont fournis par le biais de constante. Pour plus d'informations concernant les constantes, voir Ajouter une constante au mappage 468.




Vous pouvez maintenant consulter le mappage. Cliquer sur la touche **Sortie** et observez le résultat de mappage dans le volet **Sortie**, par exemple :




## 4.2.6.4 Procédures stockées dans les Composants cibles

Cet exemple vous montre comment appeler une procédure qui prend des paramètres d'entrée et met à jour une base de données. Appeler une procédure de cette manière rend l'activation des transactions possible et annule l'action en cas d'erreur, ou ajouter une instruction SQL personnalisée à exécuter avant que la procédure ne soit appelée. Ce scénario implique que la procédure stockée agisse comme un un composant cible dans MapForce et que vous n'êtes pas intéressé par la sortie retournée par la procédure stockée. Pour consulter un exemple qui illustre comment passer des paramètres et mapper des données retournées par une procédure stockée, voir Procédures stockées avec Entrée et Sortie 330.

Créons tout d'abord la procédure stockée de démonstration dans la base de données "AdventureWorks". Pour ce faire, exécuter le script ci-dessous par rapport à la base de données. Vous pouvez ce faire à partir d'une fenêtre de requête de **Microsoft SQL Server Management Studio**, ou directement depuis l'onglet **Requête BD** de MapForce, voir Parcourir et interroger des bases de données 300. Dans tous les cas, s'assurer que votre compte d'utilisateur de base de données a la permission de créer des procédures stockées.

```
CREATE PROCEDURE Production.uspAddProductModel
   @ModelName nvarchar(50)
   ,@Inst xml
AS
BEGIN
INSERT INTO [Production].[ProductModel]
           ([Name]
           ,[Instructions]
           ,[rowguid]
           ,[ModifiedDate])
     VALUES
           (@ModelName
```

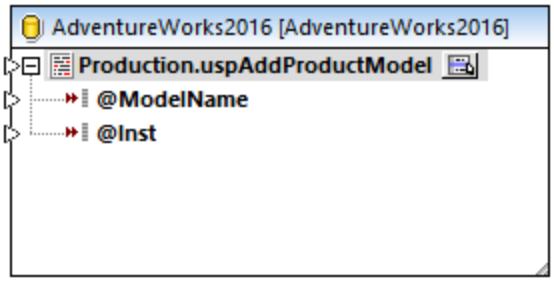
 

```
            ,@Inst
            ,NEWID()
            ,GETDATE())
END
```

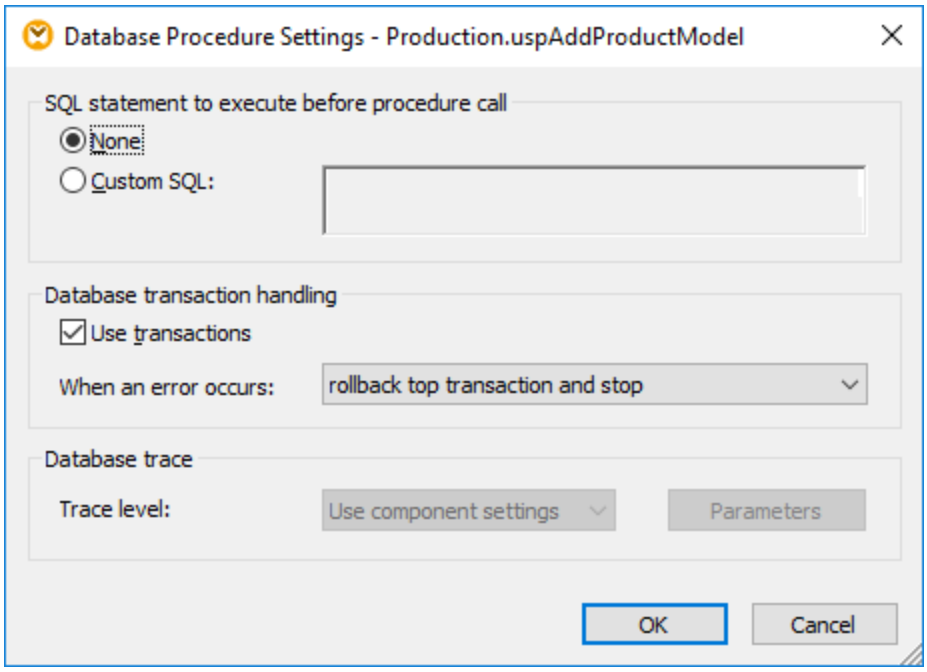
La procédure stockée ci-dessus prend deux paramètres (`@ModelName`, `@Inst`) en tant qu'entrée et insère les valeurs correspondantes dans la table `ProductModel` de la base de données AdventureWorks, avec quelques données générées par base de données.

Les étapes suivantes vous montrent comment créer un mappage qui consomme des données retournées par cette procédure.

1. Se connecter à la base de données "AdventureWorks" depuis MapForce, et ajouter la procédure stockée au mappage comme décrit dans la section Ajouter des procédures stockées dans le mappage 324. Assurez-vous que votre compte d'utilisateur de base de données détient la permission de créer des procédures stockées.
2. Cliquez sur le bouton **Afficher Menu contextuel** à côté de la procédure stockée et sélectionnez **Afficher Nœuds comme Cible**. La procédure stockée apparaît maintenant en tant que composant cible dans le mappage, à l'endroit où le côté gauche recense les paramètres d'entrée.

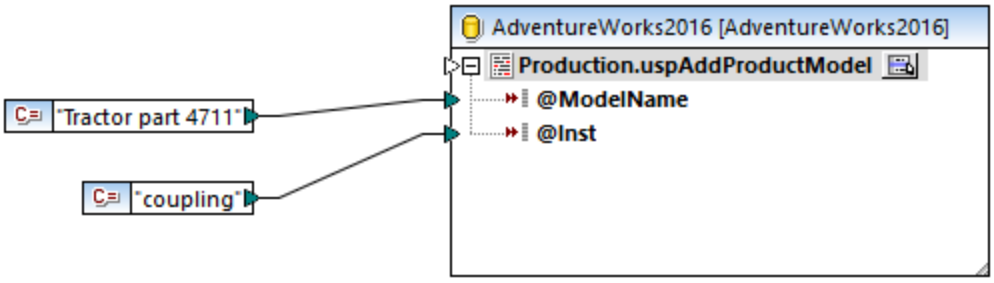



3. Cliquez sur le bouton **Afficher Menu contextuel** une nouvelle fois, et sélectionnez **Paramètres de procédure**. Cette étape optionnelle vous permet d'exécuter la procédure stockée à l'intérieur de la transaction qui peut être annulée. Vous pouvez aussi ajouter une instruction SQL personnalisée à exécuter avant que la procédure soit appelée.
4. Cochez la case à cocher **Utiliser transactions**.

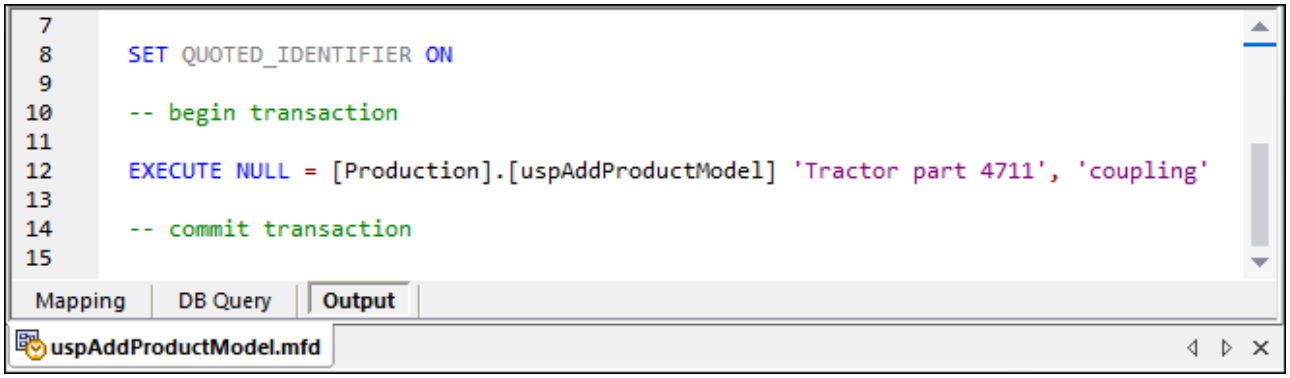



**Note :** Dans cet exemple, le traçage de base de données est désactivé au niveau du composant de base de données et aucun traçage n'est défini pour être exécuté. Néanmoins, vous pouvez activer un traçage de base de données pour les procédures stockées, le cas échéant.

5. Ajouter le composant source qui fournit les données à insérer dans la base de données. Dans cet exemple, les données de source sont fournies par des constantes ; néanmoins, tout autre composant de source pris en charge par MapForce peut agir en tant qu'entrée. Pour plus d'informations concernant les constantes, voir Ajouter une constante au mappage 468.




Puisque ce mappage met à jour une base de données, vous ne pouvez pas consulter sa sortie directement comme c'est le cas pour d'autres mappages. Au lieu, cliquer sur la touche **Sortie** pour afficher le pseudo-SQL contenant des indices concernant la manière dont la base de données sera modifiée. Si vous avez activé des transactions, celles-ci se produiront selon les indications par les commentaires.




Le pseudo-SQL afficher dans le volet **Sortie** ne montre pas les commandes de transaction réelles, il ne montre que des indices (comme des commentaires). Les commandes SQL réelles sont néanmoins envoyées à l'API de base de données sous-jacente.

Pour exécuter le mappage par rapport à la base de données, suivre une des deux étapes :

- Dans le menu **Sortie**, cliquer sur **Exécuter Script SQL**.
- Cliquer sur la touche de la barre d'outils **Exécuter Script SQL** .

## Procédures stockées et entrées doubles

Si vous souhaitez mapper des données provenant de plusieurs sources sur le mappage dans la même procédure stockée de manière à ce qu'elle accepte plusieurs entrées. Pour ce faire, cliquez avec la touche de droite sur l'item de procédure stocké dans le composant et sélectionner **Ajouter une entrée double** depuis le menu contextuel, voir aussi Doubler les entrées 45. Lorsque le mappage est exécuté, ces procédures stockées doubles seront appelées une fois pour chaque entrée double.

Veuillez noter que la commande Ajouter une entrée double **est désactivée pour les paramètres de procédure stockée, parce que chaque paramètre est une valeur atomique (et pourrait être "nullable").**

## Traçage au niveau de la procédure stockée

Pour activer le traçage au niveau de la procédure stockée :

1. Assurez-vous que le niveau de traçage au niveau du composant de base de données est défini soit comme **Toujours** ou **Erreur** (voir ci-dessus).
2. S'il s'agit d'une procédure stockée, cliquez sur la touche **Afficher le menu contextuel** , et choisir **Paramètres de procédure** depuis le menu contextuel.
3. Choisir le niveau de traçage. L'option **Utiliser les paramètres de composant** hérite des mêmes paramètres qui ont été définis au niveau du composant. L'option **Limiter aux erreurs** limite le traçage uniquement aux événements d'erreur. **Toujours désactivé** signifie qu'aucun traçage ne se produira pour cette table ou procédure stockée.

### 4.2.6.5 Procédures stockées et Relations locales

Les relations locales sont des relations logiques entre les champs de base de données que vous pouvez créer dans MapForce, ce qui vous évite de modifier la base de données sous-jacente, voir aussi Définir des relations locales 274. Vous pouvez définir des relations locales non seulement pour les champs de base de données, mais aussi pour les procédures stockées, aussi bien pour les composants de source et de cible.

Dans les composants de source, les relations locales permettent de lire des données depuis des objets liés, par exemple, lire des ID depuis une table de base de donnée et appeler une procédure stockée avec chacune de ces ID pour extraire des informations connexes. Il est également possible d'appeler une procédure stockée avec des données extraite d'une autre procédure.

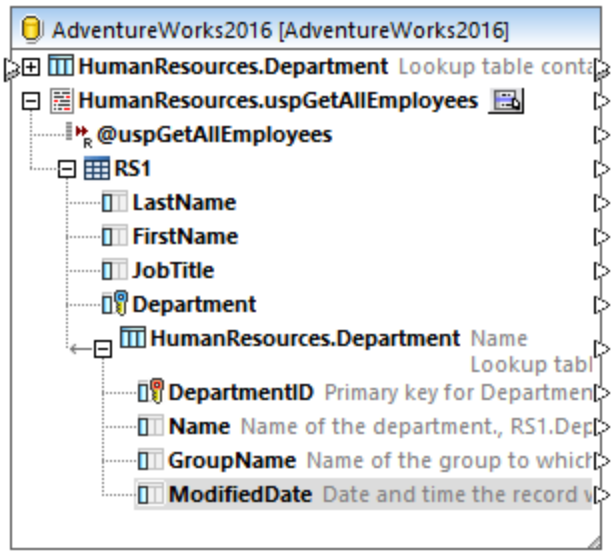
Dans les composants de cible, les relations locales permettent de définir un ordre hiérarchique dans lequel plusieurs procédures connexes doivent être appelées. Par exemple, vous pouvez d'abord appeler une procédure stockée qui crée une valeur ID, et une autre qui insère des informations connexes dans une table. Il est également possible de mélanger des procédures stockées et des tables dans des relations locales. Par exemple, vous pouvez effectuer l'insertion directement sur une table liée au lieu d'appeler une autre procédure, voir Utiliser des procédures stockées pour générer des clés 344.

**Pour créer une relation locale :**

1. Cliquer sur la barre de titre d'un composant de base de données et choisir **Ajouter/Supprimer/Éditer des objets de base de données** depuis un menu de contexte. Le dialogue "Ajouter/Supprimer/Éditer des objets de base de données" s'ouvre.
2. Cliquer sur **Ajouter/Éditer relations**.
3. Cliquer sur **Ajouter relation** et sélectionner les objets entre lesquels vous pouvez créer la relation.

Comme illustré ci-dessus, une relation locale consiste en un objet de **clé primaire/unique** et un objet de **clé étrangère**. On pourrait le comparer à une relation parent-enfant. Dans le composant de mappage, l'objet (table, mode, procédure, etc.) dans lequel la clé primaire/unique se trouve apparaîtra en tant que parent alors que l'objet dans lequel la clé étrangère se trouve apparaîtra imbriquée en-dessous. Par exemple, dans le composant de base de données comme illustré ci-dessous, une relation locale a été définie entre une colonne recordset (**RS1.Department**) et une colonne de table (**Department.Name**). Par conséquent, la table **Department** apparaît en tant qu'enfant d'une procédure stockée sur le mappage. Cet exemple est discuté avec plus de détails dans Relations locales dans les composants de source 339.




La table suivante recense tous les champs possibles entre lesquels vous pouvez définir des relations locales. Des relations mixtes sont possibles (par exemple, mapper la sortie d'une procédure stockée dans une colonne de base de données). Les champs participant à la relation doivent avoir les mêmes types de données ou des txpes de données compatibles.

| Clé primaire/unique | Clé étrangère |
|---|---|
| • La colonne d'une table de base de données ou d'un mode<br>• Paramètre de sortie ou valeur de retour d'une procédure stockée, voir aussi Procédures stockées 321<br>• La colonne d'un recordset retournée par une procédure stockée* * Applicable si la procédure stockée est appelée soit en tant que source de données (sans paramètres) ou en tant que fonction (avec paramètres d'entrée et de sortie). Pour que le recordset devienne disponible pour la sélection, vous devez exécuter la procédure stockée une fois, pour extraire le recordset.<br>• La colonne d'une instruction SELECT définie par l'utilisateur, voir aussi Instructions SELECT SQL en tant que tables virtuelles 264. | • La colonne d'une table de base de données ou d'un mode<br>• Paramètre d'entrée d'une procédure stockée<br>• Paramètre d'entrée d'une instruction SELECT définie par l'utilisateur |

## 4.2.6.6 Relations locales dans les composants de source

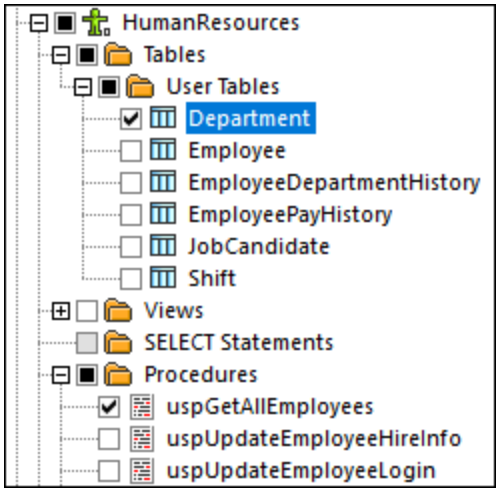
L'exemple suivant vous montre comment combiner des données retournées par une procédure stockée avec des données provenant d'une table dans la même base de données, à l'aide des relations locales.

Si vous ne l'avez pas encore fait précédemment, exécuter le script suivant pour créer la procédure stockée de démonstration dans la base de données "AdventureWorks". Vous pouvez y accéder depuis une fenêtre de requête de **Microsoft SQL Server Management Studio**, ou directement depuis le volet **Requête BD** de MapForce, voir Parcourir et interroger des bases de données 300. Dans tous les cas, veuillez vous assurer que votre compte d'utilisateur de base de données a la permission de créer des procédures stockées.

```
CREATE PROCEDURE HumanResources.uspGetAllEmployees
AS
    SELECT LastName, FirstName, JobTitle, Department
    FROM HumanResources.vEmployeeDepartment
```

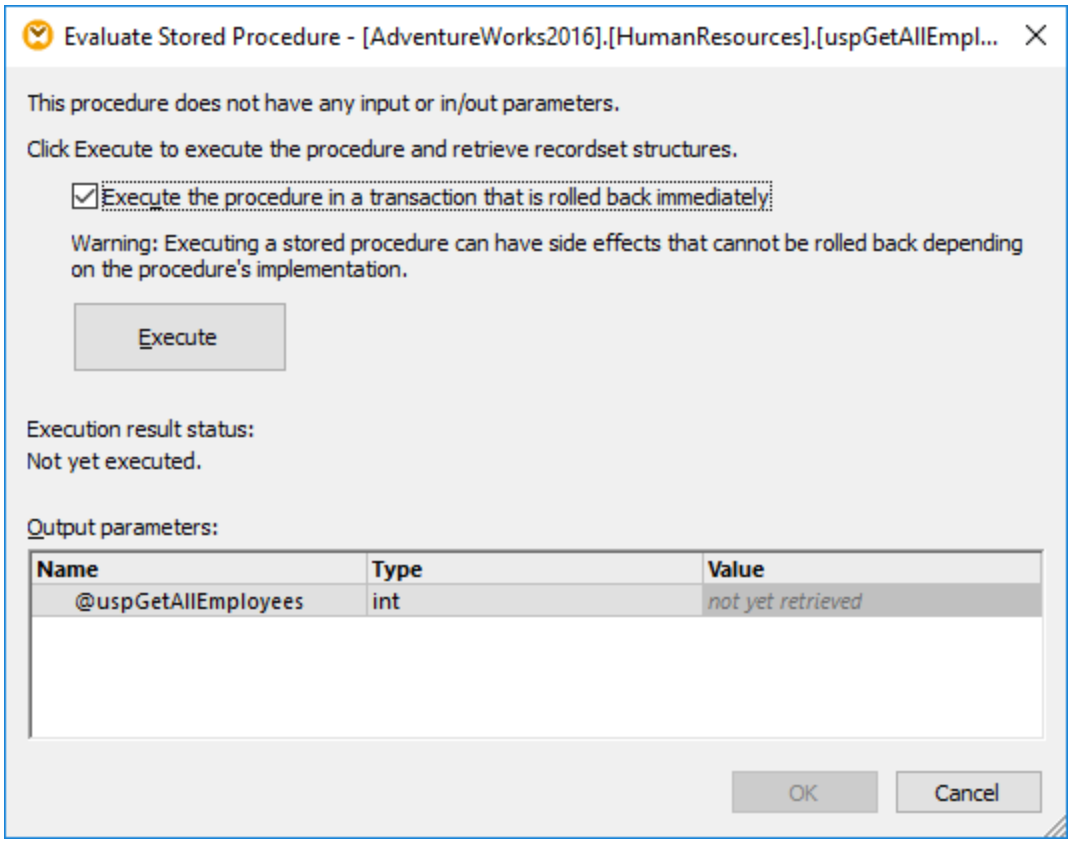
La procédure stockée ci-dessus retourne l'information de l'employé depuis le mode **vEmployeeDepartment**. Les étapes suivantes vous montrent comment créer un mappage qui consomme les données retournées par cette procédure.

1. Se connecter à la base de données "AdventureWorks" depuis MapForce, comme décrit dans Ajouter des procédures stockées dans le mappage 324. Assurez-vous que votre compte d'utilisateur de base de données détient la permission de consulter et d'exécuter les procédures stockées.
2. Lorsque vous êtes invité à choisir des objets de base de données, choisir la table, choisir la table **Department** et la procédure stockée **uspGetAllEmployees**.

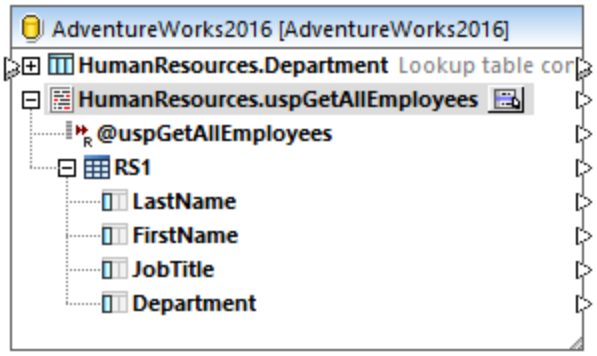



3. Cliquer sur la touche **Afficher menu contextuel** située à côté de la procédure stockée et choisir **Afficher nœuds en tant que cible**.
4. Cliquer à nouveau sur la touche **Afficher menu contextuel**, choisir **Paramètres de procédure**. Le dialogue "Structures Recordset" apparaît.
5. Cliquer sur **Définir les paramètres d'entrée et appeler la procédure**. Le dialogue "Évaluer la procédure stockée" apparaît.



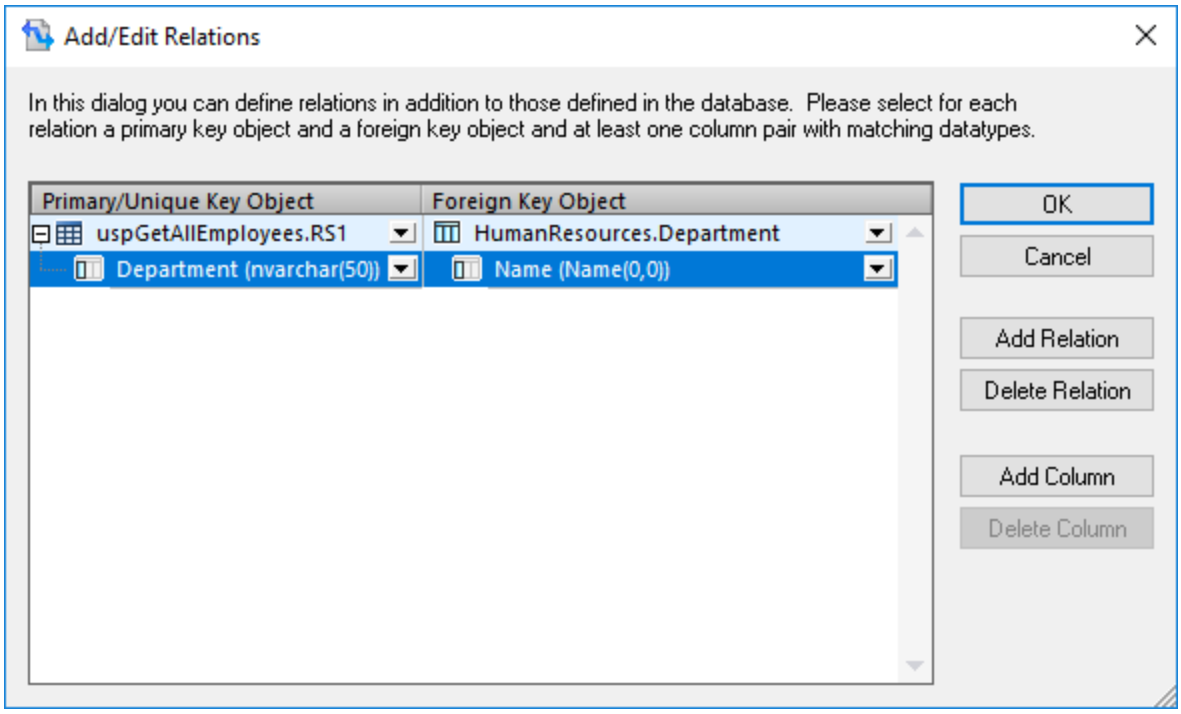

6. Cliquer sur **Exécuter**, puis sur **OK**. La structure recordset ("RS1") est maintenant visible aussi bien dans le dialogue "Structures Recordset" que dans le mappage.



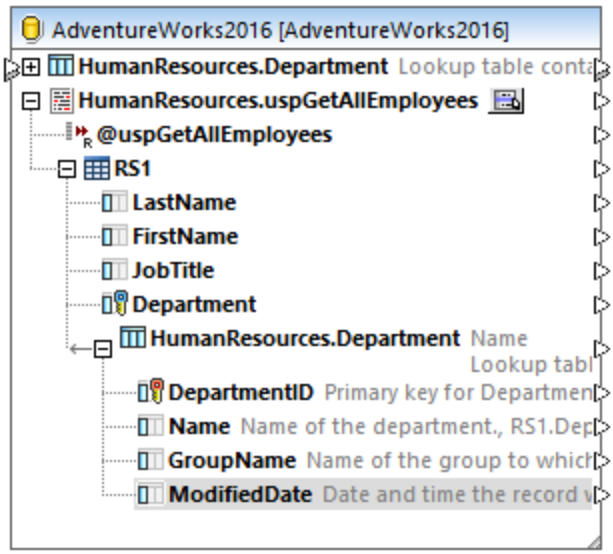

## Définir les relations locales

À présent, il s'agit de définir une relation locale entre la colonne **Department** du recordset retourné et la colonne **Name** de la table **Department**.

1. Cliquer avec la touche de droite sur la barre de titre du composant de la base de données et choisir **Ajouter/Supprimer/Éditer objets de base de données** du menu contextuel.
2. Cliquer sur **Ajouter/Éditer relations**, puis cliquer sur **Ajouter relation**. Définir les relations comme indiqué ci-dessous.



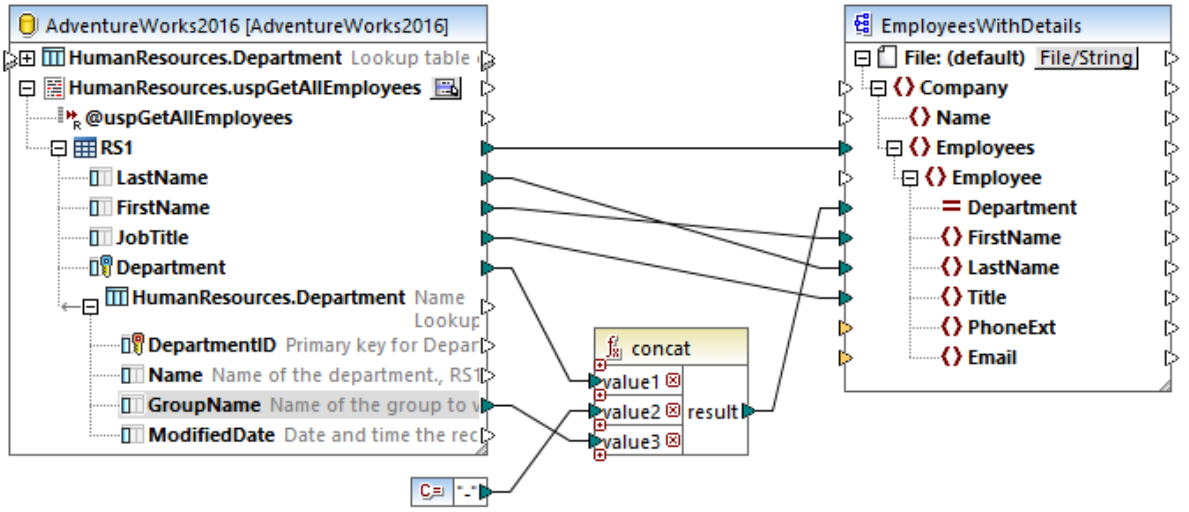

3. Cliquer sur **OK** pour fermer le dialogue. Veuillez noter que la table **Department** devient un enfant du recordset **RS1**.




## Achever le mappage

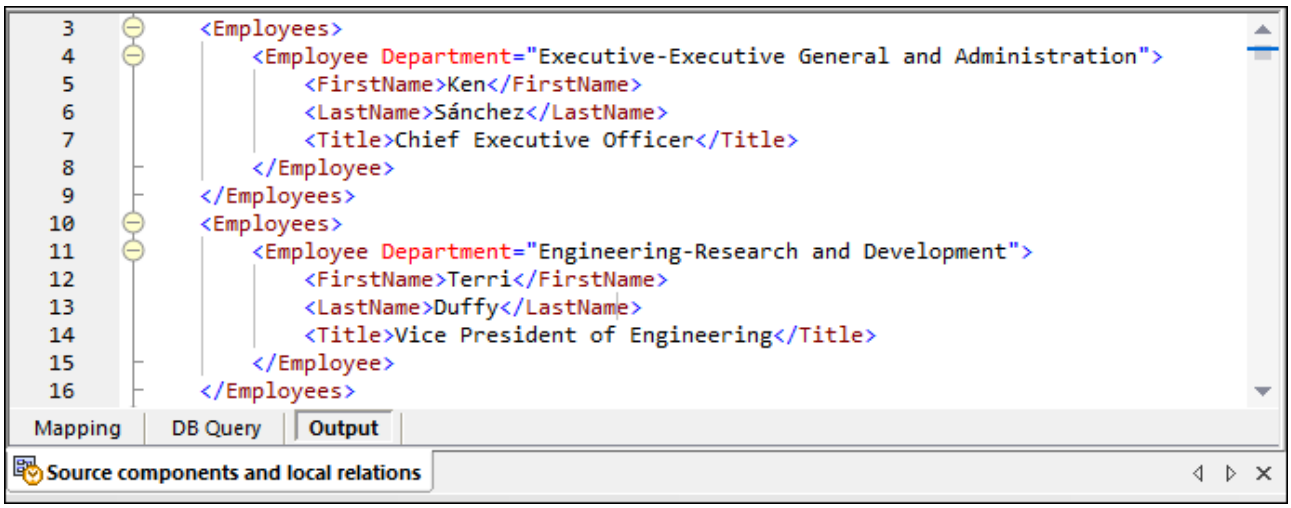
Grâce à la relation qui vient d'être créée, il est maintenant possible de mapper des données depuis le recordset combiné avec des données provenant de la table. Pour illustrer notre exemple, nous écrivons des données vers un fichier cible, comme suit :

1. Dans le menu **Insérer**, cliquer sur **Schéma XML/Fichier** et choisir le fichier suivant : **<Documents>\Altova\MapForce2024\MapForceExamples\EmployeesWithDetails.xsd**.
2. Lorsque vous êtes invité à fournir un échantillon d'instance XML, cliquer sur **Sauter**.
3. Tracer les connexions de mappage comme indiqué ci-dessous.




Le mappage illustré ci-dessus écrit des données depuis la base de données vers un fichier XML cible. Les données de source sont constituées par une combinaison de données extraites par la procédure stockée avec des données extraites directement depuis une table. Le mappage utilise la fonction **concat** 629 pour produire un string qui inclut le nom du département, suivi d'un tiret et suivi du nom du groupe.

Pour consulter le mappage, cliquer sur la touche **Sortie** et observer le résultat de mappage dans le volet **Sortie**, par exemple :



## 4.2.6.7 Utiliser des procédures stockées pour générer des clés

Cet exemple vous montre comment insérer des clés (ID) générées par une procédure stockée dans une autre table, avec l'aide des relations locales.

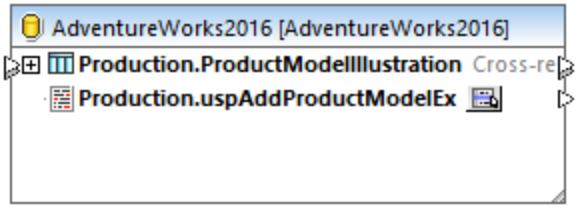
Créons tout d'abord la procédure stockée de démonstration dans la base de données "AdventureWorks". Pour ce faire, exécuter le script ci-dessous par rapport à la base de données. Vous pouvez ce faire à partir d'une fenêtre de requête de **Microsoft SQL Server Management Studio**, ou directement depuis l'onglet **Requête BD** de MapForce, voir Parcourir et interroger des bases de données 300. Dans tous les cas, s'assurer que votre compte d'utilisateur de base de données a la permission de créer des procédures stockées.

```
CREATE PROCEDURE Production.uspAddProductModelEx
  @ModelName nvarchar(50)
  ,@Inst xml
  ,@ProductModelID int OUTPUT
AS
BEGIN
INSERT INTO [Production].[ProductModel]
           ([Name]
           ,[Instructions]
           ,[rowguid]
           ,[ModifiedDate])
     VALUES
           (@ModelName
           ,@Inst
           ,NEWID()
           ,GETDATE())
   SELECT @ProductModelID = SCOPE_IDENTITY()
END
```

La procédure stockée ci-dessus prend deux paramètres (`@ModelName`, `@Inst`) en tant qu'entrée et effectue une opération `INSERT` dans la table **ProductModel**. Elle retourne ensuite le `@ProductModelID` généré en tant que paramètre de sortie. L'exigence est d'insérer le `@ProductModelID` retourné par la procédure stockée dans la table **ProductModelIllustration**.

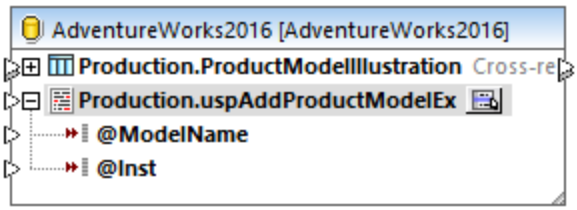
Les étapes suivantes vous montrent comment créer un mappage qui satisfait aux exigences ci-dessus.

1. Se connecter à la base de données "AdventureWorks" depuis MapForce, comme décrit dans la section Ajouter des procédures stockées dans le mappage 324. Veuillez vous assurer que votre compte utilisateur de base de données détienne la permission de consulter et d'exécuter des procédures stockées.
2. Lorsque vous êtes invité à choisir des objets de base de données, choisir la table **ProductModelIllustration** et la procédure stockée **uspAddProductModelEx**.

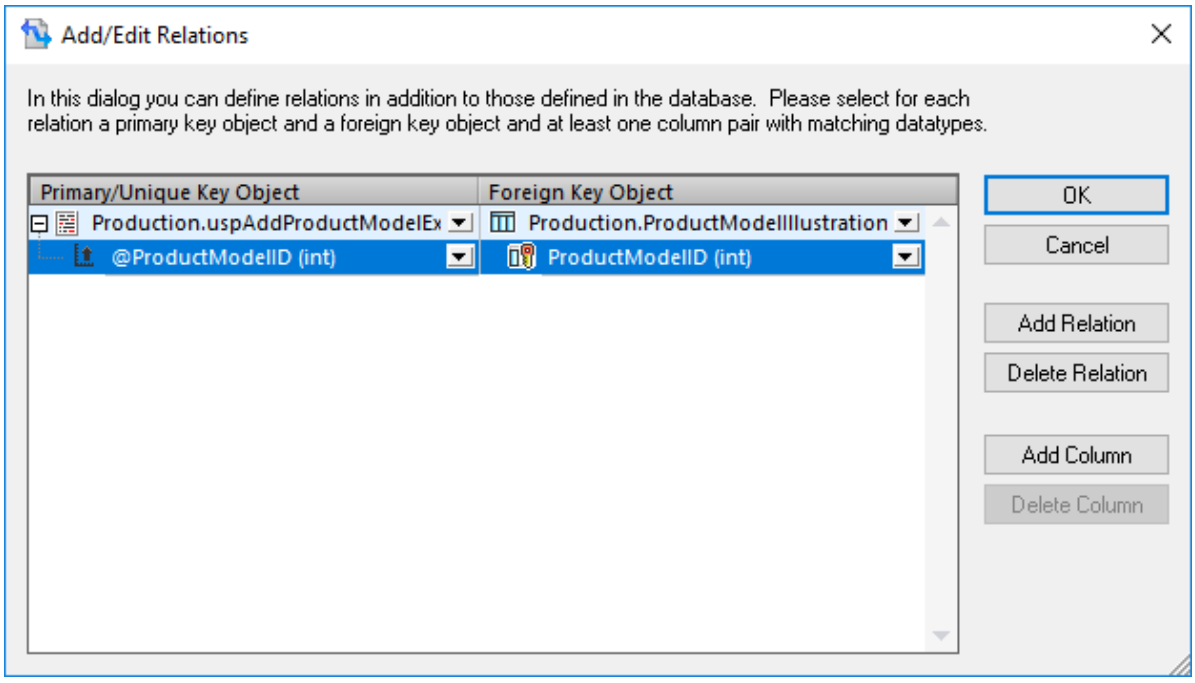


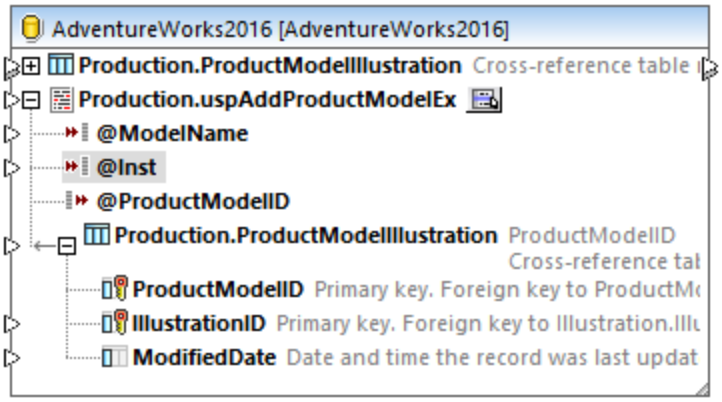
3. Cliquer sur la touche **Afficher menu contextuel** située à côté de la procédure stockée et choisir **Afficher nœuds en tant que cible**. La procédure stockée apparaît maintenant en tant que composant de cible dans le mappage, et le côté gauche recense les paramètres d'entrée.



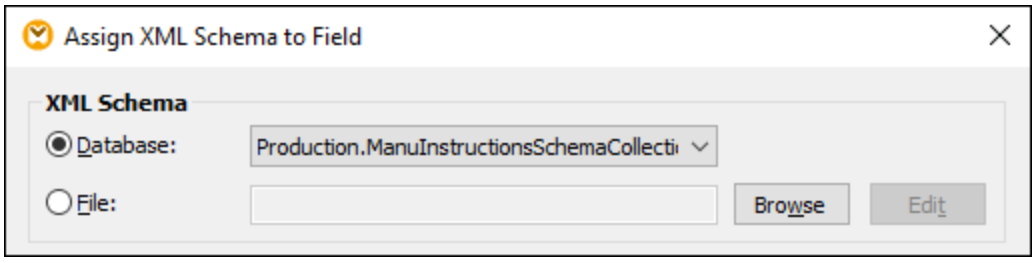

4. En option, si vous souhaitez exécuter la procédure stockée dans une transaction, cliquer à nouveau sur la touche **Afficher menu contextuel**, choisir **Paramètres de procedure**, et cocher la case **Utiliser des transactions**. Le fait de définir la transaction pour la procédure stockée permet d'assurer que l'extraction de la clé et l'insertion de l'enregistrement se produise pendant la même transaction.
5. Cliquer avec la touche de droite sur la barre de titre du composant de base de données et choisir **Ajouter/Supprimer/Éditer objets de base de données** depuis le menu contextuel.
6. Cliquer sur **Ajouter/Éditer des relations**, puis cliquer sur **Ajouter relation**. Définir les relations comme indiqué ci-dessous.




7. Cliquer sur **OK** pour fermer le dialogue. Veuillez noter que la table **ProductModelIllustration** apparaît maintenant en tant qu'enfant de la procédure stockée. Le paramètre de sortie de la procédure stockée (`@ProductModelID`) est affiché en tant qu'un indicateur qu'il sera utilisé dans la relation locale, mais il n'a pas de connecteurs d'entrée ou de sortie.

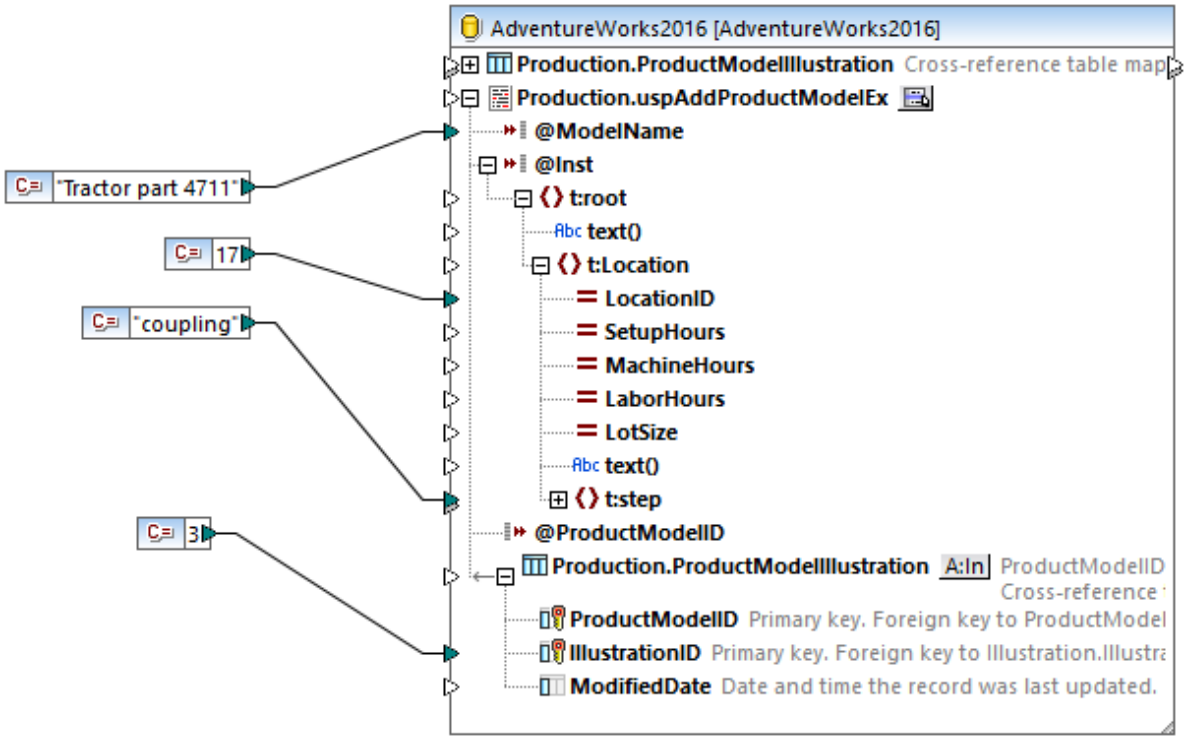



8. Dans cet exemple, le paramètre `@Inst` est de type XML. Cliquer avec la touche de droite sur le paramètre `@Inst` dans le composant et choisir **Attribuer Schéma XML au champ** depuis le menu contextuel. Ensuite, choisir le schéma **Production.ManuInstructionsSchemaCollection** depuis la base de données. Lorsque vous serez invité à choisir un élément racine, laisser la valeur par défaut telle quelle et cliquer sur **OK**. Pour plus d'informations concernant le mappage de données vers des champs XML de base de données, voir Mapper des données XML de / vers des champs de base de données 310.

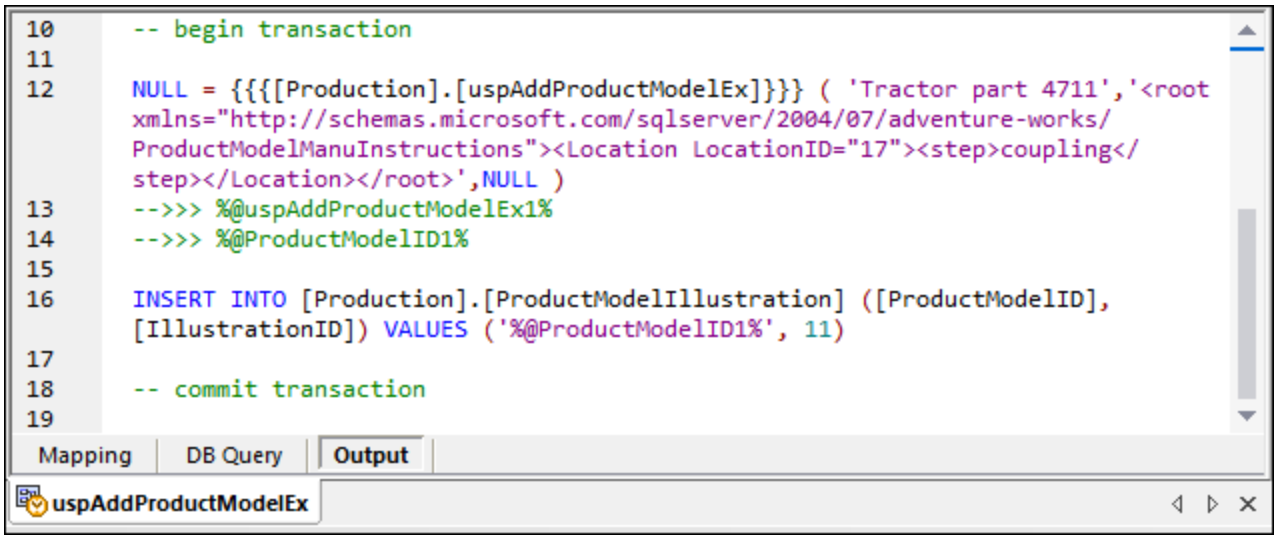



9. Ajouter les composants de source qui fournissent les données à insérer dans la base de données. Dans cet exemple, les données de source sont fournies par des constantes ; néanmoins, tout autre composant de source pris en charge par MapForce pourrait agir en tant qu'entrée. Pour plus d'informations concernant les constantes, voir Ajouter une constante au mappage 468.




Puisque ce mappage met à jour une base de données, vous ne pouvez pas consulter sa sortie directement comme c'est le cas pour d'autres mappages. Au lieu, cliquer sur la touche **Sortie** pour afficher le pseudo-SQL contenant des indices concernant la manière dont la base de données sera modifiée. Si vous avez activé des transactions, celles-ci se produiront selon les indications par les commentaires.




Le pseudo-SQL afficher dans le volet **Sortie** ne montre pas les commandes de transaction réelles, il ne montre que des indices (comme des commentaires). Les commandes SQL réelles sont néanmoins envoyées à l'API de base de données sous-jacente.

Pour exécuter le mappage par rapport à la base de données, suivre une des deux étapes :

- Dans le menu **Sortie**, cliquer sur **Exécuter Script SQL**.
- Cliquer sur la touche de la barre d'outils **Exécuter Script SQL**  .

## 4.2.7 Joindre des données de base de données

Dans des mappages qui lisent des données depuis des bases de données, vous pouvez joindre des objets de base de données comme des tables ou des vues en ajoutant un composant Join dans le mappage. Par exemple, vous pouvez combiner des données provenant de deux ou plusieurs tables liées par des relations de clé étrangère, ce qui constitue la méthode typique pour stocker des données dans des base de données relationnelles. Le résultat est le même que si vous exécutez une requête SQL par rapport à la base de données lorsque deux ou plus tables sont jointes par le biais d'une opération INNER JOIN (ou LEFT JOIN, le cas échéant).

Selon le type de données connectées au composant join, l'opération join peut se produire soit dans le mode standard (non-SQL), soit dans le mode SQL. Les jointures dans le mode non-SQL sont effectués par MapForce, alors que les jointures dans le mode SQL sont effectués par la base de données à partir de laquelle le mappage lit les données.

Des jointures en mode non-SQL sont plus flexibles car elles prennent en charge plus de types de composant en tant qu'entrée (par exemple, la jointure peut être entre des tables provenant de bases de données différentes, ou entre des structures XML et des tables de base de données). Pour consulter un exemple d'une jointure non-SQL, voir Exemple : Joindre des structures XML 432. D'un autre côté, une jointure non-SQL oblige le moteur de mappage a effectuer des comparaisons coûteuses en mémoire (parce que le nombre total de comparaisons représente la jointure croisée, ou le produit cartésien, de toutes les structures jointes). Généralement, ce processus a lieu très rapidement et est négligeable dans des mappages qui ne nécessitent pas trop de données ; néanmoins, si les sources de données jointes consistent en un nombre important d'enregistrements, le mappage prendra un temps considérable à s'exécuter. Si vos mappages doivent traiter un grand nombre d'enregistrements, nous vous recommandons de considérer la mise sous licence de MapForce Server Advanced Edition, qui contient une optimisation join dédiée pour accélérer l'exécution de mappage.

Une jointure dans le mode SQL n'accepte que les objets de base de données éligibles en tant qu'entrée (comme des tables ou des vues), elle n'est donc pas aussi flexible qu'une jointure non-SQL. Néanmoins, elle offre de meilleures performances de mappage parce qu'elle est exécutée de manière native par la base de données. Pour plus d'informations voir Joins dans le Mode SQL 349.

**Note :** L'utilisation d'un composant Join n'est pas le seul moyen de joindre des tables ou des affichages de base de données. Les jointures applicables aux bases de données peuvent aussi être effectués en utilisant des instructions SELECT SQL, voir Instructions SELECT SQL en tant que tables virtuelles 264. Une différence principale entre les instructions SELECT SQL et les composants Join est que les premières sont écrites à la main, ce qui leur confère une plus grande flexibilité. Les composants Join sont une alternative plus simple si vous ne souhaitez pas rédiger des instructions SQL à la main.

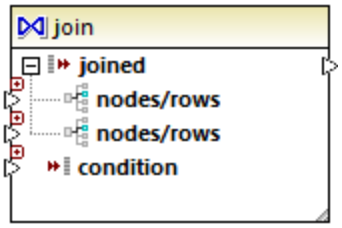
**Pour ajouter un composant Join :**

1. Définir le langage de transformation de mappage sur BUILT-IN (pour ce faire, cliquer soit sur la touche

de la barre outils , soit utiliser la commande de menu **Sortie | Moteur d'exécution Built-In**).

2. Dans le menu **Insérer**, cliquer sur **Join**. En alternative, cliquer sur la touche de la barre outils **Join** ( ). Le composant Join apparaît dans le mappage. Par défaut, il accepte des données provenant de deux structures, il a donc deux entrées `nodes/rows`. Si nécessaire, vous pouvez ajouter de nouvelles entrées dans la jointure en cliquant sur la touche **Ajouter entrée** ( ), voir Joindre trois structures ou plus 431.




3. Connecter les structures qui doivent être jointes dans l'item `nodes/rows` du composant join.
4. Ajouter la condition pour la jointure (ou plusieurs conditions). Pour ce faire, cliquer avec la touche de droite dans le composant Join et choisir **Propriétés**. Les conditions Join peuvent aussi être ajoutées directement depuis le mappage, en connectant le résultat Booléen de certaines fonctions à l'item `condition` du composant Join. Dans certains cas. lorsque des tables de base de données sont jointes, la condition (ou les conditions) de jointure peut être créée automatiquement par MapForce. Pour plus d'informations, voir Ajouter des conditions Join 802.

Notes :

- Les composants Join sont pris en charge lorsque la langue cible du mappage est configurée sur BUILT-IN. La génération de code dans C#, C++, ou Java n'est pas prise en charge.
- Lorsqu'une structure n'est pas une source d'entrée valide ou prise en charge pour la jointure, MapForce affiche des indices soit immédiatement et directement dans le mappage, soit dans la fenêtre Messages, lorsque vous validez le mappage (voir Valider des mappages 68).
- Les composants Join ne doivent pas être connectés aux paramètres d'entrée ou aux résultats des fonctions définies par l'utilisateur inline. Si de telles connexions existent, des erreurs de validation se produiront au cours de la validation de mappage.
- Lorsque vous vous connectez à des composants de base de données éligibles (comme des tables ou des modes) directement dans un composant Join, une touche **mode SQL** ( ) apparaît automatiquement en haut à droite du composant Join. Une fois activée, cette touche offre des fonctions SQL spéciales applicables à l'opération de jointure (voir Joins dans le mode SQL 349).
- Il n'est pas possible de connecter la sortie de l'item `joined` à un autre composant Join. Néanmoins, le cas échéant, vous pouvez connecter un résultat partiel à une jointure d'un autre.

## 4.2.7.1 À propos des joins dans le Mode SQL

Lorsque vous connectez des composants de base de données éligibles (comme des tables ou des vues) directement à un composant join, un bouton **mode SQL** ( ) apparaît automatiquement dans le coin supérieur droit du composant join. Lorsque le mode SQL est activé, l'opération join est entreprise par la base de données à partir de laquelle le mappage lit les données. En d'autres termes, MapForce enverra une requête en interne vers la base de données avec la syntaxe SQL appropriée pour sélectionner et combiner des données depuis toutes les tables qui participent à la jointure. Attention: vous n'avez pas besoin d'écrire de SQL ; la requête désirée est produite sur la manière dont vous avez conçu visuellement le composant Join sur le mappage, comme vous le verrez dans les exemples suivants.

Pour que le mode SQL soit possible, les conditions suivantes doivent être satisfaites :

1. Les deux objets (tables ou vues) qui doivent être joints doivent provenir de la même base de données.
2. Les deux objets qui doivent être joints doivent provenir du même composant MapForce. (Veuillez noter que vous pouvez rapidement ajouter/supprimer des objets de base de données dans un composant comme suit : cliquer avec la touche de droite sur le composant de base de données, et choisir **Ajouter/Supprimer/Éditer des objets de base de données** depuis le menu contextuel.)
3. La condition Join (ou les conditions) doivent uniquement être définies depuis les propriétés de composant (en cliquant avec la touche de droite sur l'en-tête du composant join, et choisir **Propriétés**), et pas sur le mappage (voir aussi Ajouter des conditions Join 428).

**Note :** Lorsque des tables de base de données sont jointes dans le mode SQL, MapForce créera la condition Join (ou les conditions) automatiquement, sur la base des relations de clé étrangère détectée entre les tables. Pour que des conditions Join automatiques se produisent, les tables de base de données doivent se trouver dans une relation enfant-parent dans le composant MapForce (c'est à dire, qu'une table doit être "parent" ou "child" d'un autre dans le composant), voir Exemple: Joindre des tables dans le mode SQL 352.

4. Toutes les tables de base de données ne doivent pas encore être se trouver dans le contexte cible actuel. Lorsque le résultat join est utilisé dans un composant cible, aucune des tables réunies ne peut être connectée directement ou indirectement vers des nœuds de parent cible. Pour plus d'informations concernant la manière dont un mappage est exécuté, voir Règles et stratégies de mappage 802.

Vous pouvez visualiser ou contrôler le mode SQL par le biais de la touche **SQL** (  ) situé dans le coin supérieur droit du composant join, comme suit :

 Le mode SQL est désactivé (join sera exécuté par MapForce (ou, si applicable, par MapForce Server).

 Le mode SQL est activé (join sera exécuté par la base de données).

Si la touche  est manquante, cela signifie que le mode SQL ne sera pas significatif ou pas pris en charge pour les données qui sont jointes.

Dans certains cas, le mode SQL doit être désactivé explicitement (  ), par exemple:

- Lorsque votre mappage nécessite des conditions join en-dehors des propriétés de composant join (c'est à dire, des conditions définies sur le mappage et connectées à l'item `condition` du composant join).
- Lorsque vous souhaitez joindre des tables depuis des base de données variées. Utiliser un joint standard (non-SQL) si vous souhaitez joindre des tables depuis des base de données différentes.

## Changer le mode Join

Lorsque le composant Join se trouve dans le **mode SQL**  , vous pouvez joindre les tables de base de données ou les vues des manières suivantes :

- INNER JOIN - Seuls les enregistrements qui satisfont la condition dans les deux ensembles d'entrée sont retournés par le composant Join.

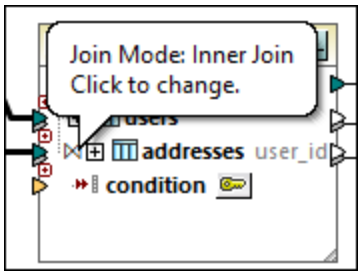
 

- LEFT OUTER JOIN - Le composant Join inclut tous les enregistrements provenant de la table "à l'extrême gauche" (dans MapForce, il s'agit de la table supérieure d'un composant Join), plus les enregistrements provenant de la table réunie par la suite qui satisfait la condition join.

Pour consulter le mode join d'une table ou consulter le composant Join, observer l'icône affichée devant la table réunie ou la vue. Une des icônes suivantes peut s'afficher pour toute table réunie ou vue sauf la première:

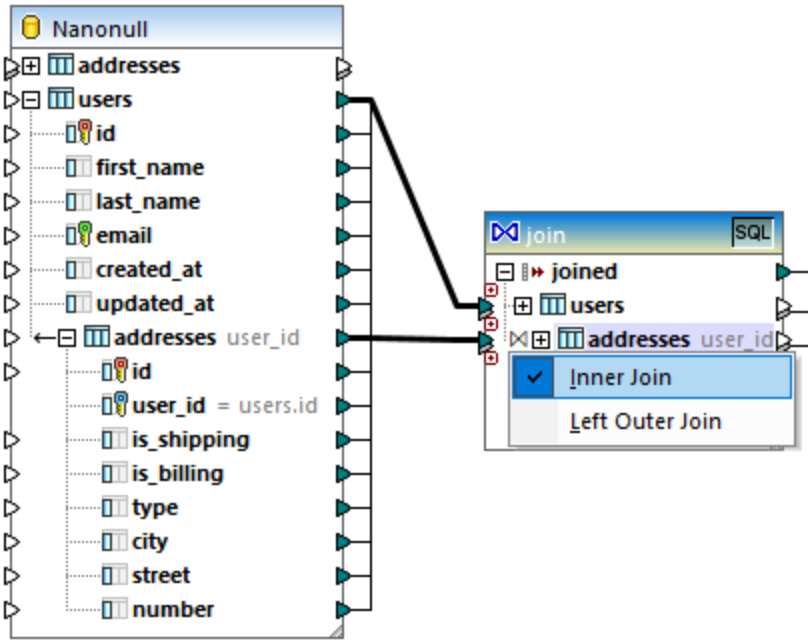
- Inner Join ⋈
- Left Join ⟕

Pour afficher une infobulle contenant des détails concernant le join, déplacer le curseur sur l'icône :




Pour modifier le mode join, :

- Cliquer sur l'icône **Inner Join** ⋈ ou **Left Join** ⟕ devant chaque table ou vue réunie et choisir **Inner Join** ou **Left Outer Join** depuis le menu contextuel.
- Cliquer avec la touche de droite sur la deuxième (ou troisième, quatrième, etc) table ou vue réunie dans le composant Join, et choisissez **Join Type | Inner Join**, ou **Join Type | Left Outer Join** depuis le menu contextuel.




Noter :

- Si vous avez changé le mode join en LEFT OUTER JOIN, la table ou la vue supérieure représentera le côté "gauche" du join.
- Le fait de modifier le mode join affecte les données retournées par le composant join de la même manière que INNER JOIN ou LEFT JOIN impacte le résultat d'une requête SQL dans une base de données.

### Noms d'alias

Il arrive souvent que des tables ou des vues de base de données jointes contiennent des noms de champ identiques dans les deux structures jointes. Lorsque le mode SQL est activé, ces items apparaissent sur le composant préfixé par le mot-clé "AS". Par exemple, si les deux tables jointes contiennent un champ "id", ce champ apparaît comme "id" sur la première table jointe et en tant que "id AS id2" sur la deuxième table jointe. Les tables jointes peuvent aussi produire des noms d'alias, par exemple, si la même table est jointe à elle-même.

Le champ alias ou les noms de table sont importants si vous souhaitez vous y référer dans un mappage ultérieurement. Par exemple, imaginez le cas où vous souhaitez filtrer ou trier le résultat de la jointure. Pour y parvenir, la sortie du composant join peut être connectée à un composant WHERE/ORDER SQL, où vous pourrez saisir les clauses SQL WHERE et ORDER BY.

Pour se référer à un champ depuis la clause WHERE, écrire le nom de la table, suivi d'un point (.), suivi par le nom de champ. Pour vous référer à un alias de table, utiliser le nom d'alias comme il apparaît dans le composant Join. Dans la clause ORDER BY, vous pouvez utiliser soit la même technique (`table.field`), ou écrire juste le nom de champ d'alias (le nom qui apparaît après "AS").

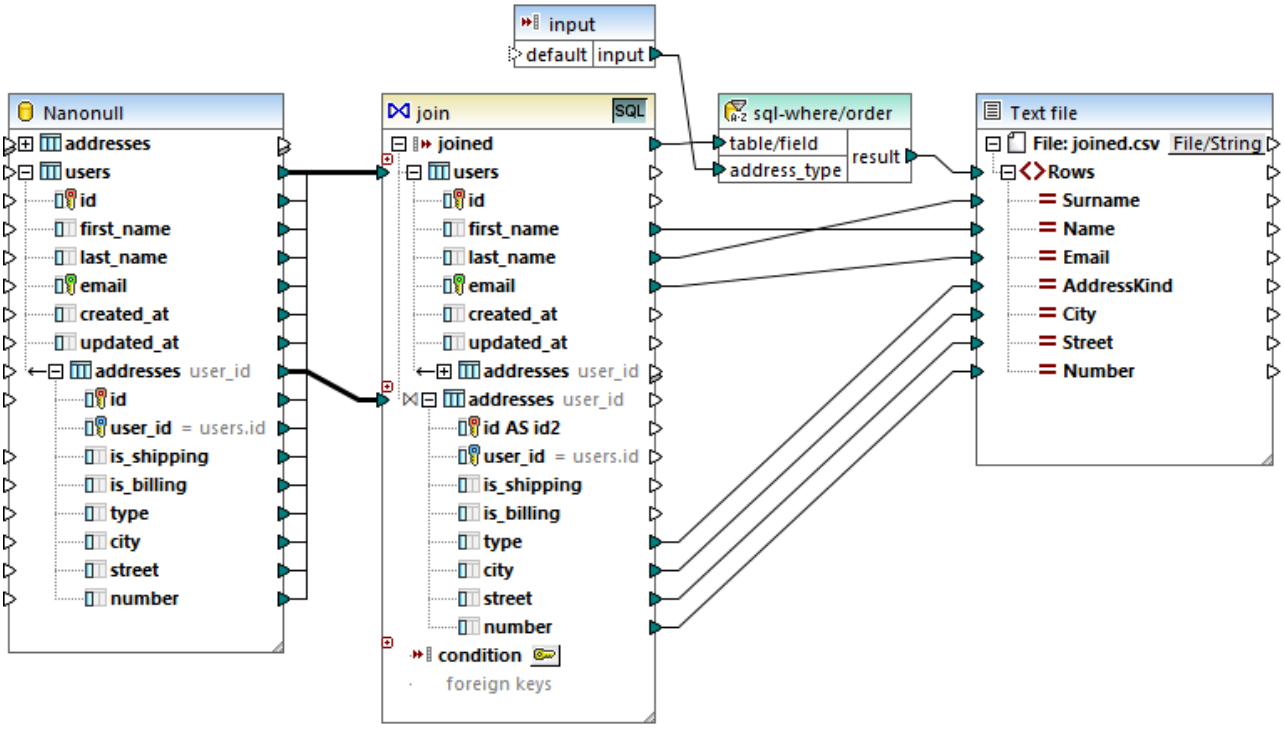
Pour un mappage d'exemple qui utilise des clauses WHERE/ORDER SQL, voir Exemple : Tables Join dans le mode SQL 352.

**Note :** Les composants WHERE/ORDER SQL ne sont pas autorisés entre une table de base de données et le composant join ; Ils peuvent uniquement être ajoutés après (et non pas avant) un composant join. Pour plus d'informations concernant les composant WHERE/ORDER SQL, voir Filtrer et trier les données de base de données (WHERE/ORDER SQL) 365.

## 4.2.7.2 Exemple : Joindre des tables dans le mode SQL

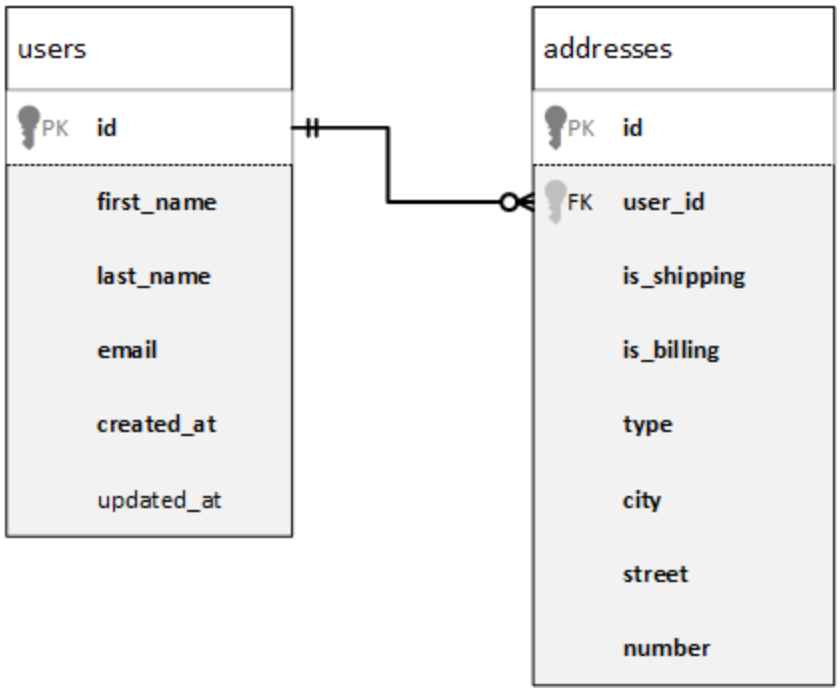
Cet exemple illustre comment joindre des données provenant de deux tables de base de données, en utilisant un composant join de MapForce. L'opération join est effectuée dans le mode SQL, comme décrit dans Joins dans le Mode SQL 349. Veuillez noter que la jointure de trois tables ou plus fonctionne de manière très semblable, voir aussi Exemple : Créer un rapport CSV à partir de plusieurs tables 361.

L'exemple est accompagné d'un échantillon de mappage disponible dans le chemin suivant :
**<Documents>\Altova\MapForce2024\MapForceExamples\Tutorial\JoinDatabaseTables.mfd**.

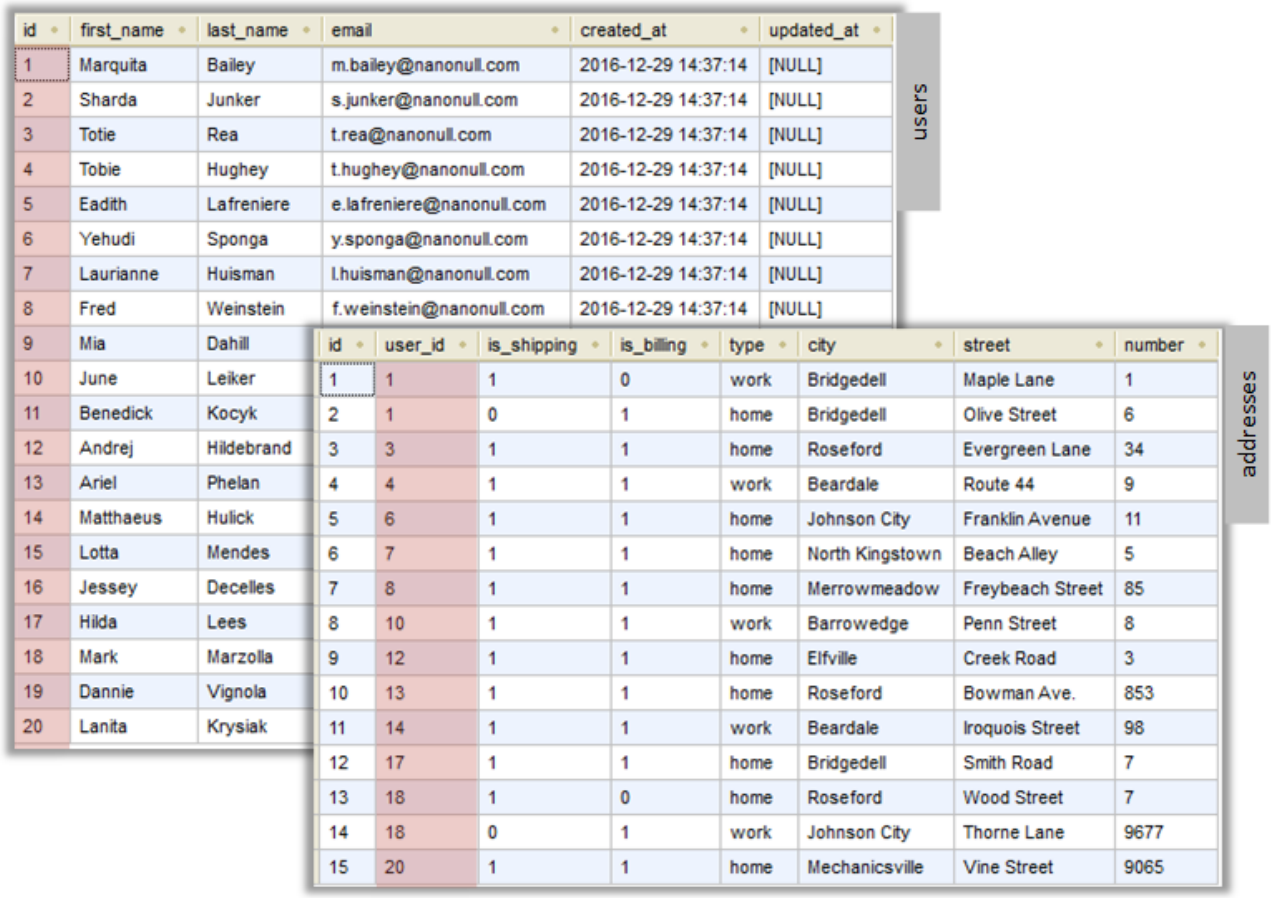



*JoinDatabaseTables.mfd*

L'objectif du mappage ci-dessus est de combiner des données à partir de deux tables de base de données de source dans un seul fichier CSV cible. Comme illustré dans le diagramme de base de données ci-dessous, la première table (`users`) stocke les adresses des personnes et la deuxième table (`addresses`) stocke les noms et les adresses des personnes. Les deux tables sont liées par un champ commun (`id` dans `users` correspond à `user_id` dans `addresses`). Dans la terminologie de la base de données, ce type de relation est aussi connu comme une "relation de clé étrangère".




Pour plus de commodité, l'image ci-dessous illustre les données actuelles dans les deux tables.



users

| id | first_name | last_name | email | created_at | updated_at |
|---|---|---|---|---|---|
| 1 | Marquita | Bailey | m.bailey@nanonull.com | 2016-12-29 14:37:14 | [NULL] |
| 2 | Sharda | Junker | s.junker@nanonull.com | 2016-12-29 14:37:14 | [NULL] |
| 3 | Totie | Rea | t.rea@nanonull.com | 2016-12-29 14:37:14 | [NULL] |
| 4 | Tobie | Hughey | t.hughey@nanonull.com | 2016-12-29 14:37:14 | [NULL] |
| 5 | Eadith | Lafreniere | e.lafreniere@nanonull.com | 2016-12-29 14:37:14 | [NULL] |
| 6 | Yehudi | Sponga | y.sponga@nanonull.com | 2016-12-29 14:37:14 | [NULL] |
| 7 | Laurianne | Huisman | l.huisman@nanonull.com | 2016-12-29 14:37:14 | [NULL] |
| 8 | Fred | Weinstein | f.weinstein@nanonull.com | 2016-12-29 14:37:14 | [NULL] |
| 9 | Mia | Dahill | | | |
| 10 | June | Leiker | | | |
| 11 | Benedick | Kocyk | | | |
| 12 | Andrej | Hildebrand | | | |
| 13 | Ariel | Phelan | | | |
| 14 | Matthaeus | Hulick | | | |
| 15 | Lotta | Mendes | | | |
| 16 | Jessey | Decelles | | | |
| 17 | Hilda | Lees | | | |
| 18 | Mark | Marzolla | | | |
| 19 | Dannie | Vignola | | | |
| 20 | Lanita | Krysiak | | | |

addresses

| id | user_id | is_shipping | is_billing | type | city | street | number |
|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 0 | work | Bridgedell | Maple Lane | 1 |
| 2 | 1 | 0 | 1 | home | Bridgedell | Olive Street | 6 |
| 3 | 3 | 1 | 1 | home | Roseford | Evergreen Lane | 34 |
| 4 | 4 | 1 | 1 | work | Beardale | Route 44 | 9 |
| 5 | 6 | 1 | 1 | home | Johnson City | Franklin Avenue | 11 |
| 6 | 7 | 1 | 1 | home | North Kingstown | Beach Alley | 5 |
| 7 | 8 | 1 | 1 | home | Merrowmeadow | Freybeach Street | 85 |
| 8 | 10 | 1 | 1 | work | Barrowedge | Penn Street | 8 |
| 9 | 12 | 1 | 1 | home | Elfville | Creek Road | 3 |
| 10 | 13 | 1 | 1 | home | Roseford | Bowman Ave. | 853 |
| 11 | 14 | 1 | 1 | work | Beardale | Iroquois Street | 98 |
| 12 | 17 | 1 | 1 | home | Bridgedell | Smith Road | 7 |
| 13 | 18 | 1 | 0 | home | Roseford | Wood Street | 7 |
| 14 | 18 | 0 | 1 | work | Johnson City | Thorne Lane | 9677 |
| 15 | 20 | 1 | 1 | home | Mechanicsville | Vine Street | 9065 |

Chaque enregistrement d'utilisateur dans la table `users` peuvent avoir zéro ou plusieurs adresses dans la table `addresses`. Par exemple, un utilisateur peut avoir une adresse de type "home", ou deux adresses (une adresse de type "maison" et une autre adresse de type "travail"), ou aucune adresse.

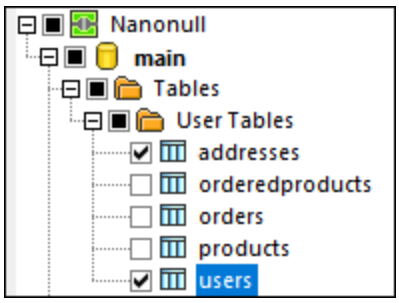
L'objectif du mappage est d'extraire les données complètes (prénom, nom de famille, e-mail, ville, rue, numéro) de tous les utilisateurs ayant au moins une adresse dans la table d'adresses. Il devrait également être possible d'extraire aisément uniquement les adresses d'un certain type (par exemple, uniquement les adresses "maison" ou les adresses "travail"). Le type d'adresses à extraire ("maison" ou "travail") doit être fourni en tant que paramètres du mappage. Les enregistrements extraits des personnes doivent être triés alphabétiquement par le nom de famille.

Les exigences de mappage seront réalisées à l'aide d'un composant Join, comme illustré dans les étapes ci-dessous.

**Note :** L'utilisation d'un composant Join n'est pas le seul moyen de joindre des tables ou des affichages de base de données. Les jointures applicables aux bases de données peuvent aussi être effectués en utilisant des instructions SELECT SQL, voir Instructions SELECT SQL en tant que tables virtuelles 264. Une différence principale entre les instructions SELECT SQL et les composants Join est que les premières sont écrites à la main, ce qui leur confère une plus grande flexibilité. Les composants Join sont une alternative plus simple si vous ne souhaitez pas rédiger des instructions SQL à la main.

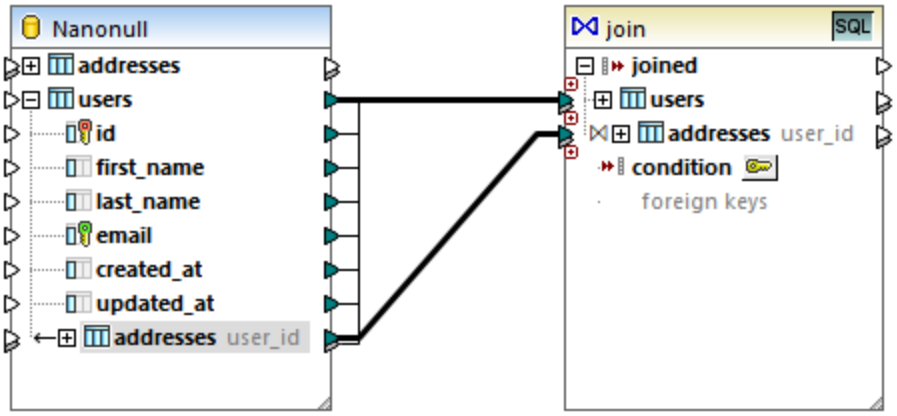
## Étape 1: Ajouter la base de données de source

1. Dans le menu **Insérer**, cliquer sur **Base de données**. (En alternative, cliquer sur la touche de la barre d'outils **Insérer la base de données** ).
2. Sélectionner "SQLite" en tant que type de base de données, et cliquer sur **Suivant**.
3. Chercher le fichier **Nanonull.sqlite** disponible dans le dossier : **<Documents>\Altova\MapForce2024\MapForceExamples\Tutorial**, et cliquer sur **Se connecter**.
4. Sur invitation, choisir les tables `addresses` et `users`.



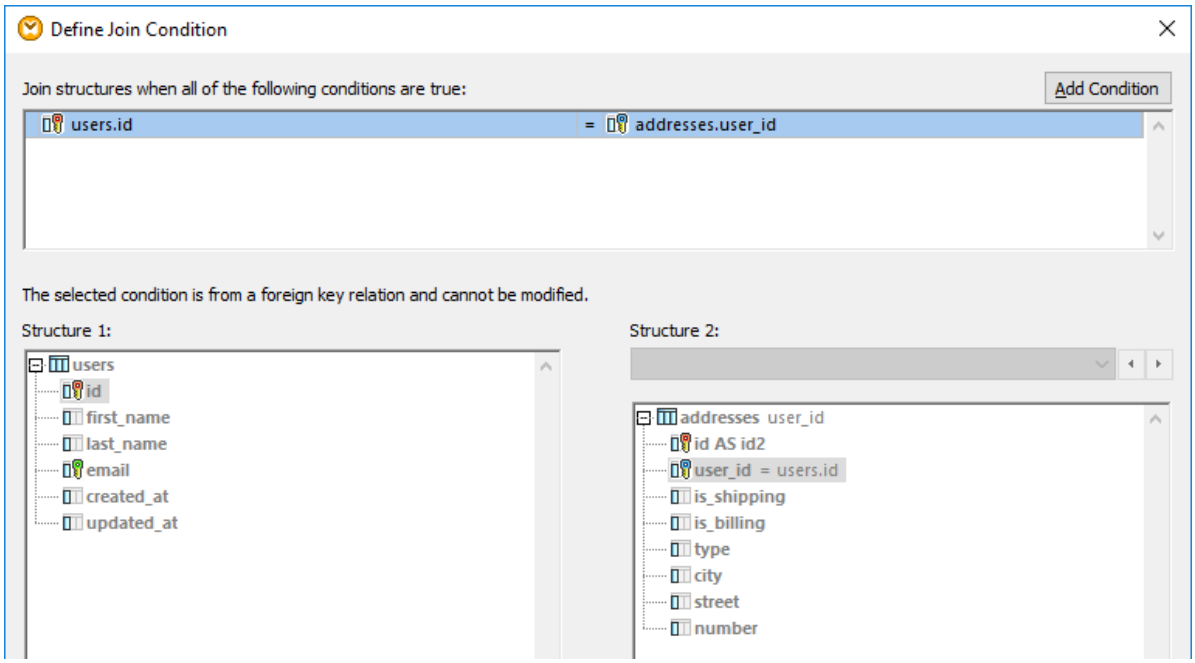

## Étape 2: Ajouter le composant join

1. Dans le menu **Insérer**, cliquer sur **Join**. (En alternative, cliquer sur touche de la barre d'outils **Join** ).
2. Tirer la connexion depuis la table `users` vers la première entrée du composant join.
3. Agrandir la table `users` et tirer une connexion depuis la table `addresses` (enfant de `users`) vers la deuxième entrée du composant join. La touche vous permet d'ajouter plus de tables si nécessaire ; néanmoins, dans cet exemple, seules deux tables sont jointes.



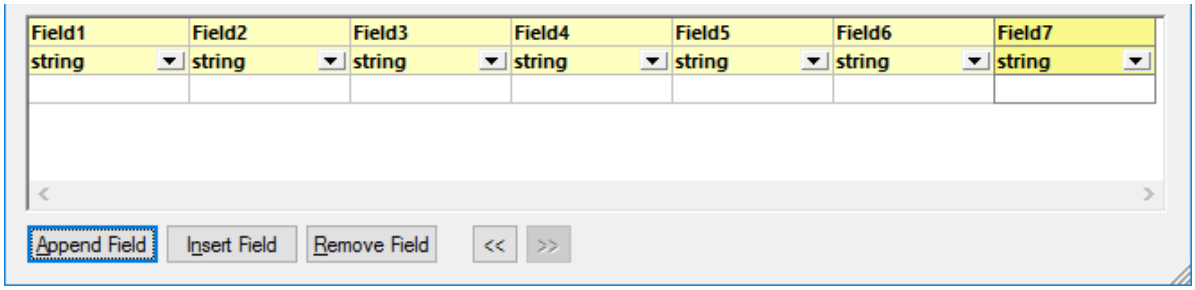

**Note :** Il est également possible d'ajouter la connexion directement depuis la table `addresses` (celui qui n'est pas enfant d'`users`) ; néanmoins, dans ce cas, les conditions join devront être définies manuellement, comme décrit dans Ajouter les conditions Join 428. Dans l'objectif de cet exemple, veuillez vous assurer de créer les connexions comme indiqué ci-dessus. Cela garantit que la condition join requise est créée automatiquement.

4. Cliquer sur la touche **Définir la condition Join** disponible sur le composant join. Veuillez noter que la condition join a été créée automatiquement (`users.id = addresses.user_id`).

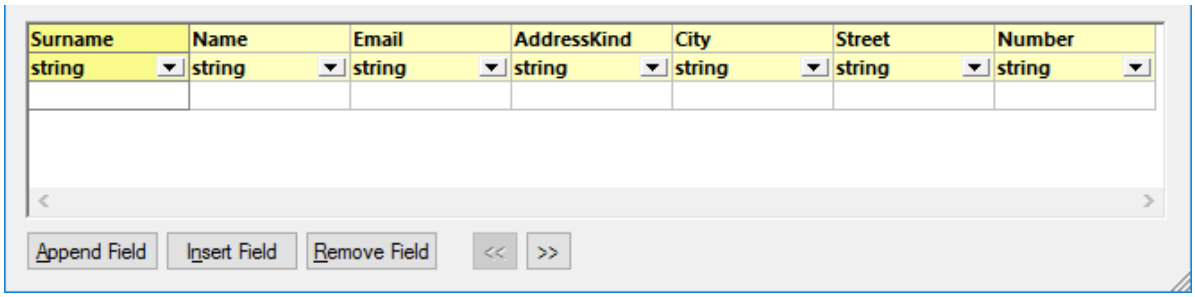



## Étape 3: Ajouter le composant CSV cible

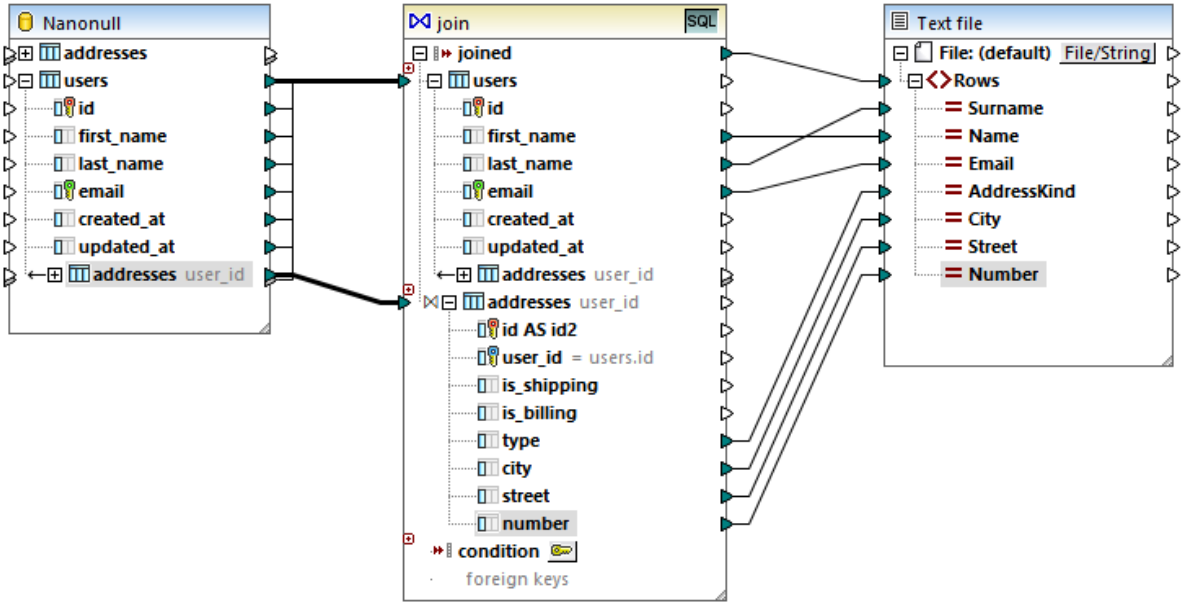
1. Dans le menu **Insérer**, cliquer sur **Fichier texte**. (En alternative, cliquer sur la touche de la barre outils**Insérer fichier de texte** ).
2. Lorsque vous êtes invité à choisir un mode de traitement de texte, choisir **Utiliser le traitement simple pour le CSV standard...** .
3. Cliquer sur **Apposer champ** plusieurs fois pour créer sept champs CSV. Laisser les autres paramètres tels qu'ils sont.




4. Double-cliquer sur la cellule de titre de chaque champ pour lui donner un nom descriptif (cela rendre votre mappage plus facile à lire).

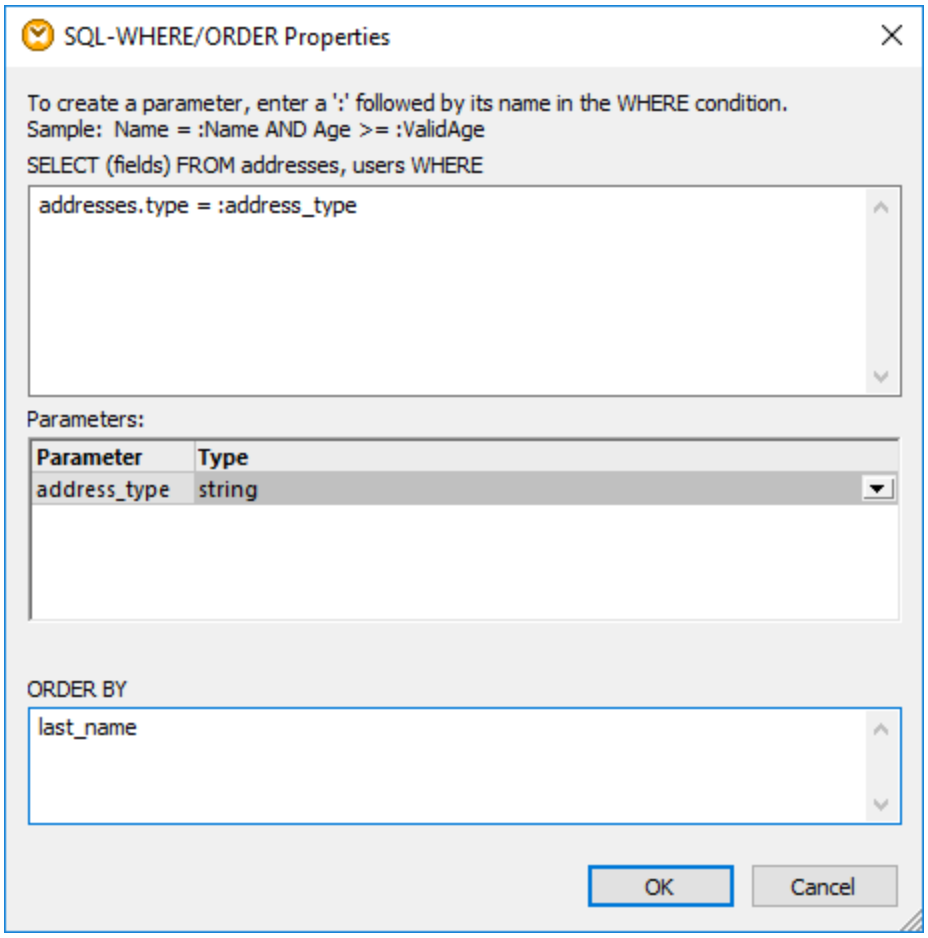



5. Tirer les connexions de mappage entre le composant Join et le composant CSV comme indiqué ci-dessous. La connexion entre l'item `joined` du composant join et l'item `Rows` du composant de cible "créer autant d'enregistrements (lignes) dans la cible étant donné qu'il y a des enregistrements qui satisfont à la condition join".

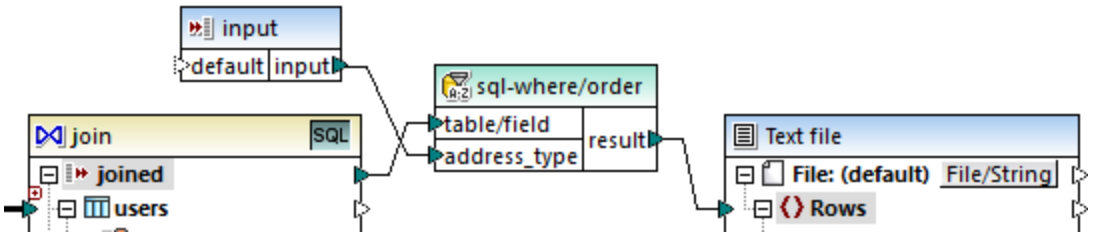



## Étape 4: Ajouter la condition WHERE/ORDER SQL et le paramètre d'entrée

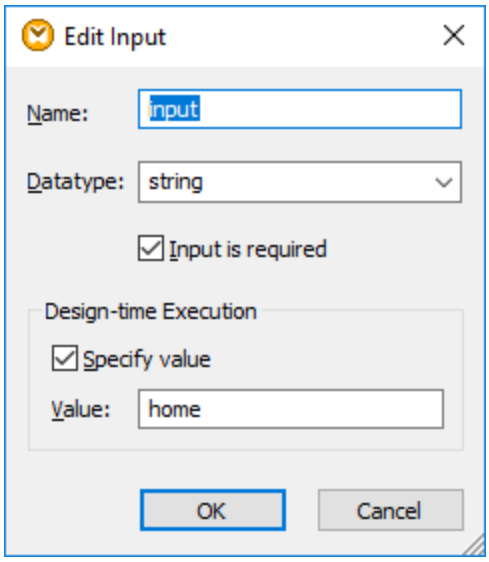
1. Cliquer avec la touche droite dans la connexion entre l'item `joined` du composant Join et l'item `Rows` du composant CSV cible, et choisir **Insérer WHERE/ORDER SQL**.
2. Saisir les clauses WHERE et ORDER BY comme indiqué ci-dessous.




3. Dans le mappage, ajouter un composant d'entrée (en utilisant la commande de menu **Insérer | Insérer entrée**) et connecter sa sortie au paramètre `address_type` créé dans l'étape précédente.




4. Double-cliquer sur le composant d'entrée et le configurer comme indiqué ci-dessous. Une valeur de design-time est nécessaire (dans ce cas, "home") pour consulter la sortie de mappage dans MapForce. Si vous souhaitez que l'aperçu extraie les adresses de travail, remplacer cette valeur par "work".




## L'explication de mappage

La condition join créée automatiquement dans l'étape 2 garantit que seuls des enregistrements qui satisfont à la condition join `users.id = addresses.user_id` sont copiés dans la cible. La condition join est ajoutée automatiquement parce que les deux tables sont liées par une relation étrangère clé et les connexions de mappage ont été tirées de manière appropriée. Pour plus d'informations concernant les relations de table, voir Gérer des relations de base de données[268]. Puisque cet exemple a été fait en utilisant les relations de table déjà existantes, vous n'avez pas dû définir de conditions join manuellement . Pour consulter un exemple qui vous montre comment définir des conditions join manuellement, voir Exemple : Créer un rapport CSV depuis plusieurs tables[361].

Les deux tables de source proviennent de la même base de données et du même composant, ce qui permet de réunir des avantages du mode SQL ( SQL ). Puisque le mode SQL est activé, l'opération join est entreprise par la base de données, et non pas par MapForce. En d'autres termes, une instruction INNER JOIN est générée internement par MapForce et est envoyée à la base de données pour l'exécution. Le type du join (INNER JOIN) est indiqué par l'icône **Join intérieur** ⋈ en face de la table **addresses** dans le composant Join. Vous pouvez aussi modifier le type join en LEFT OUTER JOIN ⋈, comme décrit dans Changer le Mode Join[350] . Noter, néanmoins que la modification du mode join n'a pas d'effet sur la sortie de cet exemple.

Le composant WHERE/ORDER SQL ajouté dans l'étape 4 permet le filtrage (afin d'extraire soir les adresses home (maison) soit work (travail)) et trier les enregistrements. Veuillez noter que la clause WHERE a créé un paramètre **`:address_type`** de type `string`. Ce paramètre permet de fournir le type d'adresse (home (maison) soit work (travail) depuis le mappage. Pour plus d'information à propos de WHERE/ORDER SQL, voir Filtrer et trier les données de base de données (SQL WHERE/ORDER)[365].

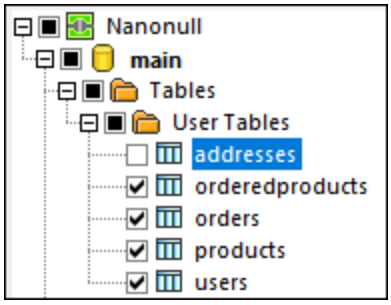
Enfin, le composant d'entrée permet de fournir la valeur de paramètre réelle lorsque le mappage est exécuté. Veuillez noter que, lorsque le mappage est exécuté en-dehors de MapForce (par exemple, lorsqu'il est exécuté par MapForce Server sur un autre appareil), l'entrée doit être fournie au moment de l'exécution du mappage en tant que paramètre de ligne de commande, afin que la valeur design-time mentionnée ci-dessus soit ignorée. Pour plus d'informations, voir Fournir des paramètres au mappage[398].

### 4.2.7.3 Exemple : Créer un rapport CSV depuis plusieurs tables

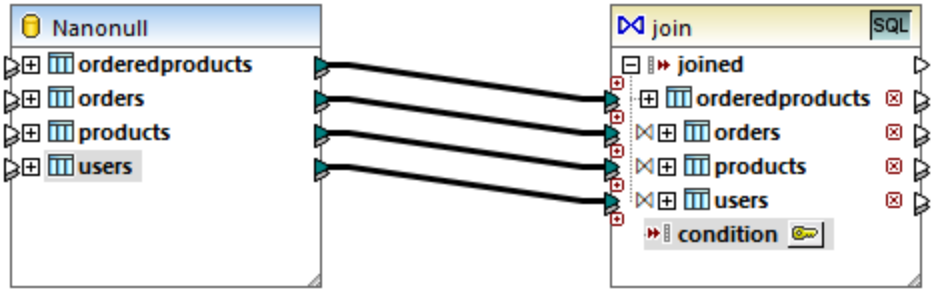
Cet exemple montre comment joindre plusieurs tables de base de données afin d'extraire des données dans un seul rapport en format CSV. La base de données utilisée dans cet exemple est appelée **Nanonull.sqlite** et est disponible dans le chemin suivant : **<Documents>\Altova\MapForce2024\MapForceExamples\Tutorial**. Cette base de données stocke des informations à propos d'un commerce fictif (comprenant des commandes, des produits, des utilisateurs et leurs adresses). Comme cela est souvent le cas avec des base de données relationnelles, l'information est normalisée et distribuée sur plusieurs tables. Par exemple, la table `users` stocke des données personnelles d'utilisateur (qui contiennent le prénom, le nom de famille et l'e-mail). La base de donnée stocke également des informations concernant des produits commandés par les utilisateurs, dans deux tables différentes : `orders` (qui inclut l'ID unique de la commande et la date à laquelle elle a été passée) et `orderedproducts` (qui inclut une liste de produits commandés, et leurs quantités). De plus, les noms des produits eux-mêmes sont stockés dans une table séparée appelée `products`.

Le but de cet exemple est de produire un rapport sur la base des données extraites depuis des tables variées, afin de montrer clairement qui a commandé certains produits, quand et en quelles quantités. Pour atteindre l'objectif du mappage, suivre les étapes suivantes :

1. Dans le menu **Insérer**, cliquer sur **Base de données**.
2. Lorsque vous êtes invité à choisir un type de base de données, cliquer sur **SQLite**, puis cliquer sur **Suivant**.
3. Chercher la base de données **Nanonull.sqlite** mentionnée ci-dessus et cliquer sur **Se connecter**.
4. Sur invitation, choisir les tables `orderedproducts`, `orders`, `products`, et `users`, et cliquer sur **OK**.




5. Ajouter un composant Join au mappage et créer quatre items `nodes/rows` en cliquant sur la touche **Ajouter entrée** ( ⊡ ).
6. Connecter les quatre tables depuis le composant de base de données dans les items entrée correspondants du composant Join.



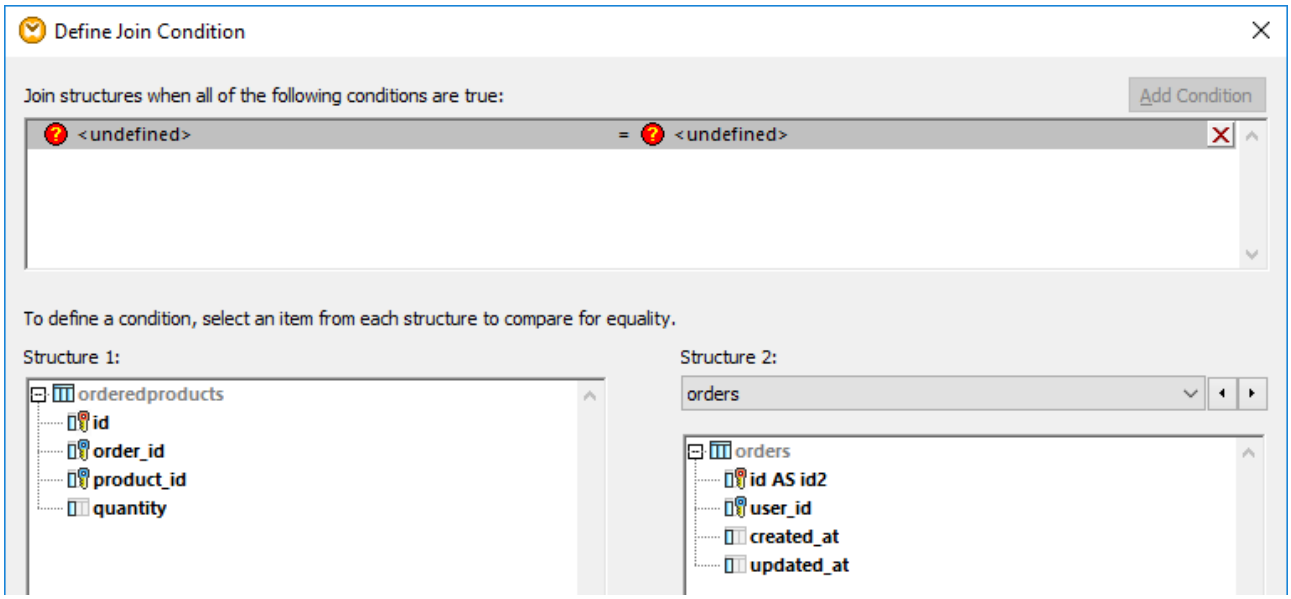

**Note :** Dans un autre scénario, vous pouvez connecter la table `orderedproducts` au composant Join, ensuite la table `orders` (celle qui est imbriquée en-dessous, pas celle se trouvant au même niveau), etc. afin

que toutes les tables jointes sont imbriquées sous la même table "racine", voir aussi Gérer des relations de base de données[268]. Le résultat de mappage serait le même que si vous joignez les tables de cette manière. La différence est que dans cet exemple, les conditions join sont créées manuellement, alors que dans le scénario alternatif les conditions join sont créées automatiquement par MapForce. Pour consulter un exemple de jointure des tables sans devoir définir des conditions join manuellement, voir Exemple: Joindre des tables en mode SQL[352]. Un autre mappage où toutes les tables jointes se trouvent sous la même table "racine" est disponible dans le chemin suivant : **<Documents>\Altova\MapForce2024\MapForceExamples\DB_Denormalize.mfd**.

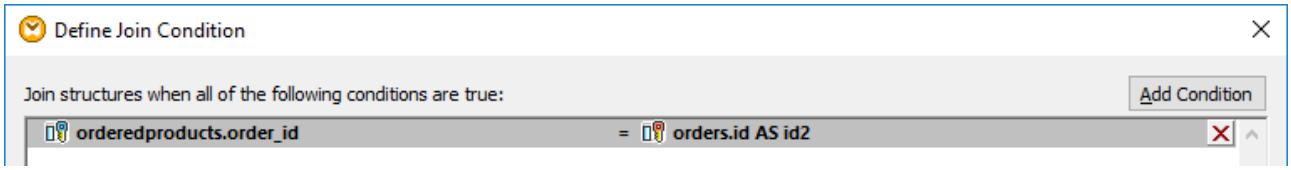
Dans cet exemple, les tables connectées au composant Join sont classées dans l'ordre suivant :

1. `orderedproducts`
2. `orders`
3. `products`
4. `users`

Cet ordre touche la manière dont les structures respectives sot affichées dans le dialogue "Définir une condition Join", lorsque vous cliquez sur la touche **Définir la Condition Join** ( ). Concrètement, la première table (`orderedproducts`) apparaît par défaut sous **Structure 1**, et la table juste après (`orders`) apparaît sous **Structure 2**.



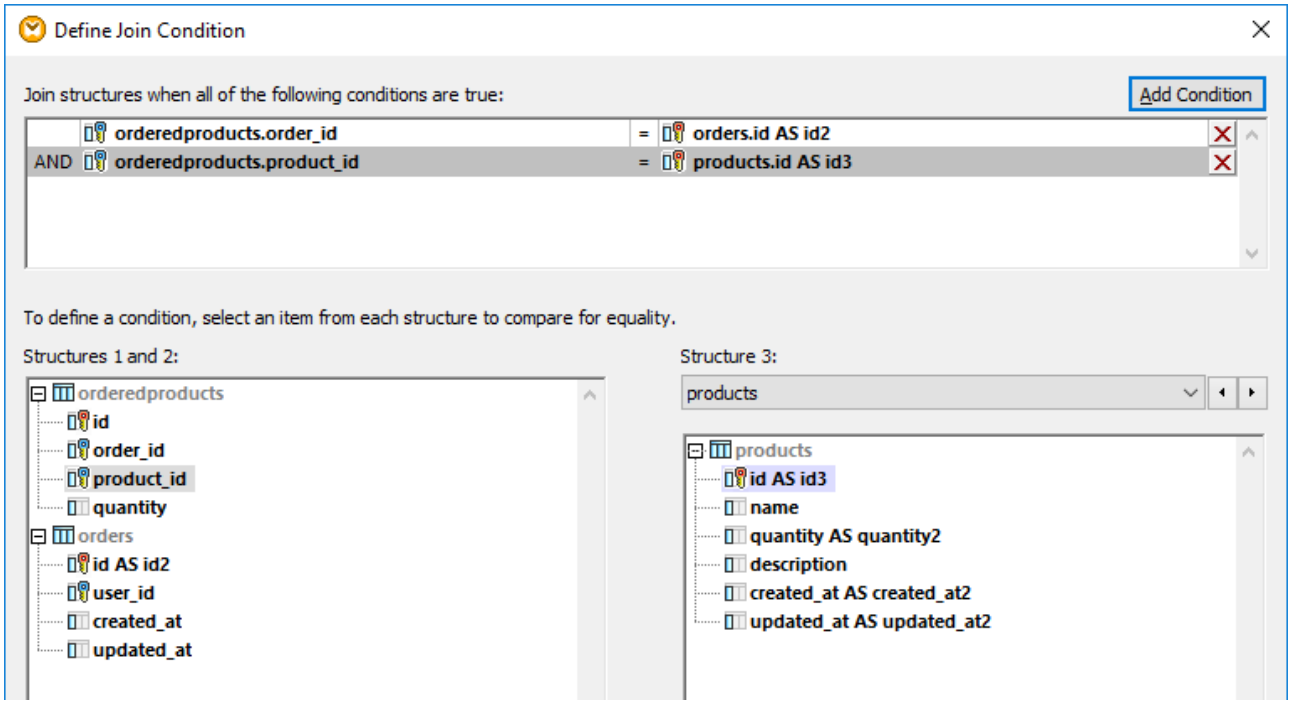

Pour définir la première condition join, cliquer sur l'item `order_id` dans le volet gauche et l'item `id` dans le volet de droite. À présent, les champs `orderedproducts.order_id` et `orders.id`. sont jumelés :



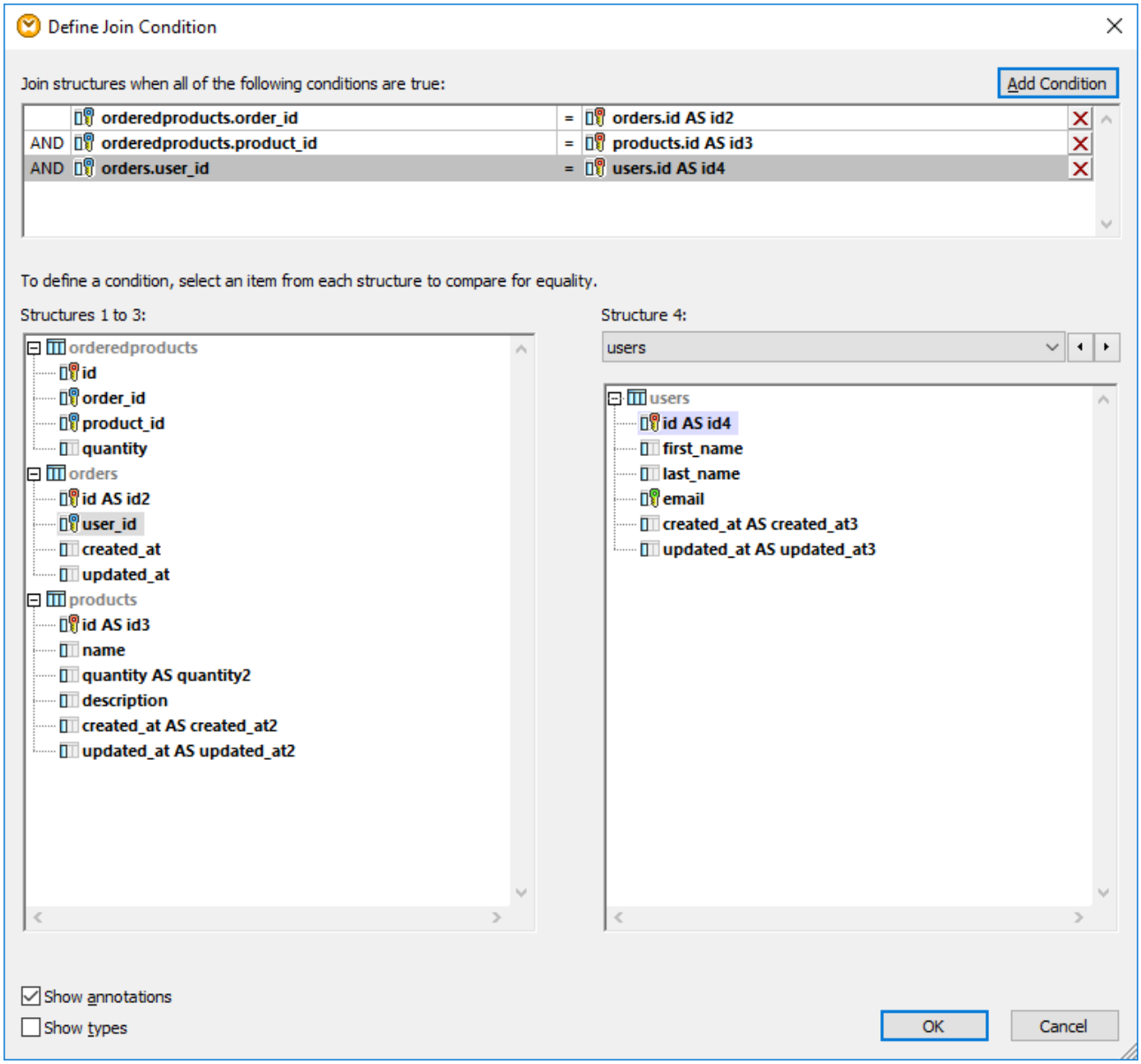

Jusqu'à présent, seules deux tables ont été jointes. Afin de définir des conditions join qui impliquent une troisième table, sélectionner la table désirée depuis la liste déroulante disponible au-dessus du volet de droite. Dans ce cas, le panneau de gauche affiche toutes les tables qui se produisent *avant* dans le composant Join.

Par exemple, si vous choisissez `products` dans le côté droit, le volet de gauche affiche `orderedproducts` et `orders` (étant donné que ces tables se produisent avant `products` dans le composant Join). Vous pouvez maintenant jumeler des paires de tables `products` avec des champs de tables les précédant (dans ce cas, `orderedproducts.product_id` et `products.id`).

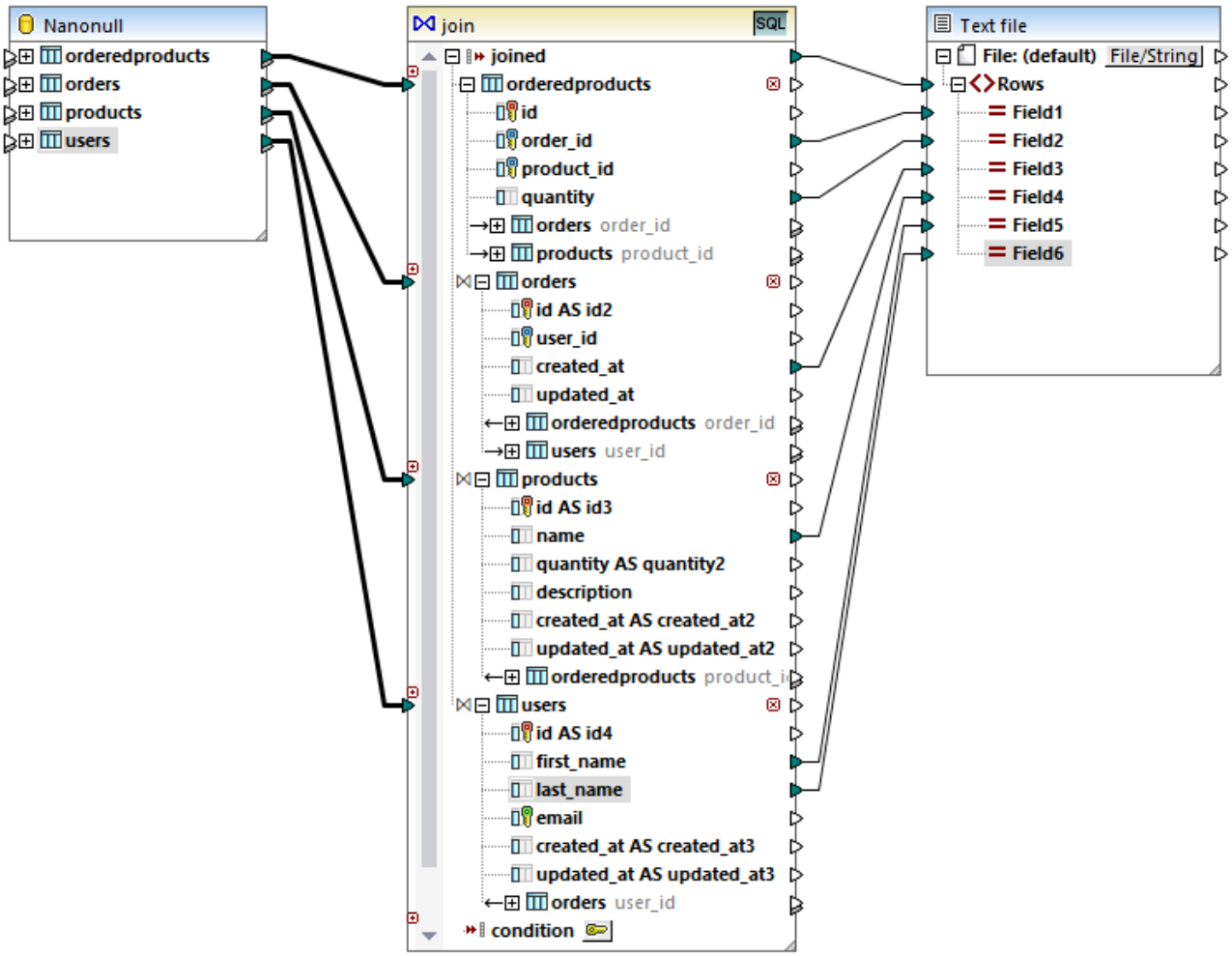



Pour joindre une quatrième table (`users`), choisir la table `users` depuis la liste déroulante. Vous pouvez maintenant jumeler les champs `orders.user_id` et `users.id`.




Maintenant que toutes les conditions join requises ont été définies, les items du composant Join peuvent être mappés dans un composant de cible. Pour terminer le mappage, ajouter un composant CSV (voir CSV et fichiers de texte 373), et connecter les items depuis le composant Join vers le composant CSV cible comme illustré ci-dessous :




Le mappage illustré ci-dessus produit un rapport (en format CSV) compilé à partir des quatre tables comprises dans la jointure, comme suit :

- ID de la commande (prélevée dans la table `orderedproducts`)
- Quantité des items commandés (prélevée dans la table `orderedproducts`)
- Moment auquel la commande a été passée (prélevé dans la table `orders`)
- Nom du produit commandé (prélevé dans la table `products`)
- Prénom et nom de famille de l'utilisateur qui a commandé le produit (prélevés dans la table `users`).

Toutes les tables présentées dans cet exemple sont jointes en utilisant le mode INNER JOIN. Pour plus d'information concernant la modification du mode join en LEFT OUTER JOIN, voir Changer le mode join 350.

## 4.2.8 Filtrer et trier les données de base de données (WHERE/ORDER SQL)

Lorsque vous souhaitez filtrer et trier des données de base de données, utilisez un composant **SQL/NoSQL-WHERE/ORDER**. Ceci vous permet de saisir manuellement une clause SQL WHERE qui filtre les données. En

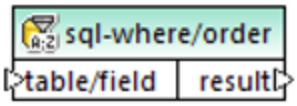
option, vous pouvez aussi spécifier une clause ORDER BY si vous souhaitez trier les enregistrements par un champ de base de données particulier, dans un ordre ascendant ou descendant.

Le composant **SQL/NoSQL-WHERE/ORDER** doit être connecté à une table ou un champ du composant de mappage de base de données. Il est également possible de connecter un composant **SQL/NoSQL-WHERE/ORDER** avec un composant **Join** si vous avez besoin de filtrer un ensemble rejoint ou des enregistrements. Pour plus d'informations, voir Rejoindre les données de base de données 348.

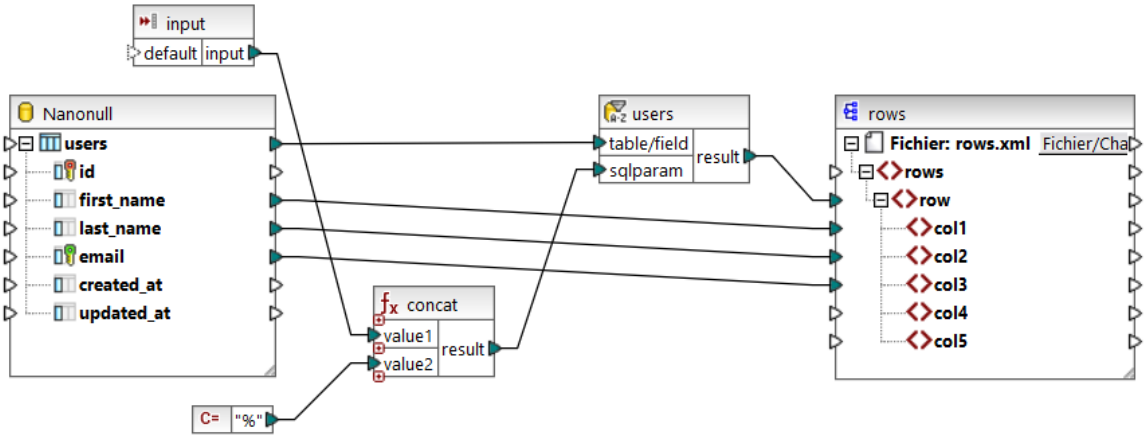
## Ajouter un composant SQL/NoSQL-WHERE/ORDER

Pour ajouter un composant **SQL/NoSQL-WHERE/ORDER** au mappage, suivez les instructions suivantes :

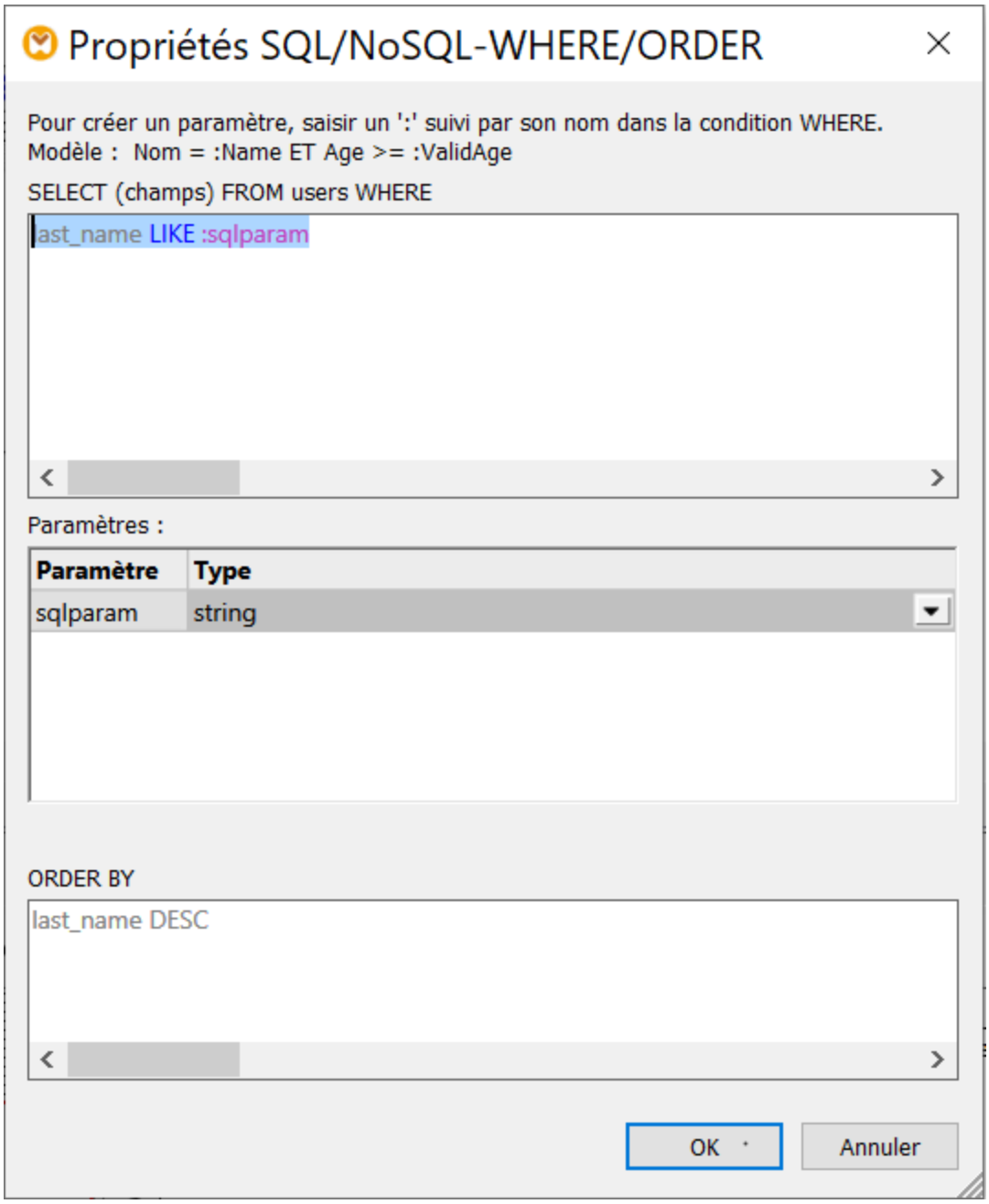
1. Allez au menu **Insérer** and cliquez sur **SQL/NoSQL-WHERE/ORDER**. Par défaut, le composant a la structure suivante :




2. Connecter une table ou un champ de base de données de source à l'item `table/field` du composant **SQL/NoSQL-WHERE/ORDER**. Vous pouvez trouver un mappage échantillon **`FilterDatabaseRecords.mfd`** (*voir la capture d'écran ci-dessous*) dans le dossier suivant : **`<Documents>\Altova\MapForce2024\MapForceExamples\Tutorial\`**. Dans ce mappage, le composant **SQL/NoSQL-WHERE/ORDER** prend les données de la table source `utilisateurs`, filtre tous ses enregistrements et sélectionne uniquement ceux où le last name commence par la lettre M (*voir l'explication dans la sous-section ci-dessous*).




3. Double-cliquez sur l'en-tête du composant **SQL/NoSQL-WHERE/ORDER**. De manière alternative, cliquez avec la touche droite sur la barre de l'en-tête et sélectionnez **Propriétés** depuis le menu contextuel. Cela ouvre la boîte de dialogue **Propriétés SQL/NoSQL-WHERE/ORDER**.



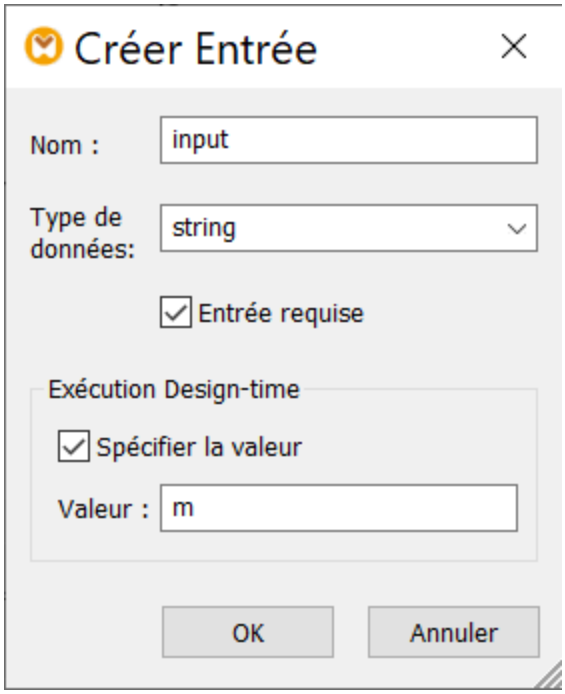

4. Saisir la clause WHERE SQL dans le champ de saisie en haut. Dans notre exemple, la clause SQL Where est la suivante : **`last_name LIKE :sqlparam`**. En option, saisir la clause ORDER BY. L'image ci-dessus illustre les clauses WHERE et ORDER BY définies dans le mappage **`FilterDatabaseRecords.mfd`** (ces paramètres sont expliqués plus bas). Pour plus d'exemples, voir Créer des clauses WHERE et ORDER BY 369.

## Paramètres dans les composants SQL/NoSQL-WHERE/ORDER

Le composant **SQL/NoSQL-WHERE/ORDER** est utilisé dans le mappage **`FilterDatabaseRecords.mfd`** a une clause WHERE suivante : **`last_name LIKE :sqlparam`**, où `last_name` se réfère au nom de champ de la base

de données dans la table connectée ; `LIKE` est un opérateur SQL ; `:sqlparam` crée un paramètre appelé `sqlparam` dans le mappage.

Les paramètres dans le composant **SQL/NoSQL-WHERE/ORDER** sont optionnels. Ils sont utiles si vous souhaitez passer une valeur dans la clause WHERE depuis le mappage. Sans paramètres, la clause WHERE ci-dessus pourrait avoir été écrite comme suit : `Last LIKE "M%"`. Ceci extrairaient toutes les personnes dont le nom de famille commence par la lettre M. En vue de rendre cette requête même plus flexible, nous avons ajouté un paramètre au lieu de `"M%"`. Cela permet de fournir d'autres lettres depuis le mappage, par ex., D, et donc d'extraire les personnes dont le nom de famille commence avec D simplement en changeant une constante, ou un paramètre d'entrée de mappage. Dans le mappage ci-dessus, la lettre d'entrée provient d'un simple composant d'entrée appelé `input`. Si vous double-cliquez sur la barre de titre de ce composant et que vous ouvrez ses propriétés, vous remarquerez que `m` est donné en tant que valeur d'exécution au moment du design (*voir la capture d'écran ci-dessous*) :



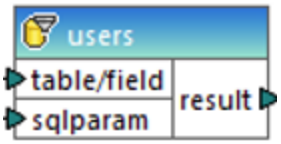
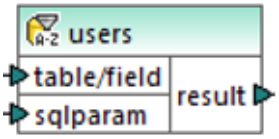

Dans le mappage, le caractère générique % est fourni par un constant. Ce caractère générique est concaténé avec une valeur de paramètre à l'aide de la fonction **`concat`**. L'avantage étant que vous ne devez pas saisir de caractère générique dans la ligne de commande si le mappage est exécuté dans un autre environnement (par ex., le serveur MapForce Server).

## Apparence de composants SQL/NoSQL-WHERE/ORDER

Des composants **SQL/NoSQL WHERE/ORDER** changent leur apparence dépendant de leurs paramètres y définis. Ainsi, vous pouvez consulter rapidement directement depuis le mappage ce que le composant **SQL/NoSQL WHERE/ORDER** fait (*voir la table ci-dessous*)

| | |
|---|---|
|  | Une clause WHERE avec un paramètre a été définie. |

| | |
|---|---|
|  | Une clause WHERE avec un paramètre a été définie. Le paramètre name est visible sous l'item `table/field`. |
|  | Une clause WHERE avec un paramètre a été définie. De plus, une clause ORDER BY a été définie. Le tri est indiqué par l'icône de tri A-Z. |

Placer le curseur de la souris sur l'en-tête **SQL/NoSQL WHERE/ORDER** ouvre une infobulle affichant les différentes clauses qui ont été définies.

### 4.2.8.1 Créer des clauses WHERE et ORDER BY

Une fois qu'un composant **SQL/NoSQL-WHERE/ORDER** est ajouté au mappage, celui-ci nécessite une condition WHERE (clause) par le biais duquel vous pouvez spécifier comment vous souhaitez filtrer les données connectées au composant. La condition WHERE doit être saisie dans le dialogue **Propriétés SQL/NoSQL-WHERE/ORDER** (*voir la section précédente*).

L'écriture d'une condition WHERE depuis MapForce est semblable à l'écriture de la même clause SQL en dehors de MapForce. Veuillez utiliser la syntaxe applicable au dialecte SQL de la base de données correspondante. Par exemple, vous pouvez utiliser des opérateurs, des caractères génériques et des sous-sélections ou des fonctions d'agrégat. Pour créer des paramètres que vous pouvez passer depuis le mappage, saisir un double-point ( : ) suivi par le nom du paramètre.

**Note :** Lorsque vous terminez l'écriture de la clause WHERE et cliquez sur OK, MapForce valide l'intégrité de l'instruction SQL finale. Un dialogue vous relance en cas d'apparition d'erreurs de syntaxe.

La table ci-dessous recense certains des opérateurs typiques qui peuvent être utilisés dans la clause WHERE :

| Opérateur | Description |
|---|---|
| = | Égal |
| <> | Non égal |
| < | Inférieur à |
| > | Supérieur à |
| >= | Supérieur à/égal |
| <= | Inférieur à/égal |
| IN | Extrait une valeur connue d'une colonne |
| J'AIME | Cherche un motif particulier |

| Opérateur | Description |
|---|---|
| BETWEEN | Cherche dans une fourchette |

Utilise le caractère générique % (pourcentage) pour représenter tout nombre de caractères dans un motif. Par exemple, pour extraire tous les enregistrements se terminant en "r" depuis un champ appelé lastname, utilisez l'expression suivante :

```
lastname = "%r"
```

Lorsque vous requêtez des bases de données qui prennent en charge le stockage et la requête des données de base de données XML (par exemple, IBM DB2, Oracle, SQL Server), vous pouvez utiliser des fonctions et des mots-clés XML applicables à cette base de données particulière, par exemple :

```
xmlexists('$c/Client/Address[zip>"55116"]' passing USER.CLIENTS.CONTACTINFO AS "c")
```

Voir aussi Exemple : Extraire des données depuis les colonnes IBM DB2 de type XML 318.

En option, si vous souhaitez trier les enregistrements par un champ particulier, ajoutez une clause ORDER BY dans le champ de texte correspondant du dialogue **Propriétés SQL/NoSQL-WHERE/ORDER**. Pour trier par plusieurs champs, séparez les noms de champ avec des virgules. Pour changer l'ordre du tri, utilisez les mots-clés ASC et DESC. Par exemple, la clause ORDER BY suivante extrait des enregistrements par lastname, puis par firstname, dans l'ordre descendant :

```
lastname, firstname DESC
```

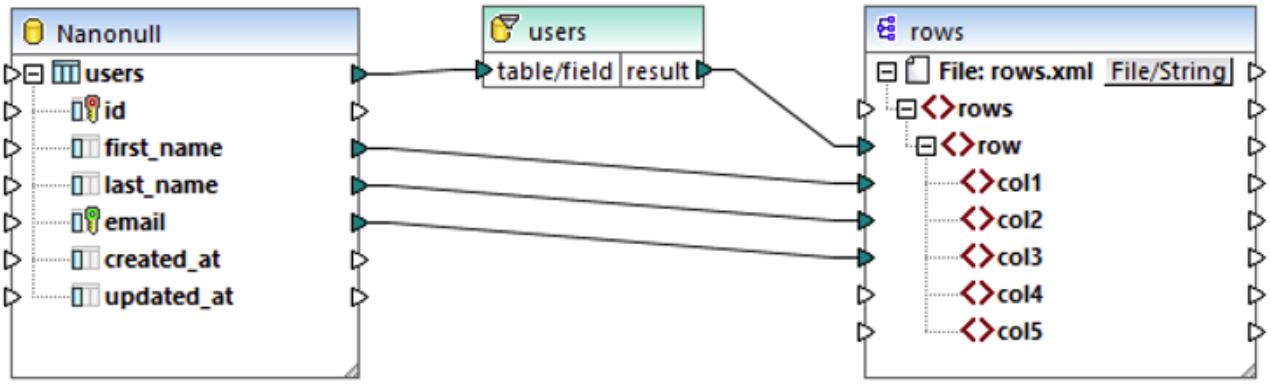
## Exemple 1

La condition WHERE suivante est attachée à la table utilisateurs du composant de base de données **Nanonull.sqlite**. Elle extrait les enregistrements dans lesquels last_name sont supérieurs à la lettre "M". En d'autres termes, il extrait tous les noms depuis "Marzolla".

```
last_name > "M"
```

Notez comment les connexions sont placées :

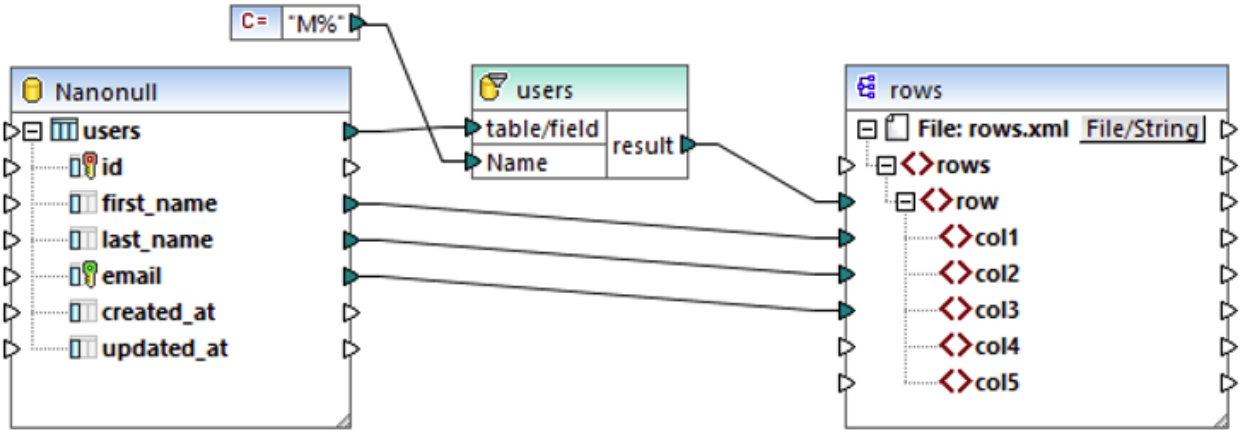
- La connexion vers table/field provient dans la table que vous souhaitez interroger, (utilisateurs dans ce cas).
- La sortie result est connectée à un item "parent" des champs qui sont requêtés/filtrés (dans ce cas l'item row).




## Exemple 2

La condition WHERE suivante crée un paramètre `param` qui apparaît ensuite dans le composant WHERE/ORDER SQL dans le mappage.

```
last_name LIKE :param
```

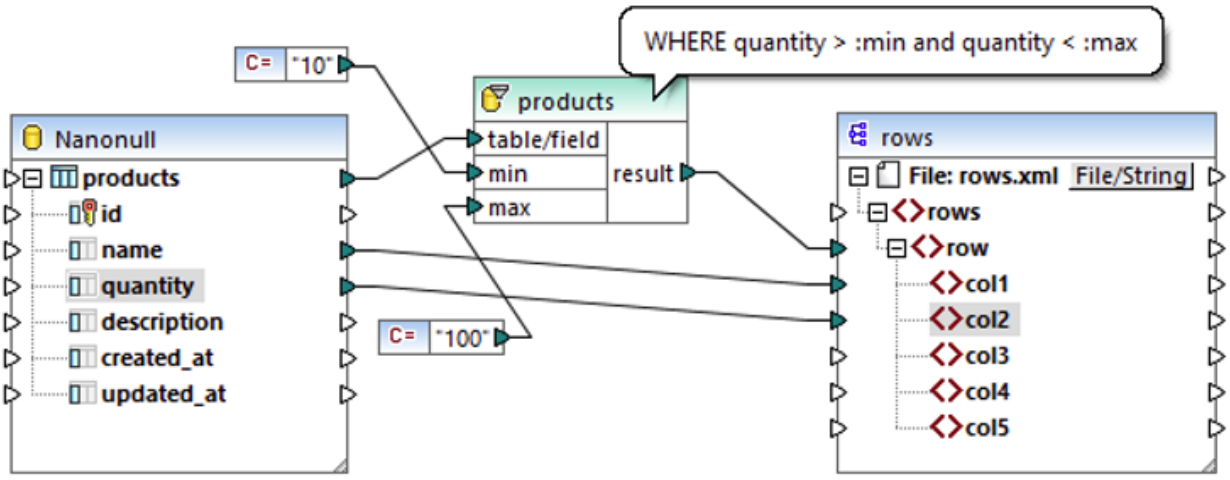



Le composant de constante `%S` fournit la valeur du paramètre `param`. Le caractère générique % dénote un nombre quelconque de caractères. Le mappage cherche donc un motif dans la colonne `last_name` (tous les noms de famille commençant par "M").

## Exemple 3

La condition WHERE suivante crée deux paramètres, `min` et `max`, auxquelles les valeurs actuelles de `quantity` sont comparées. Les valeurs min et max sont fournies par deux composants constants du mappage.

```
quantity > :min and quantity < :max
```




La condition WHERE dans cet exemple peut aussi être écrite en utilisant l'opérateur BETWEEN :

```
quantity BETWEEN :min and :max
```

# 4.3 Fichiers CSV et Texte

MapForce propose une prise en charge pour le mappage de données depuis ou vers des formats de fichier basés sur texte comme des fichiers de textes CSV (valeurs séparées par des virgules) et FLF (champ à longueur fixe). Par exemple, vous pouvez créer des transformations de données comme :

- schéma XML de/vers des formats de fichier plat
- base de données de/vers des formats de fichier plat

Veuillez noter que, en ce qui concerne les CSV, vos fichiers peuvent contenir non seulement des virgules en tant que délimiteurs, mais aussi des tabulateurs, des point-virgules, des espaces ou toute autre valeur personnalisée.

Outre les fichiers CSV et FLF, le mappage de et vers des fichiers de texte contenant des structures plus complexes ou personnalisées est possible grâce à MapForce FlexText (ce module est disponible dans l'édition MapForce Enterprise Edition). À la base, FlexText vous permet de définir la structure de vos données de texte personnalisé (en utilisant un soit-disant "modèle FlexText"), pour le mapper vers d'autres formats.

Le mappage de données de ou vers des fichiers de texte est pris en charge dans un des langages suivants : Java, C#, C++, ou BUILT-IN.

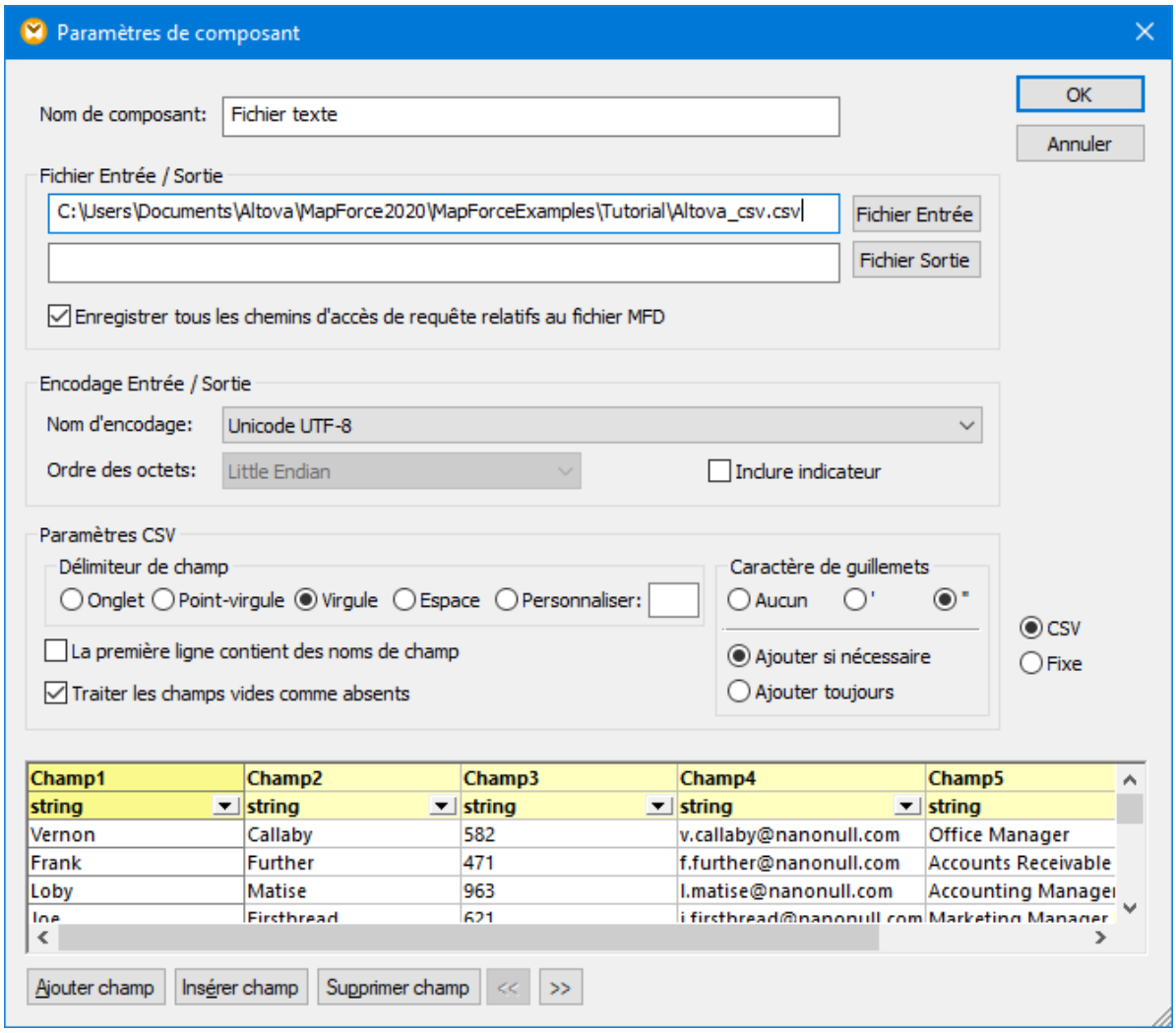
Il existe deux moyens de générer des données avec un fichier plat mappé :

- En cliquant sur l'onglet Sortie qui génère un aperçu en utilisant le moteur d'exécution Built-in. Vous pouvez aussi enregistrer les résultats de mappage en sélectionnant l'option de menu **Sortie | Enregistrer le fichier de sortie**, ou en cliquant sur l'icône .
- En sélectionnant **Fichier | Générer du code dans | Java, C#, ou C++** , et en compilant et en exécutant le code généré.

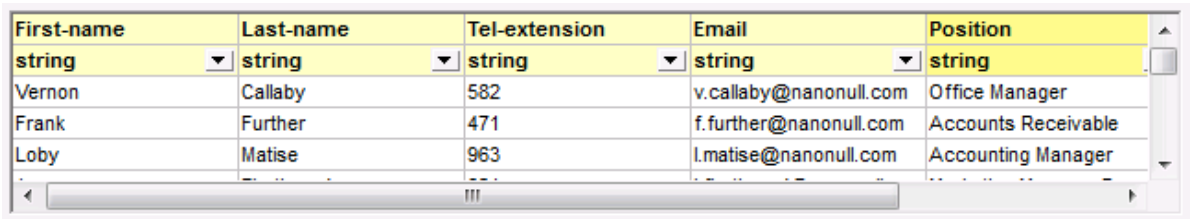
## 4.3.1 Exemple : Mapper des fichiers CSV vers XML

L'objectif de cet exemple est de créer un mappage qui lit les données depuis un simple fichier CSV et l'écrit dans un fichier XML. Les fichiers utilisés dans l'exemple sont disponibles dans le dossier **<Documents>\Altova\MapForce2024\MapForceExamples\Tutorial**.

1. Choisir un des langages de transformation : Java, C#, C++ ou BUILT-IN.
2. Ajouter un composant de fichier de texte dans la surface de mappage (dans le menu **Insérer**, cliquer sur **Fichier texte**, ou cliquer sur la touche de la barre d'outils **Insérer fichier texte**( ).
3. Dans le dialogue Paramètres de composant, cliquer sur **Fichier d'entrée** et chercher le fichier **Altova_csv.csv**. Les contenus de fichier sont à présent visibles dans la partie inférieure du dialogue. Veuillez noter que seules les 20 premières lignes du fichier de texte sont affichées lorsque vous vous trouvez en mode d'aperçu.




4. Cliquer dans l'en-tête **Champ1** et changer le texte en First-name. Procéder de même pour les autres champs, en suivant notre exemple : Champ 2 => Last-name, Champ 3 =>Tel-extension, Champ 4 => Email, Champ 5 => Position.
   ASTUCE : Appuyer sur la touche **Tab** pour feuilleter rapidement dans les champs : header1, header2 etc.

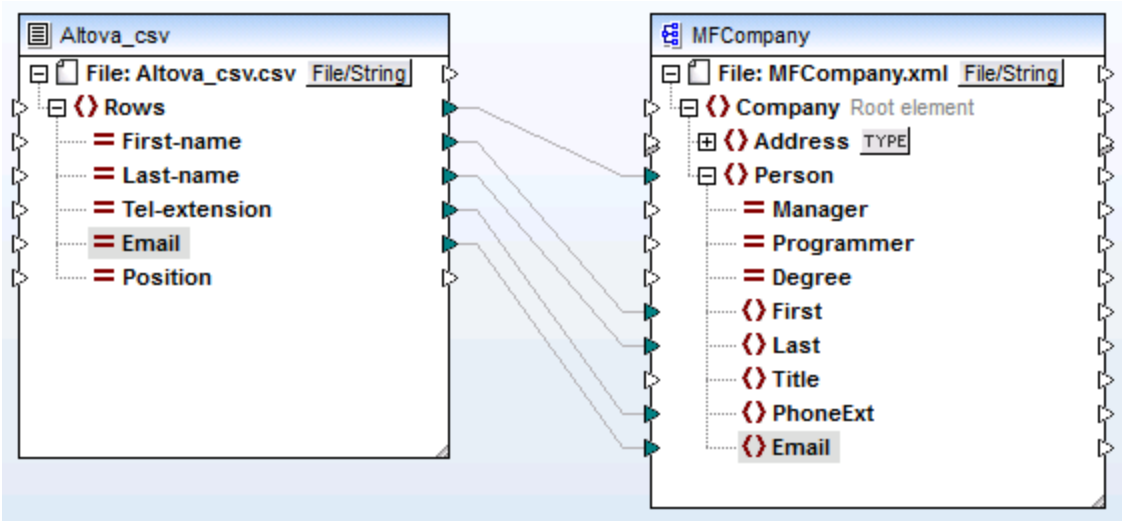



5. Cliquer sur OK.
6. Lorsque vous êtes invité à changer le nom de composant, cliquer sur "Changer le nom de composant". Le composant CSV est maintenant visible dans le mappage.
7. Ajouter **MFCompany.xsd** en tant que le composant XML cible du mappage (dans le menu **Insérer**, cliquer sur **Fichier /Schema XML**).

8. Cliquer sur **Sauter** lorsque vous êtes invité à fournir un fichier XML échantillon et choisir `Company` en tant que l'élément racine.
9. Mapper les items correspondants des deux composants, en s'assurant de mapper l'item **Rows** à l'item **Person** dans la cible de schéma.

> Le connecteur provenant de l'item `Rows` dans le composant CSV vers l'item `Person` dans le schéma est essentiel, étant donné qu'il définit quels éléments seront itérés. Autrement dit, pour chaque ligne dans le fichier CSV, un nouvel élément `Person` sera créé dans le fichier de sortie XML.



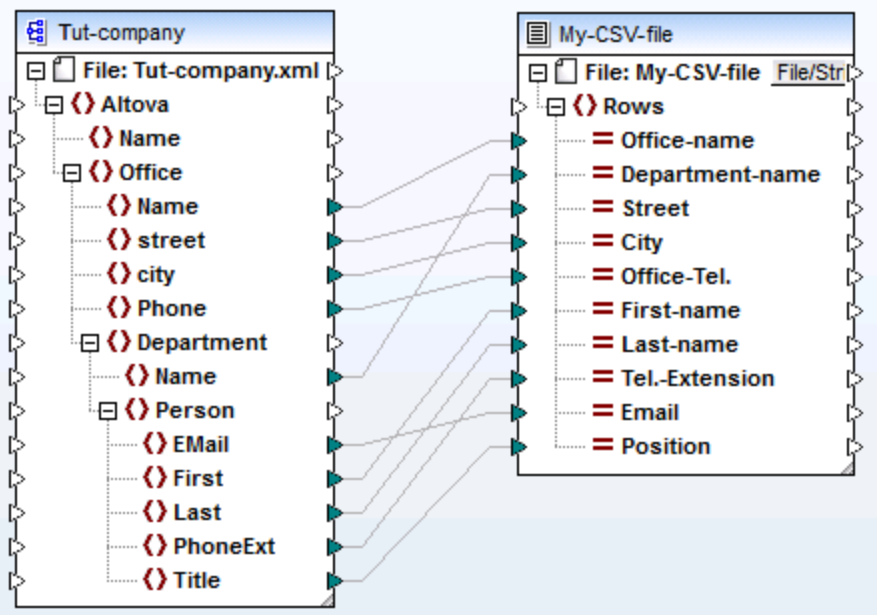

10. Cliquer sur l'onglet Sortie pour voir le résultat.

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <Company xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:
3    <Person Manager="true">
4      <First>Vernon</First>
5      <Last>Callaby</Last>
6      <PhoneExt>582</PhoneExt>
7      <Email>v.callaby@nanonull.com</Email>
8    </Person>
9    <Person Manager="true">
10     <First>Frank</First>
11     <Last>Further</Last>
12     <PhoneExt>471</PhoneExt>
13     <Email>f.further@nanonull.com</Email>
14   </Person>
15   <Person Manager="true">
```

Les données provenant du fichier CSV sont maintenant mappées avec succès à un fichier XML.

## 4.3.2 Exemple : Itérer à travers des items

Cet exemple illustre comment créer des itérations (plusieurs lignes) dans un fichier CSV cible. Le fichier de design de mappage qui accompagne cet exemple est disponible sous le chemin suivant : **<Documents>\Altova\MapForce2024\MapForceExamples\Tutorial\Tut-xml2csv.mfd**.

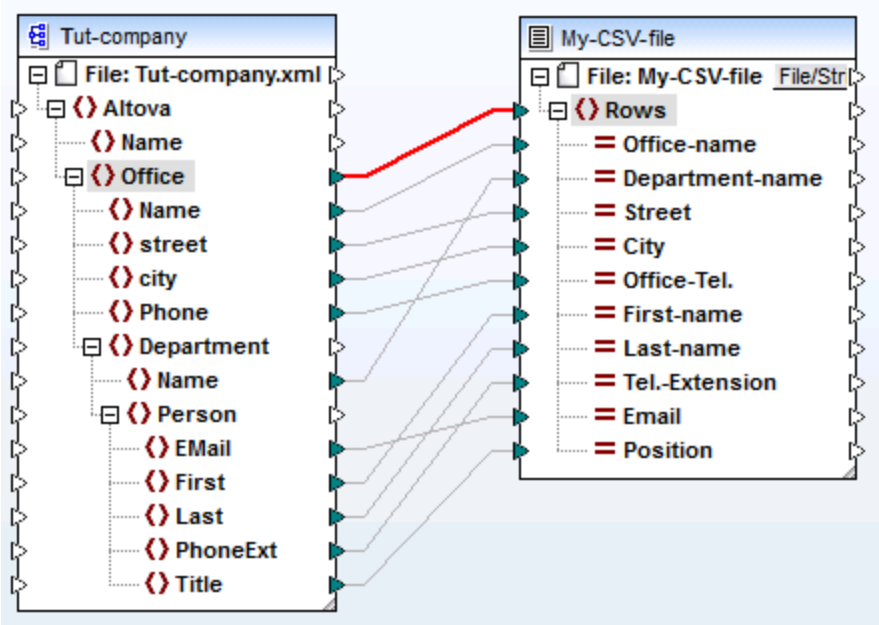



*Tut-xml2csv.mfd*

Ce mappage est resté inachevé intentionnellement. Si vous tentez de valider le fichier d'exemple en utilisant la commande de menu **Fichier | Valider le mappage**, vous remarquerez que des avertissements de validation se produiront. De même, si vous prévisualisez la sortie de mappage, une ligne individuelle est produite, ce qui peut constituer votre objectif (ou pas).

Partons du principe que votre objectif est de créer plusieurs lignes dans le fichier CSV depuis une séquence d'items dans le fichier XML. Vous pouvez obtenir ce résultat en tirant une connexion dans l'item `Rows` du fichier CSV cible.

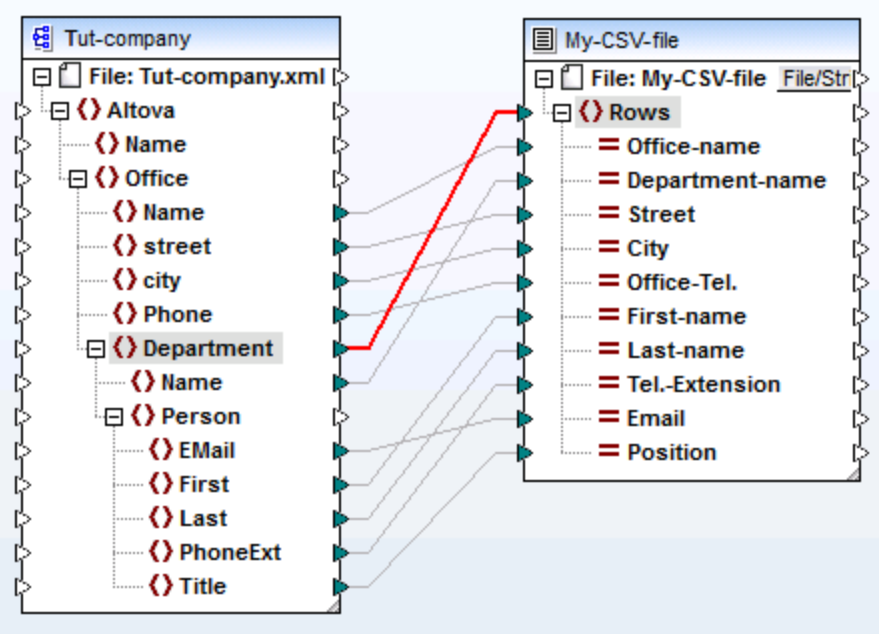
Par exemple, pour itérer dans tous les bureaux et obtenir une sortie qui apparaisse dans le fichier CSV, il est nécessaire de connecter `Office` à `Rows`. De fait, vous indiquez à MapForce que : pour chaque item `Office` du XML de source, il faut créer une ligne dans le fichier CSV cible.




Les items Rows dans le composant CSV agissent comme un itérateur pour la séquence des items qui y sont connectés. Ainsi, si vous connectez l'item Office, la sortie crée une ligne pour chaque bureau trouvé dans le XML de source.

```
1   "Microtech, Inc.",Level 1 support,Major Ave 1,Vancouver,558833
2   "Microtech Partners, Inc.",Level 2 support,Perro Bvd 1324,Otto
3
```

De manière semblable, si vous connectez **Department** à l'item **Rows**, une ligne sera produite pour chaque département trouvé dans le XML de source.




La sortie ressemblerait à l'exemple suivant :

```
1  "Microtech, Inc.",Admin,Major Ave 1,Vancouver,5588339,Clive,Clo
2  "Microtech, Inc.",Sales and Marketing,Major Ave 1,Vancouver,558
3  "Microtech, Inc.",Manufacturing,Major Ave 1,Vancouver,5588339,K
4  "Microtech, Inc.",Level 1 support,Major Ave 1,Vancouver,5588339
5  "Microtech Partners, Inc.",Admin,Perro Bvd 1324,Ottowa,3549202,
6  "Microtech Partners, Inc.",Sales and Marketing,Perro Bvd 1324,O
7  "Microtech Partners, Inc.",Level 2 support,Perro Bvd 1324,Ottow
8
```

Enfin, le mappage de `Person` à l'item `Rows` résulte dans une sortie de toutes les Persons. Dans ce cas, MapForce itérera dans tous les enregistrements comme suit : chaque Person dans chaque Department, dans chaque Office.

### 4.3.3 Exemple : Créer des hiérarchies depuis des fichiers texte CSV et de longueur fixe

Cet exemple est disponible sous le chemin suivant : **<Documents>\Altova\MapForce2024\MapForceExamples\Tutorial\Tut-headerDetail.mfd**. L'exemple utilise un fichier CSV (Orders.csv) qui dispose du format suivant :

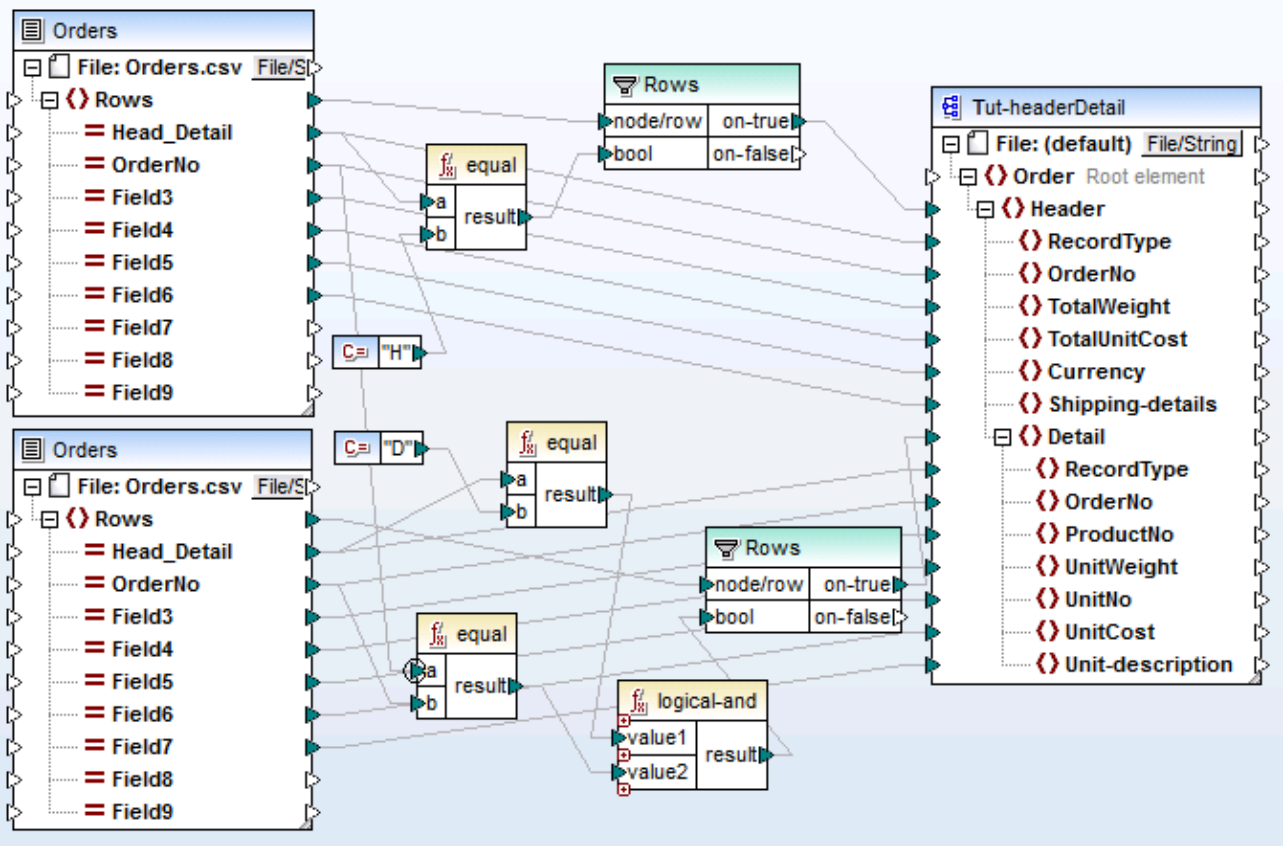
- Champ 1 : H définit un enregistrement d'en-tête et D un enregistrement de détail.
- Champ 2 : Une clé commune pour les enregistrements d'en-tête et de détail.
- Chaque enregistrement Header ou Detail sur une ligne séparée.

Les contenus du fichier Orders.csv sont indiqués ci-dessous.

```
H,111,332.1,22537.7,,Container ship,,,
D,111,A-1579-227,10,3,400,Microtome,,
D,111,B-152-427,7,6,1200,Miscellaneous,,
H,222,978.4,7563.1,,Air freight,,,
D,222,ZZ-AW56-1,10,5,10000,Gas Chromatograph,,
```

L'objectif du mappage est le suivant :

- Mapper le fichier plat CSV à un fichier XML hiérarchique
- Filtrer les enregistrements Header, conçus avec un H
- Associer les enregistrements de détail respectifs, conçus avec un D, avec chacun des enregistrements d'en-tête.




*tut-headerDetail.mfd*

Afin d'y parvenir, les enregistrements d'en-tête et de détail doivent avoir un champ en commun. Dans ce cas, le champ commun, ou la clé, est le deuxième champ du fichier CSV, c'est à dire OrderNo. Dans le fichier CSV, l'enregistrement du premier en-tête et les deux enregistrements de détail suivants contiennent la valeur commune 111.

Le fichier Orders.csv a été inséré deux fois pour rendre le mappage plus intuitif.

Le fichier de schéma **Tut-headerDetail.xsd** possède une structure hiérarchique : "Order" est l'élément racine, avec "Header" en tant que son élément enfant, et "Detail" étant un élément enfant de "Header".

Le premier fichier Orders.csv fournit les enregistrements **Header** (et tous les champs mappés) vers l'item "Header" dans le fichier cible de schéma. Le composant de filtre est utilisé pour filtrer tous les enregistrements ne commençant pas avec H. L'item **Rows** fournit ces enregistrements filtrés à l'item "Header" dans le fichier de schéma.

Le second fichier Orders.csv fournit les enregistrements **Detail** (et tous les champs mappés) en filtrant les enregistrements "Detail" qui correspondent à la clé OrderNo de l'enregistrement "Header". Cela est obtenu en :

- Comparant le champ **OrderNo** de l'enregistrement "Header" avec le même champ des enregistrements "Detail", en utilisant la fonction **equal** (le contexte de priorité 814 est défini sur le paramètre **a** pour une meilleure performance).
- Utilisant la fonction **Logical-and** non seulement pour fournir ces enregistrements "Detail" contenant le même champ OrderNo, en tant qu'enregistrement "Header".

L'item **Rows** fournit ces enregistrements filtrés aux items "Header" et "Detail" dans le fichier de schéma, par le biais du paramètre on-true du composant de filtre.

Cliquer sur l'onglet Sortie pour produire le fichier XML affiché ci-dessous. Chaque enregistrement "Header" contient ses données et tous les enregistrements "Detail" associés qui ont le même "OrderNo".

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <Order xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation
3    <Header>
4      <RecordType>H</RecordType>
5      <OrderNo>111</OrderNo>
6      <TotalWeight>332.1</TotalWeight>
7      <TotalUnitCost>22537.7</TotalUnitCost>
8      <Currency/>
9      <Shipping-details>Container ship</Shipping-details>
10     <Detail>
11       <RecordType>D</RecordType>
12       <OrderNo>111</OrderNo>
13       <ProductNo>A-1579-227</ProductNo>
14       <UnitWeight>10</UnitWeight>
15       <UnitNo>3</UnitNo>
16       <UnitCost>400</UnitCost>
17       <Unit-description>Microtome</Unit-description>
18     </Detail>
19     <Detail>
20       <RecordType>D</RecordType>
21       <OrderNo>111</OrderNo>
22       <ProductNo>B-152-427</ProductNo>
23       <UnitWeight>7</UnitWeight>
24       <UnitNo>6</UnitNo>
25       <UnitCost>1200</UnitCost>
26       <Unit-description>Miscellaneous</Unit-description>
27     </Detail>
28   </Header>
```

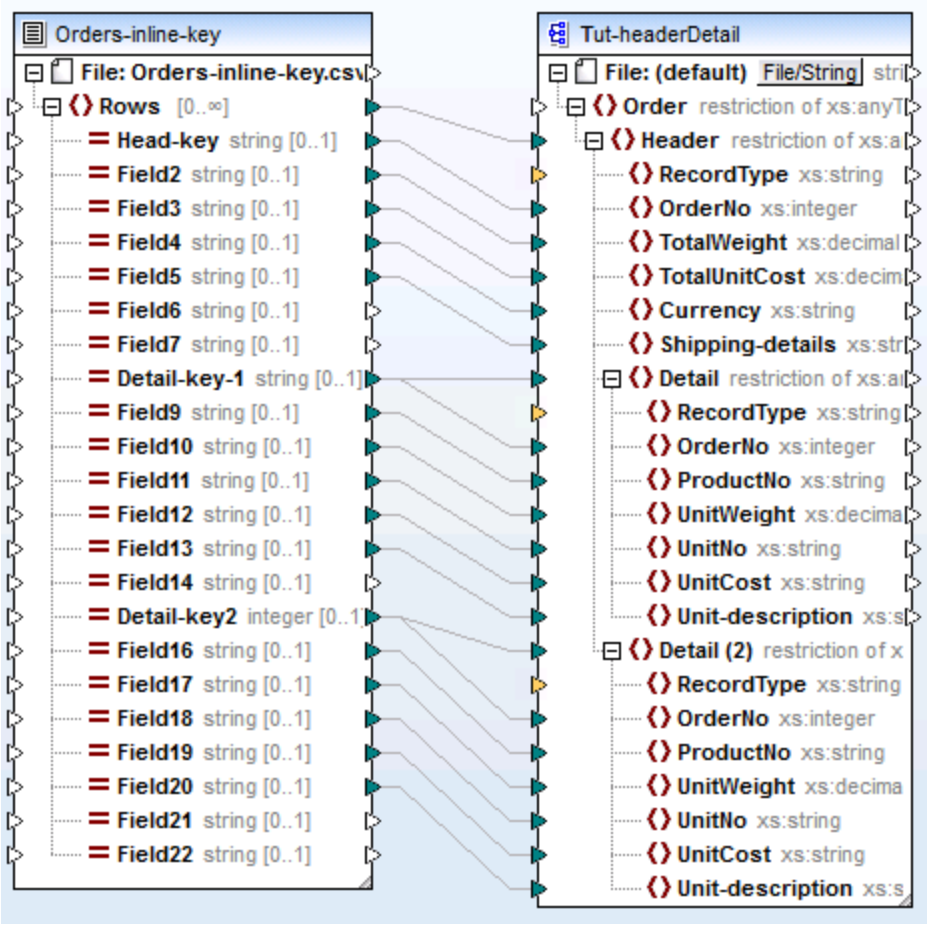
Examinons à présent un autre exemple, qui utilise un fichier CSV légèrement différent et qui est disponible dans le dossier **<Documents>\Altova\MapForce2024\MapForceExamples\Tutorial** en tant que **Head-detail-inline.mfd**. La différence est que :

- Aucun désignateur d'enregistrement (H ou D) n'est disponible
- Un champ de clé commun, le premier champ du fichier CSV, existe toujours aussi bien pour les enregistrements header que detail (Head-key, Detail-key...). Le champ est mappé sur OrderNo dans la cible de schéma
- Le champ "Header" et tous les champs "Detail" respectifs se trouvent sur la même ligne.

```
111,332.1,22537.7,,Container ship,,,111,A-1579-227,10,3,400,Microtome,,111,B-15
222,978.4,7563.1,,Air freight,,,222,ZZ-AW56-1,10,5,10000,Gas Chromatograph,,
```

Le mappage a été conçu comme suit :

- Les champs clé sont mappés vers les items OrderNo respectifs dans la cible de schéma.
- L'item "Detail" dans le fichier cible de schéma a été dupliqué et est affiché en tant que **Detail (2)**. Cela vous permet de mapper le deuxième ensemble d'enregistrements de détail vers l'item correct.
- Le résultat de ce mappage est en principe le même fichier XML qui a été produit dans le premier exemple.

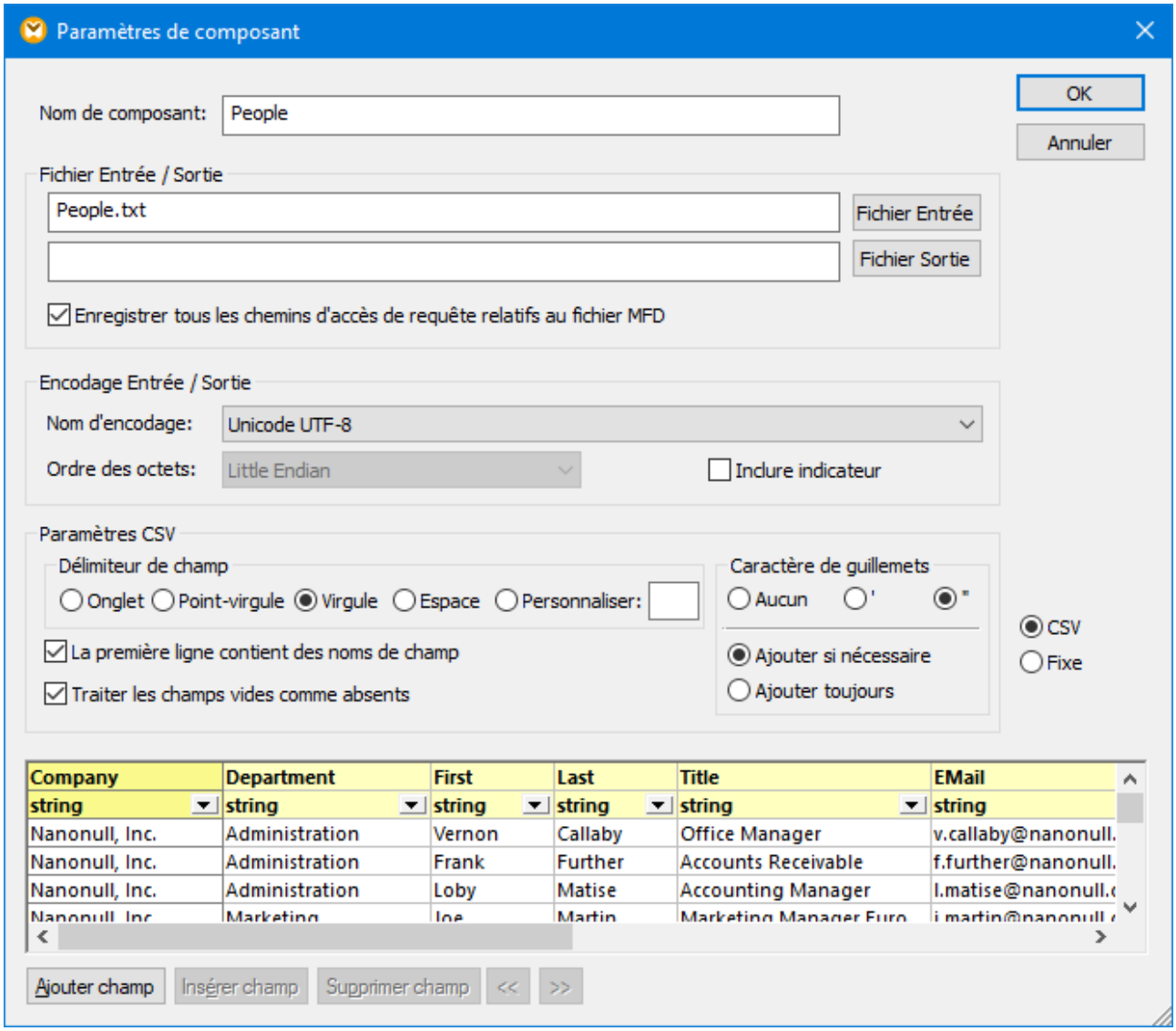



*Head-detail-inline.mfd*

## 4.3.4 Configurer les options CSV

Une fois avoir ajouté un composant de texte dans la surface de mappage, vous pouvez configurer les paramètres applicables depuis le dialogue Paramètres de composant. Vous pouvez ouvrir ce dialogue d'une des trois manières suivantes :

- Choisir 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**.

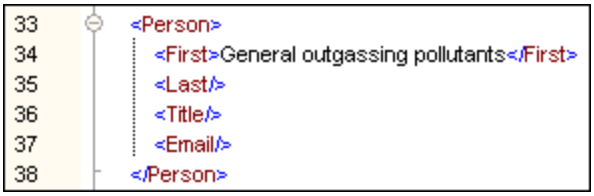
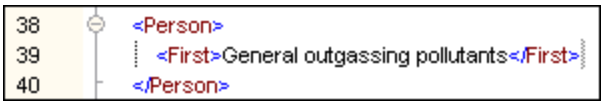
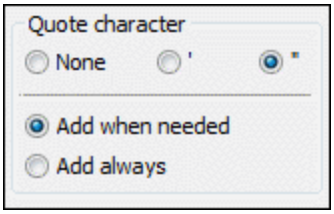



*Dialogue Paramètre de composant de texte (en mode CSV)*

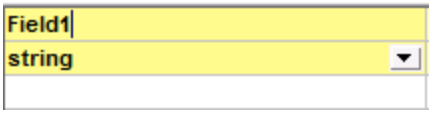
Les paramètres disponibles sont les suivants :

| | |
|---|---|
| *Nom du 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 :<br><br>• Si vous souhaitez déployer le mappage vers FlowForce Server, le nom du composant doit être unique.<br>• 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 d'entrée* | Spécifie le fichier à partir duquel MapForce lira les données. Ce champ est pertinent pour un composant de source et est rempli lorsque vous créez tout d'abord le composant puis que vous l'attribuez à un fichier de texte. Le champ peut rester vide si vous utilisez le composant de fichier de texte en tant que cible pour votre mappage.<br><br>Dans un composant de source, MapForce utilise la valeur de ce champ pour lire des noms de colonne et pour consulter les contenus du fichier de texte d'instance.<br><br>Pour choisir un nouveau fichier, cliquer sur **Fichier d'entrée**. |
| *Fichier de sortie* | Spécifie le fichier vers lequel MapForce écrira des données. Ce champ est pertinent pour un composant de cible<br><br>Pour choisir un nouveau fichier, cliquer sur **Fichier de sortie**. |
| *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 des Paramètres de composant relatifs à l'emplacement du fichier MapForce Design (.mfd). Ce paramètre touche les fichiers d'entrée et de sortie utilisés par le composant du texte. Voir aussi Utiliser des chemins relatifs sur un composant 47. |
| *Encodage Entrée / Sortie* | Vous permet de spécifier les paramètres suivants du fichier d'instance de sortie :<br><br>• Nom d'encodage<br>• Ordre d'octet<br>• Si le caractère "byte order mark" (BOM) doit être inclus.<br><br>Par défaut, tout nouveau composant doit avoir un encodage défini dans l'option **Encodage par défaut pour les nouveaux composants**. Vous pouvez accéder à cette option depuis **Outils \| Options**, onglet Généralité. |
| *Délimiteur de champ* | Les fichiers CSV sont délimités par des virgules "," par défaut. Cette option vous permet de sélectionner les caractères **Tabulateur**, **Point-** |

| | |
|---|---|
| | **virgule**, ou **Espace** en tant que délimiteurs. Vous pouvez aussi saisir un délimiteur personnalisé dans le champ **Personnaliser**. |
| *Première ligne contient des noms de champ* | Choisir cette option pour instruire MapForce à traiter les valeurs dans le premier enregistrement du fichier de texte en tant qu'en-têtes de colonne. Les en-têtes de colonne apparaissent ensuite en tant que noms d'item dans le mappage. |
| *Traiter les champs vides en tant qu'absent* | Lorsque cette option est activée, les champs vides dans le fichier de source ne produiront pas un item vide correspondant (élément ou attribut) dans le fichier cible.<br><br>Par exemple, l'enregistrement CSV "`General outgassing pollutants,,,`" consiste en quatre champs, dont les trois derniers sont vides.<br><br>Si on part du principe que la sortie est un fichier XML, lorsque cette option est désactivée, les champs vides seront créés dans la sortie avec une valeur vide (dans cet exemple, les éléments `Last`, `Title` et `Email`):<br><br>  <br><br><br>Lorsque cette option est activée, les champs vides ne seront pas créés dans la sortie :<br><br>  <br> |
| *Caractère de guillemets* | Si votre sortie contient des guillemets autour des valeurs de champ, choisir le caractère de guillemet qui existe dans le fichier de source. Le même paramètre sera aussi utilisé pour les fichiers de sortie.<br><br>  <br><br><br>En ce qui concerne les fichiers de sortie, vous pouvez spécifier des paramètres supplémentaires : |

| | |
|---|---|
| | **Ajouter si nécessaire** — Ajoute le caractère de guillemet sélectionné uniquement aux champs dans lesquels le texte contient le délimiteur de champ, ou des sauts ligne.<br>**Ajouter toujours** — Ajoute le caractère de guillemet sélectionné à les champs du fichier CSV généré. |
| *CSV / Fixe* | Change le type de composant soit sur CSV ou FLF (champ de longueur fixe). |
| *Zone d'aperçu* | La partie inférieure du dialogue affiche un aperçu de jusqu'à 20 lignes du fichier sélectionné en tant qu'entrée ou sortie.<br><br>Si nécessaire, vous pouvez créer la structure du fichier (ou changer la structure du fichier existant), comme suit.<br><br>**Ajouter champ** — Crée un nouveau champ après le dernier enregistrement CSV.<br>**Insérer champ** — Crée un nouveau champ juste avant enregistrement CSV sélectionné actuellement.<br>**Supprimer champ** — Supprime le champ sélectionné actuellement.<br>**<<** — Déplace le champ sélectionné actuellement d'u position vers la gauche.<br>**>>** — Déplace le champ sélectionné actuellement d'u position vers la droite.<br><br>Pour changer le nom d'un champ, cliquer sur l'en-tête (par exemple, **Field1**), et saisir la nouvelle valeur. Veuillez noter que les noms de champ ne sont pas éditables lorsque l'option **Première ligne contient des noms de champ** est activée.<br><br>  <br><br>Pour changer le type de données d'un champ, choisir la valeur requise depuis la liste déroulante. MapForce contrôle le type de données, ainsi, si les données d'entrée et le format de champ ne concordent pas, les données sont marquées en rouge. |

| | |
|---|---|
| |  <br>Les types de champ sont basés sur les types de données de schéma XML par défaut. Par exemple, le type Date se présente sous la forme **YYYY-MM-DD**. |

## 4.3.5 FLF à base de données

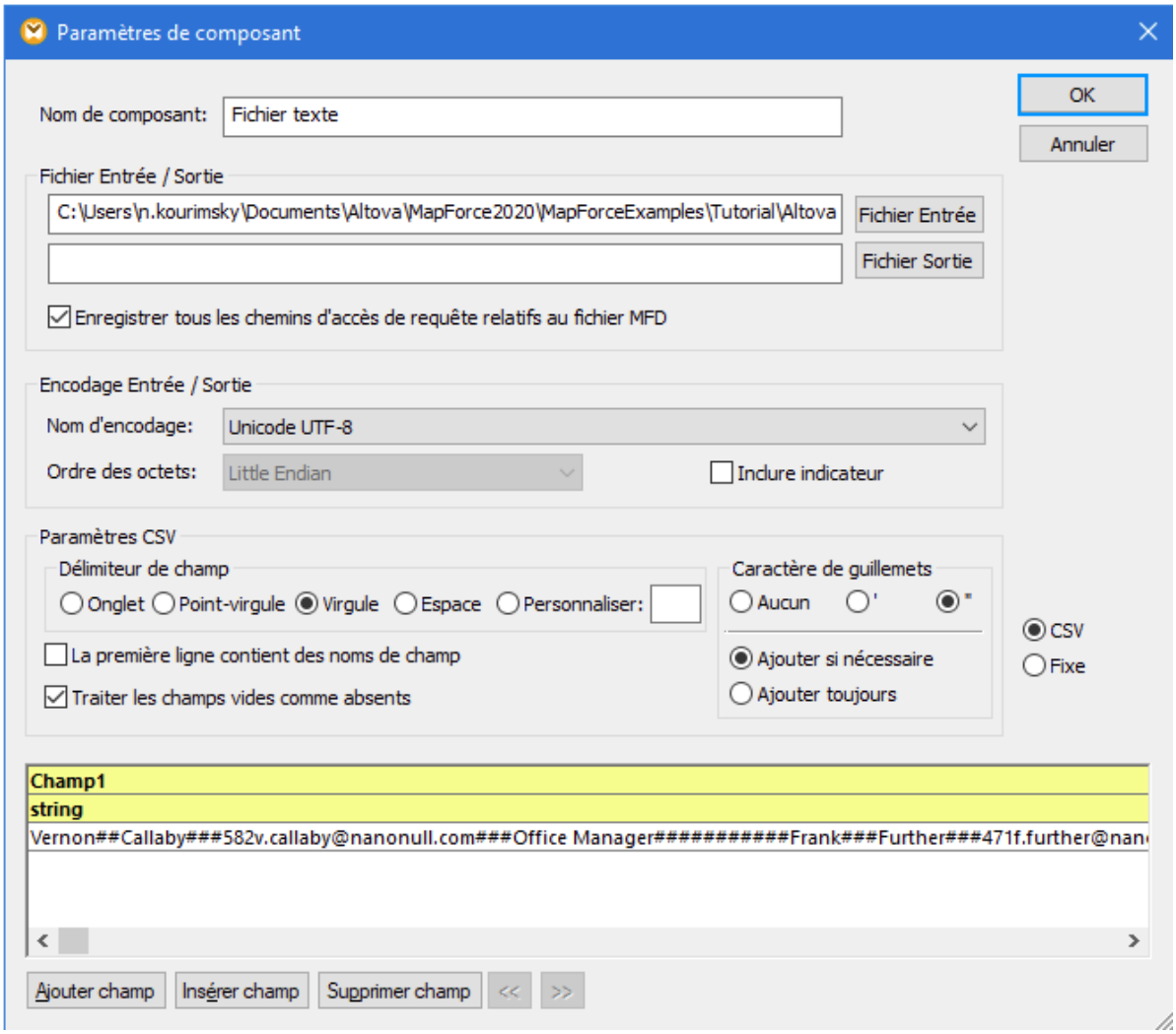
Cet exemple illustre une opération de mappage de données entre un fichier de texte de longueur fixe (FLF) et une base de données SQLite. Les fichiers utilisés dans l'exemple sont disponibles dans le dossier **`Tutorial`**. Le fichier de texte de source et la base de données cible stockent une liste d'employés. Dans le fichier de source, les enregistrements sont délimités implicitement par leur taille, comme suit :

| Position et nom du champ | Taille (en caractères) |
|---|---|
| Champ 1 (Prénom) | 8 |
| Champ 2 (Nom de famille) | 10 |
| Champ 3 (Numéro de poste) | 3 |
| Champ 4 (E-mail) | 25 |
| Champ 5 (Position) | 25 |

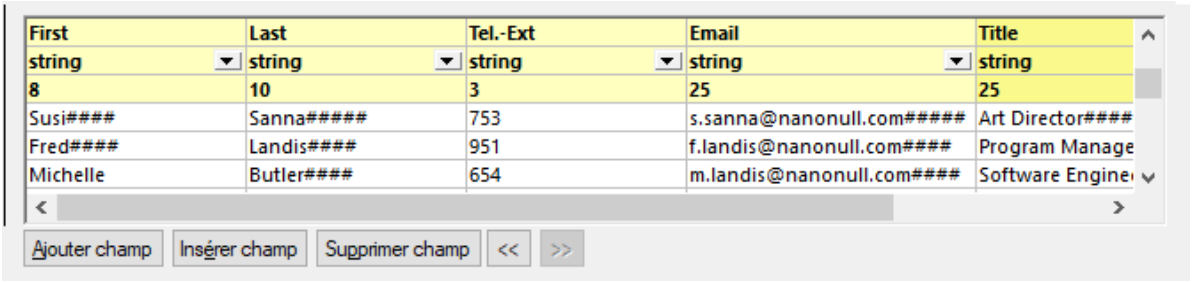
Le but du mappage est de mapper les données FLF au composant de base de données. Nous voulons également mapper les extensions de téléphone avec un nouveau préfixe. Pour ce faire, suivre les étapes suivantes :

### Étape 1: Insérer et configurer le composant de texte

1. Choisir l'option de menu **Insérer | Fichier texte**, ou cliquer sur l'icône Insérer fichier texte ((**Insérer fichier texte**)  .
2. Cliquez sur **fichier d'entrée** dans la boîte de dialogue **Paramètres de composant** (*voir ci-dessous*) et sélectionnez **`Altova-FLF.txt`**.



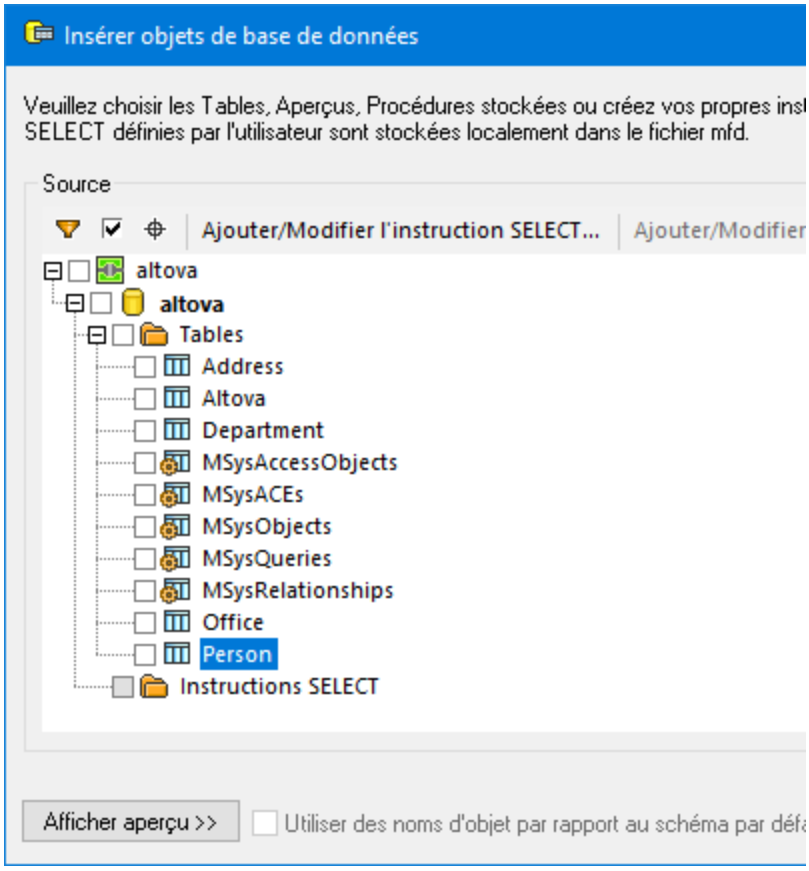

3. Choisir **Fixe**.
4. Décocher la case **Assumer la présence des délimiteurs d'enregistrement**.
5. Les trois lignes marquées en jaune sont éditables, et vous permettent de spécifier i) le nom du champ ii) le type de données et iii) la taille du champ. Saisir **8** en guise de la nouvelle taille du champ et appuyer sur **Entrée**. Vous verrez apparaître plus de données dans la première colonne qui est maintenant définie avec une largeur de 8 caractères.
6. Cliquer sur **Ajouter champ** pour ajouter un nouveau champ, et définir la longueur du deuxième champ à 10 caractères.
7. Utiliser la même méthode pour créer trois autres champs avec les longueurs suivantes : 3, 25, et 25 caractères. Puis changer les en-tête de champ,tel qu'affiché ci-dessous.




8. Dans le groupe **Paramètres de champ à longueur fixe**, choisissez **Personnaliser** et saisissez le caractère dièse (#). Cela indique à MapForce de traiter le caractère # en tant que caractère de remplissage.
9. Cliquer sur **OK**.
10. MapForce vous demandera si vous souhaitez changer le nom du composant afin qu'il corresponde aux fichiers d'instance. Cliquer sur **Changer le nom du composant**. Le composant `Altova-FLF` apparaît dans la fenêtre de mappage.

## Étape 2: Insérer le composant de base de données

1. Choisir la commande de menu **Insérer | Base de données**, choisir **SQLite**, puis cliquer sur **Suivant**.
2. Choisir la base de données **`Altova.sqlite`** et cliquer sur **Se connecter**.
3. Choisir la table **Person** (*voir ci-dessous*) et cliquer sur **OK**.

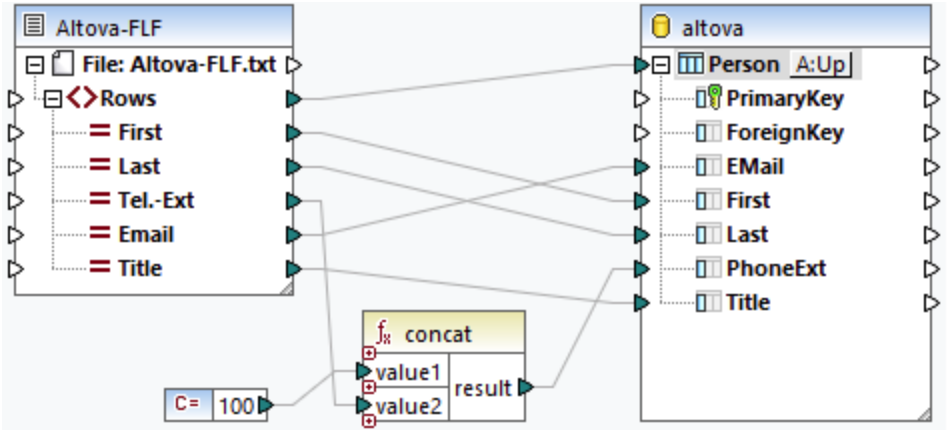


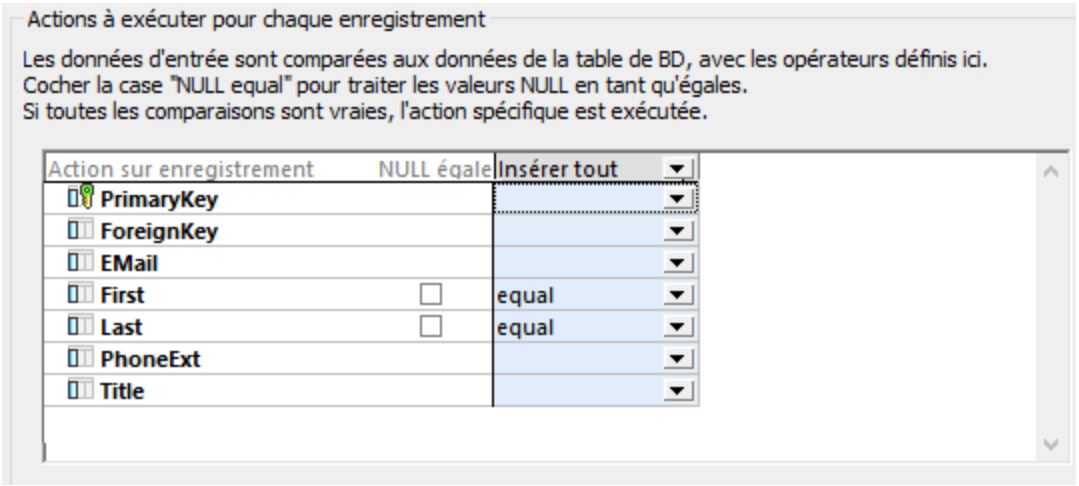
## Étape 3: Concevoir le mappage

Dans la prochaine étape, nous allons créer un mappage :

1. Glisser la fonction **core | concat** 629 depuis la fenêtre Bibliothèques dans le mappage.
2. Choisir la commande de menu **Insérer | Constante**, choisir *Nombre* en tant que type et saisir une valeur de 100 dans le champ de texte. Cette constante stocke le nouveau préfixe de poste téléphonique.
3. Créer le mappage comme indiqué ci-dessous.




4. Dans le composant de base de données, cliquer sur la touche **Action de table**. Ceci ouvre la boîte de dialogue **Actions de table de la base de données** (*voir la capture d'écran ci-dessous*).
5. À côté de *Action sur données d'entrée*, choisir **Mettre à jour si.** Définissez les champs **First** et **Last** sur l'action `equal`. Cliquez sur **OK**. MapForce sera invité à mettre à jour la table `Person` uniquement si les prénom et nom de famille dans le fichier source sont égaux champs de base de données correspondants. Si cette condition est vraie, numéro de poste est préfixé avec 100, et sera copié dans le champ `PhoneExt` de la table `Person`.

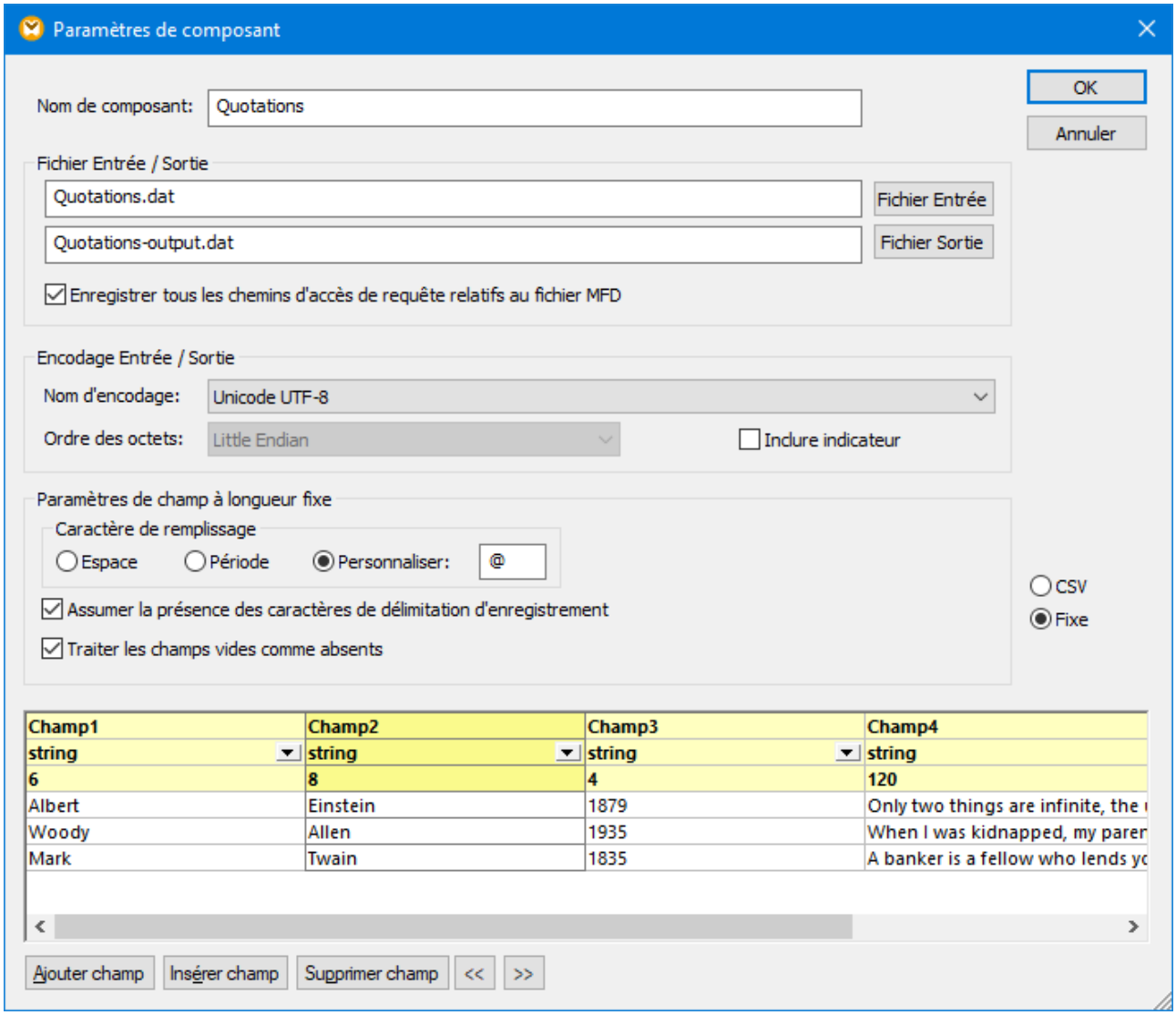



6. Pour générer l'instruction SQL (pour un aperçu dans MapForce), cliquer sur l'onglet **Sortie**. Pour exécuter les instructions SQL par rapport à la base de données, cliquer sur la touche **Exécuter script SQL**  .

## 4.3.6 Configurer les options FLF

Une fois avoir ajouté un composant de texte dans la surface de mappage, vous pouvez configurer les paramètres applicables depuis le dialogue Paramètres de composant. Vous pouvez ouvrir ce dialogue d'une des trois manières suivantes :

- Choisir 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**.

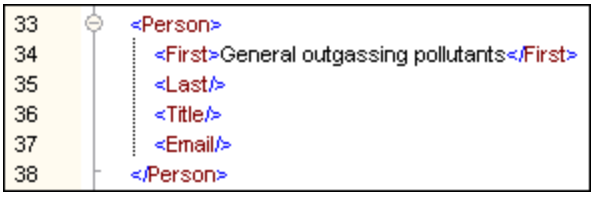



*Dialogue Paramètre de composant de texte (en mode champ de longueur fixe)*

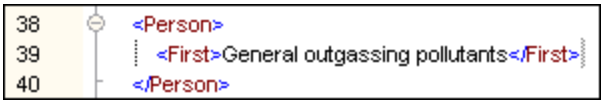
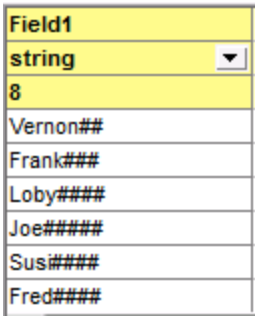
Les paramètres disponibles sont les suivants :

| | |
|---|---|
| *Nom du 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 :<br><br>• Si vous souhaitez déployer le mappage vers FlowForce Server, le nom du composant doit être unique.<br>• 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 d'entrée* | Spécifie le fichier à partir duquel MapForce lira les données. Ce champ est pertinent pour un composant de source et est rempli lorsque vous créez tout d'abord le composant puis que vous l'attribuez à un fichier de texte. Le champ peut rester vide si vous utilisez le composant de fichier de texte en tant que cible pour votre mappage.<br><br>Dans un composant de source, MapForce utilise la valeur de ce champ pour lire des noms de colonne et pour consulter les contenus du fichier de texte d'instance.<br><br>Pour choisir un nouveau fichier, cliquer sur **Fichier d'entrée**. |
| *Fichier de sortie* | Spécifie le fichier vers lequel MapForce écrira des données. Ce champ est pertinent pour un composant de cible.<br><br>Pour choisir un nouveau fichier, cliquer sur **Fichier de sortie**. |
| *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 des Paramètres de composant relatifs à l'emplacement du fichier MapForce Design (.mfd). Ce paramètre touche les fichiers d'entrée et de sortie utilisés par le composant du texte. Voir aussi Utiliser des chemins relatifs sur un composant 47. |
| *Encodage Entrée / Sortie* | Vous permet de spécifier les paramètres suivants du fichier d'instance de sortie :<br><br>• Nom d'encodage<br>• Ordre d'octet<br>• Si le caractère "byte order mark" (BOM) doit être inclus. |

| | |
|---|---|
| | Par défaut, tout nouveau composant doit avoir un encodage défini dans l'option **Encodage par défaut pour les nouveaux composants**. Vous pouvez accéder à cette option depuis **Outils \| Options**, onglet Généralité. |
| *Caractère de remplissage* | Cette option vous permet de définir les caractères qui doivent être utilisées pour terminer ou remplir le reste du champ (fixe) lorsque les données entrantes sont plus courtes que les définitions de champ respectives. Le champ personnalisé vous permet de définir votre propre caractère de remplissage dans le champ Custom.<br><br>Si les données entrantes contiennent déjà des caractères de remplissage spécifiques et que vous saisissez le même caractère de remplissage dans le champ Custom, alors les données entrantes seront débarrassés de ces caractères de remplissage ! |
| *Assumer la présence des délimiteurs d'enregistrement* | Cette option est utile lorsque vous souhaitez lire des données provenant d'un fichier plat de source qui ne contient pas de délimiteurs d'enregistrement comme CR/LF, ou lorsque vous souhaitez produire un fichier FLF plat cible sans délimiteur d'enregistrement.<br><br>Voir la section Comprendre l'option "Assumer la présence des délimiteurs d'enregistrement" 394 ci-dessous. |
| *Traiter les champs vides comme absents* | Lorsque cette option est activée, les champs vides dans le fichier de source ne produiront pas un item vide correspondant (élément ou attribut) dans le fichier cible.<br><br>En partant du principe que la sortie est un fichier XML, lorsque cette option est désactivée, les champs vides seront créés dans la sortie avec une valeur vide (dans cet exemple, les éléments `Last`, `Title`, et `Email`):<br><br>  <br><br><br>Lorsque cette option est activée, les champs vides ne seront pas créés dans la sortie : |

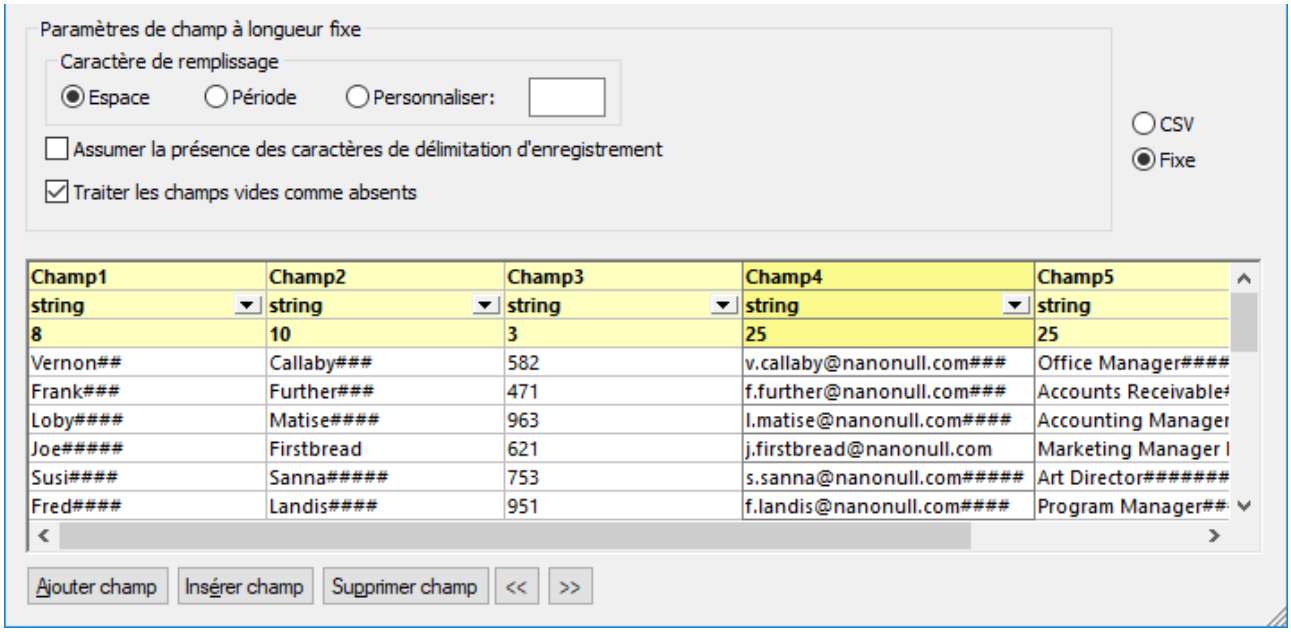
 

| | |
|---|---|
| |  |
| *CSV / Fixe* | Change le type de composant soit sur CSV ou FLF (champs de longueur fixe). |
| *Zone d'aperçu* | La partie inférieure du dialogue affiche un aperçu de jusqu'à 20 lignes choisies en tant qu'entrée ou sortie.<br><br>Si nécessaire, vous pouvez créer la structure du fichier (ou changer la structure de celle existante), comme suit :<br><br>**Ajouter champ** — Crée un nouveau champ après le dernier enregistrement .<br><br>**Insérer champ** — Crée un nouveau champ juste avant l'enregistrement sélectionné actuellement.<br><br>**Supprimer champ** — Supprime le champ sélectionné actuellement.<br><br>**<<** — Déplace le champ sélectionné actuellement d'u position vers la gauche.<br><br>**>>** — Déplace le champ sélectionné actuellement d'u position vers la droite.<br><br>Pour changer le nom d'un champ, cliquer sur l'en-tête (par exemple, **Champ1**), et saisir la nouvelle valeur.<br><br>  <br><br>Pour changer le type de données d'un champ, choisir la valeur requise depuis la liste déroulante. MapForce contrôle le type de données, |

| | |
|---|---|
| | ainsi, si les données d'entrée et le format de champ ne concordent pas, les données sont marquées en rouge.<br><br>Field1<br>decimal<br>8<br>Vernon##<br>Frank###<br>Loby####<br>Joe#####<br>Susi####<br>Fred####<br><br>Pour définir la taille du champ dans les caractères, saisir la taille de champ dans la troisième ligne depuis le haut. |

## Comprendre l'option "Assumer la présence des délimiteurs d'enregistrement"

Afin de mieux comprendre cette option, ouvrir le fichier **Altova-FLF.txt** disponible dans le dossier **<Documents>\Altova\MapForce2024\MapForceExamples\Tutorial**. Noter que le fichier consiste en des enregistrements longs de 71 caractères, sans délimiteurs comme CR/LF. Si vous souhaitez lire des données depuis ce fichier spécifiquement, vous devez partager ce fichier dans des enregistrements. C'est à dire, créer plusieurs champs dont la taille totalise jusqu'à 71 caractères (comme indiqué ci-dessous), et puis désactiver **Assumer la présence des délimiteurs d'enregistrement**. Pour un exemple étape par étape, voir Exemple : Mapper des fichiers de texte de longueur fixe dans des bases de données 386.



Si vous souhaitez écrire des données depuis ce fichier dans un fichier de destination qui utilise la même structure, puis activer **Assumer la présence des délimiteurs d'enregistrement** crée un nouvel enregistrement tous les 71 caractères.

```
1  Vernon##Callaby###582v.callaby@nanonull.com###Office Manager##########
2  Frank###Further###471f.further@nanonull.com###Accounts Receivable######
3  Loby####Matise####963l.matise@nanonull.com####Accounting Manager#######
4  Joe#####Firstbread621j.firstbread@nanonull.comMarketing Manager Europe#
5  Susi####Sanna#####753s.sanna@nanonull.com#####Art Director#############
6  Fred####Landis####951f.landis@nanonull.com####Program Manager##########
7  MichelleButler####654m.landis@nanonull.com####Software Engineer########
8  Ted#####Little####852t.little@nanonull.com####Software Engineer########
9  Ann#####Way#######951a.way@nanonull.com#######Technical Writer#########
10 Liz#####Gardner###753l.gardner@nanonull.com###Software Engineer########
11 Paul####Smith#####334p.smith@nanonull.com#####Software Engineer########
12 Alex####Martin####778a.martin@nanonull.com####IT Manager###############
13 George##Hammer####223g.hammer@nanonull.com####Web Developer############
14 Jessica#Bander####241j.band@nanonull.com######Support Engineer#########
15 Lui#####King######345l.king@nanonull.com######Support Engineer#########
16 Steve###Meier#####114s.meier@nanonull.com#####Office Manager##########
17 Theo####Bone######331t.bone@nanonull.com######Accounts Receivable######
18 Max#####Nafta#####122m.nafta@nanonull.com#####PR & Marketing Manager US
19 ValentinBass######716v.bass@nanonull.com######IT Manager###############
20 Carl####Franken###147c.franken@nanonull.com###Support Engineer#########
21 Mark####Redgreen##152m.redgreen@nanonull.com##Support Engineer#########
22
```

*Le résultat de mappage lorsque "Assumer la présence des délimiteurs d'enregistrement" est activé*

Si **Assumer la présence des délimiteurs d'enregistrement** est désactivé, le résultat de mappage apparaît sous la forme d'un long string.

```
1    Vernon##Callaby###582v.callaby@nanonull.com###Office
     Manager##########Frank###Further###471f.further@nanonull.com###Accounts
     Receivable######Loby####Matise####963l.matise@nanonull.com####Accounting
     Manager#######Joe#####Firstbread621j.firstbread@nanonull.comMarketing Manager
     Europe#Susi####Sanna#####753s.sanna@nanonull.com#####Art
     Director############Fred####Landis####951f.landis@nanonull.com####Program
     Manager#########MichelleButler####654m.landis@nanonull.com####Software
     Engineer########Ted#####Little####852t.little@nanonull.com####Software
     Engineer########Ann#####Way#######951a.way@nanonull.com#######Technical
     Writer#########Liz#####Gardner###753l.gardner@nanonull.com###Software
     Engineer########Paul####Smith#####334p.smith@nanonull.com#####Software
     Engineer########Alex####Martin####778a.martin@nanonull.com####IT
     Manager###############George##Hammer####223g.hammer@nanonull.com####Web
     Developer############Jessica#Bander####241j.band@nanonull.com######Support
     Engineer#########Lui#####King######345l.king@nanonull.com######Support
     Engineer#########Steve###Meier#####114s.meier@nanonull.com#####Office
     Manager##########Theo####Bone######331t.bone@nanonull.com######Accounts
     Receivable######Max#####Nafta#####122m.nafta@nanonull.com#####PR & Marketing Manager
     USValentinBass######716v.bass@nanonull.com######IT
     Manager###############Carl####Franken###147c.franken@nanonull.com###Support
     Engineer#########Mark####Redgreen##152m.redgreen@nanonull.com##Support
     Engineer#########
```

*Le résultat de mappage lorsque "Assumer la présence des délimiteurs d'enregistrement" est désactivé*

# 5 Composants de transformation

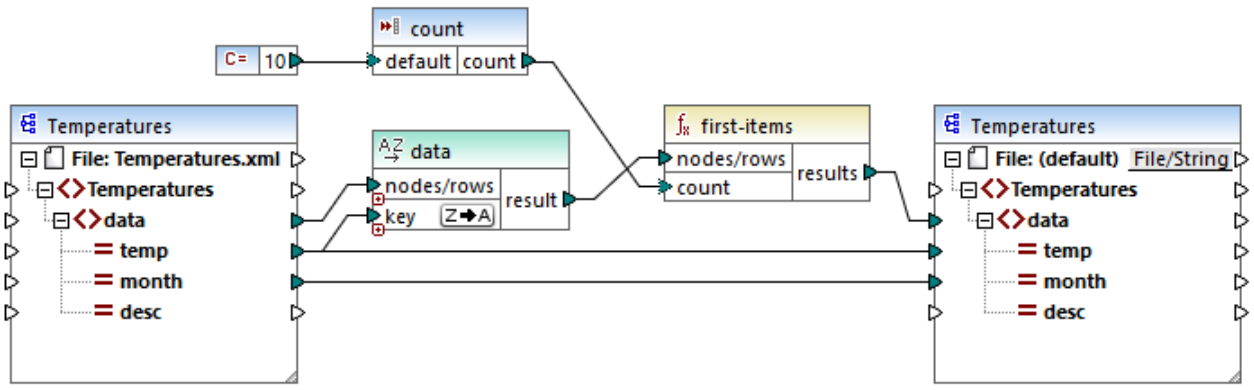
Cette section décrit les composants de transformation qui peuvent être utilisés pour transformer les données ou les stocker temporairement pour traitement ultérieur. La liste des composants de transformation est recensée ci-dessous :



Notez que les fonctions appartiennent également aux composants de transformation. Toutefois, les fonctions 467 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[601] 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[787].

Vous pouvez utiliser des composants d'entré simples dans un des langages de transformation MapForce suivants :

- BUILT-IN (lorsque vous consultez la transformation de mappage directement dans MapForce, depuis l'onglet **Aperçu**)
- BUILT-IN (lorsque vous exécutez un fichier d'exécution de MapForce Server compilé)
- XSLT 1.0, XSLT 2.0, XSLT 3.0
- XQuery
- C++
- C#
- Java

Dans le cas des mappages exécutés avec MapForce Server ou par le biais du code généré, des composants d'entrée simple deviennent des paramètres de ligne de commande. Dans les cas de mappages générés comme transformations XSLT, les composants d'entrée simple correspondent aux paramètres de feuille de style dans le fichier XSLT généré.

Vous pouvez créer chaque composant d'entrée simple (ou paramètre) comme optionnel ou obligatoire (voir Ajouter des composants d'entrée simple[399]). 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[402]. 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 403.

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 491. 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. |
| Applicable à tout le mappage. | Applicable uniquement dans le contexte de la fonction dans laquelle ils ont été définis. |

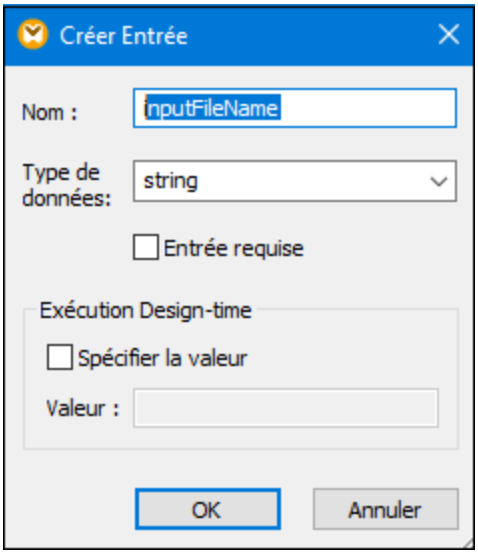
Lorsque vous avez créez un mappage inversé (utiliser la commande de menu **Outils | Créer le mappage inversé**), un composant d'entrée simple devient un composant de sortie simple.

## 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 400.

**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**.

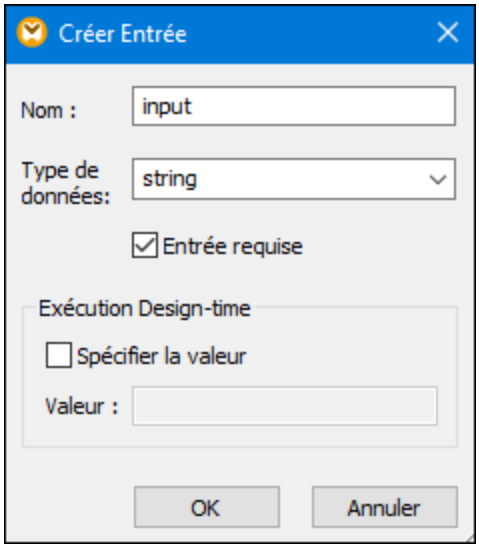
Vous pouvez changer ultérieurement tous les paramètres définis ici (voir Paramètres de composant d'entrée simple 400).

## 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**.

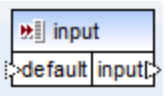
Les paramètres disponibles sont les suivants.

| | |
|---|---|
| *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).<br><br>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 402). |
| *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.<br><br>**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

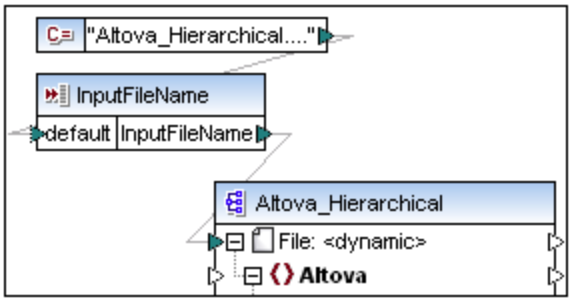
Une fois avoir ajouté un composant d'entrée dans la zone de mappage, veuillez noter l'item de **défaut** situé à gauche du composant.



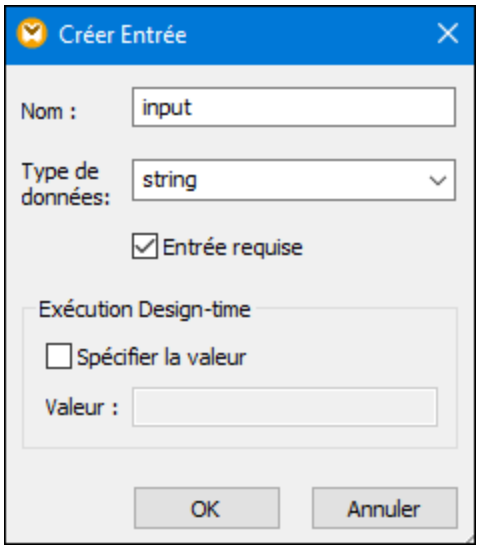

*Composant d'entrée simple*

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**.

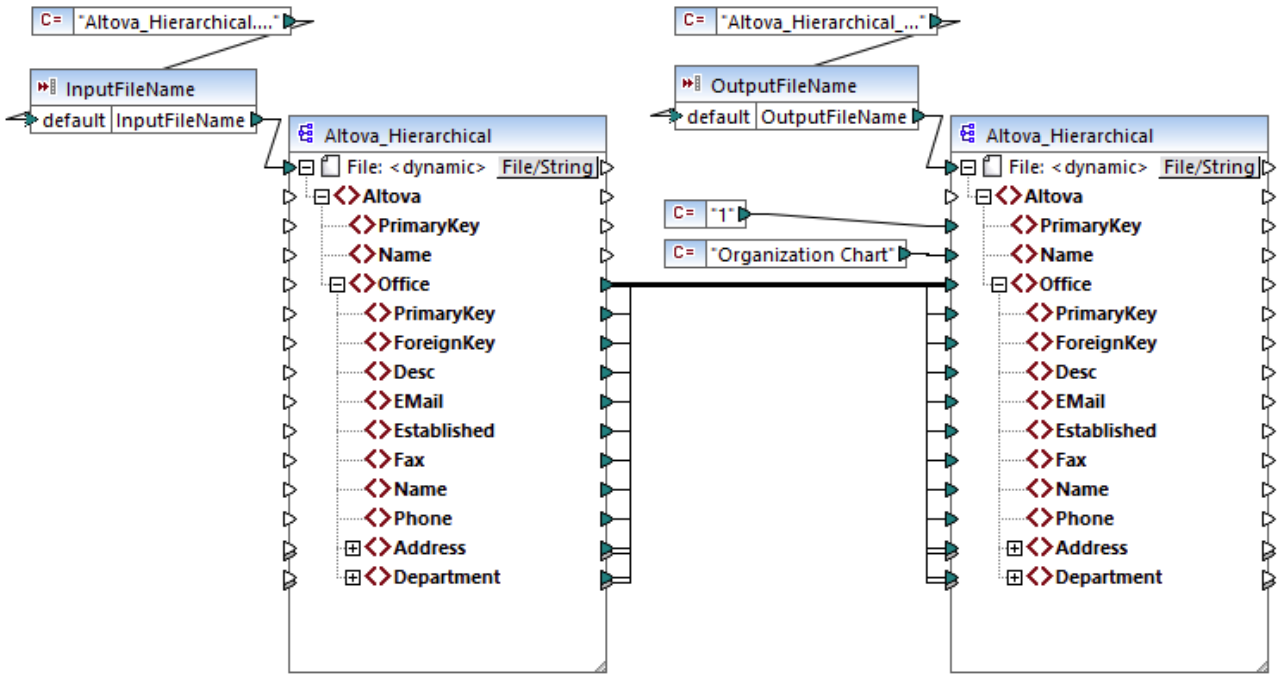
**Note :** Si vous cochez la case **Spécifier valeur** et saisissez une valeur dans la case adjacente, la valeur saisie prend précédence sur la valeur par défaut lorsque vous consultez le mappage (c'est à dire au moment de l'exécution au moment du design). Néanmoins, la valeur au moment du design n'a 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/String** ( File/String ) , 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 400. 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 2.0 405, utiliser RaptorXML Server
- Built-in (fichier d'exécution MapForce Server) 405, avec MapForce Server
- Java 406
- C# 407
- C++ 407

## XSLT 2.0

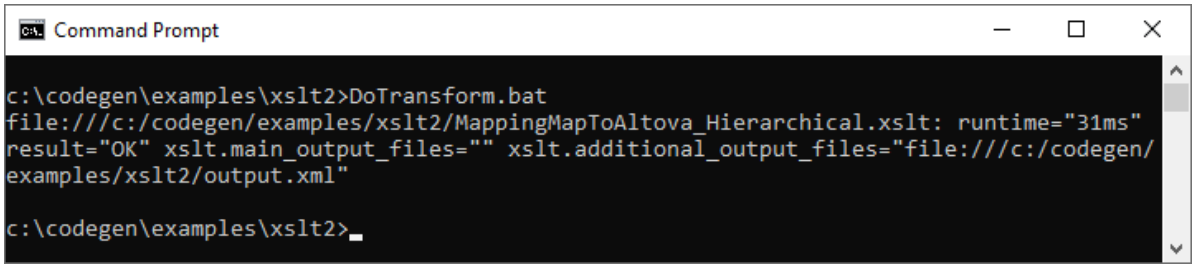
Si vous générez du code dans XSLT 1.0, XSLT 2.0 ou XSLT 3.0, un fichier batch **DoTransform.bat** est généré dans le répertoire de cible choisi, en plus du fichier XSLT. Le fichier **DoTransform.bat** vous permet d'exécuter le mappage avec RaptorXML Server, voir Automatisation avec RaptorXML Server 856.

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**.




## Fichier d'exécution MapForce Server

Pour fournir des paramètres d'entrée personnalisés vers un fichier d'exécution MapForce Server :

1. Si vous ne l'avez pas encore fait, ouvrir l'exemple **FileNamesAsParameters.mfd** depuis le répertoire **<Documents>\Altova\MapForce2024\MapForceExamples**.

2. Dans le menu **Fichier**, cliquer sur **Compiler sur le fichier d'exécution MapForce**, voir aussi Compiler des Mappages dans les fichiers d'exécution MapForce Server[864]. Sur invitation, enregistrer le fichier d'exécution .mfx dans un répertoire sur votre ordinateur (dans cet exemple, **c:\codegen\examples\mfx** ).
3. Copier le fichier **Altova_Hierarchical.xml** depuis le répertoire **<Documents>\Altova\MapForce2024\MapForceExamples** vers le répertoire dans lequel vous avez enregistré le fichier .mfx.
4. Exécuter MapForce Server avec la commande suivante :

```
MapForceServer.exe run "C:\codegen\examples\mfx\FileNamesAsParameters.mfx" -
p=InputFileName:"C:\codegen\examples\mfx\Altova_Hierarchical.xml" -
p=OutputFileName:"C:\codegen\examples\mfx\OutputFile.xml"
```

Dans la commande MapForce Server ci-dessus, `-p=InputFileName` et `-p=OutputFileName` sont les paramètres d'entrée dans le mappage. Vous pouvez utiliser tout nom de fichier en tant que la valeur de **-OutputFileName**. Néanmoins le nom de fichier fourni dans le paramètre dans **-InputFileName** doit exister en tant que fichier physique ; sinon le mappage échouera.

**Note :** Si vous voyez le message "MapForceServer.exe n'est pas reconnu en tant que commande interne ou externe, programme opérable, ou le fichier batch", changer le répertoire actuel sur celui dans lequel l'exécutable de MapForce Server est installé. Pour éviter de changer de chemin à chaque fois que vous exécutez un mappage, ajouter à la variable d'environnement PATH du système d'exploitation le chemin du répertoire dans lequel l'exécutable MapForce Server est installé (par exemple, **C:\Program Files (x86)\Altova\MapForceServer2024\bin**).

Avec MapForce Server, l'exécution d'un mappage est aussi possible en appelant l'API de MapForce Server (qui est invocable depuis des langages comme C++, C#, ou Java). Pour plus d'informations concernant ce scénario, se référer à la documentation MapForce Server (https://www.altova.com/fr/documentation).

## Java

Pour fournir un paramètre d'entrée personnalisé dans une application Java .jar :

1. Si vous ne l'avez pas encore fait, ouvrir l'exemple **FileNamesAsParameters.mfd** depuis le répertoire **<Documents>\Altova\MapForce2024\MapForceExamples**.
2. Dans le menu **Fichier**, cliquer sur **Générer du Code dans | Java**.
3. Compiler code Java dans un fichier JAR exécutable. Pour un exemple de la manière de procéder dans Eclipse, voir Exemple : Générer et exécuter du code Java.
4. Copier le fichier **Altova_Hierarchical.xml** depuis **<Documents>\Altova\MapForce2024\MapForceExamples** vers le répertoire dans lequel le fichier .jar se trouve. Comme indiqué précédemment, le mappage tentera de lire ce fichier si vous ne fournissez pas de valeur personnalisée dans le paramètre **InputFileName**.
5. Exécuter l'application Java avec la commande suivante :

```
java -jar Mapping.jar /OutputFileName "output.xml"
```

Dans la commande ci-dessus, le paramètre d'entrée `/OutputFileName` fournit le nom du fichier de sortie à générer.

**Note :** Si vous utilisez des caractères génériques lors de la transmission de paramètres vers des fichiers .jar, il faut contenir les paramètres de caractères génériques entre guillemets, par exemple :

```
java -jar Mapping.jar /InputFileName "altova-*.xml"
```

### C#

Pour fournir un paramètre d'entrée personnalisé dans une application de ligne de commande C# générée par MapForce :

1. Si vous ne l'avez pas encore fait, ouvrir l'exemple **FileNamesAsParameters.mfd** depuis le répertoire **<Documents>\Altova\MapForce2024\MapForceExamples**.
2. Dans le menu **Fichier**, cliquer sur **Générer du Code dans | C#**, et sélectionner un répertoire cible (**C:\codegen\examples\cs**, dans cet exemple).
3. Ouvrir la solution dans Visual Studio et la construire (**Ctrl + Shift + B**).
4. Copier le fichier **Altova_Hierarchical.xml** depuis **<Documents>\Altova\MapForce2024\MapForceExamples** dans le répertoire où **Mapping.exe** a été généré (dans cet exemple, **C:\codegen\examples\cs\Mapping\bin\Debug**). Comme indiqué précédemment, le mappage tentera de lire ce fichier si vous ne fournissez pas de valeur personnalisée dans le paramètre **InputFileName**.
5. Ouvrir une fenêtre d'Invite de commande et passer au répertoire dans lequel se trouve **Mapping.exe**.

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

6. Exécuter l'application avec la commande suivante :

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

Dans la commande ci-dessus, le paramètre d'entrée `/OutputFileName` fournit le nom du fichier de sortie à générer.

### C++

Pour fournir un paramètre d'entrée personnalisé dans une application de ligne de commande C++ générée par MapForce :

1. Si vous ne l'avez pas encore fait, ouvrir l'exemple **FileNamesAsParameters.mfd** depuis le répertoire **<Documents>\Altova\MapForce2024\MapForceExamples**.
1. Dans le menu **Fichier**, cliquer sur **Générer du Code dans | C++**, et sélectionner un répertoire cible (**C:\codegen\examples\cpp**, dans cet exemple).
2. Ouvrir la solution dans Visual Studio et la construire (**Ctrl + Shift + B**).
3. Copier le fichier **Altova_Hierarchical.xml** depuis **<Documents>\Altova\MapForce2024\MapForceExamples** dans le répertoire où **Mapping.exe** a été généré (dans cet exemple, **C:\codegen\examples\cpp\Mapping\Debug**). Comme indiqué précédemment, le mappage tentera de lire ce fichier si vous ne fournissez pas de valeur personnalisée dans le paramètre **InputFileName**.
4. Ouvrir une fenêtre d'Invite de commande et passer au répertoire dans lequel se trouve **Mapping.exe**.

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

5. Exécuter l'application avec la commande suivante :

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

Dans la commande ci-dessus, le paramètre d'entrée `/OutputFileName` fournit le nom du fichier de sortie à générer.

## 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[35], 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[411]). L'objectif principal d'un composant de sortie simple est, néanmoins, de retourner un string lors de l'appel de MapForce Server API, sans écrire de fichiers.

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[491]). 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 et de base de données. 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 ? |
|---|---|
| BUILT-IN (lors de la consultation de la transformation de mappage) | Vous pouvez consulter les composants de Sortie de la même manière que vous consulteriez une sortie de mappage basée sur fichier, en cliquant sur l'onglet **Sortie** dans la fenêtre Mappage. |
| BUILT-IN (lors de l'exécution du fichier d'exécution MapForce Server) | Lorsque vous exécutez un fichier d'exécution compilé de MapForce Server (voir Compiler un mappage MapForce[864]), la sortie de mappage est renvoyée dans le stream de sortie standard (stdout) pour que vous puissiez le voir ou le rediriger vers un fichier. Par exemple, en partant du principe que le nom du fichier d'exécution MapForce est **MyMapping.mfx**, utilisez la syntaxe suivante pour rediriger la sortie de mappage vers le fichier **output.txt** et toute erreur vers le fichier **log.txt** : |

| Langage | Comment cela fonctionne ? |
|---|---|
| | `MapForceServer.exe run MyMapping.mfx >output.txt 2>log.txt` |
| 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.<br><br>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`.<br><br>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).<br><br>`RaptorXML xslt --xslt-version=2 --input="MappingMapToResult1.xslt" --output="Output.txt" %* "MappingMapToResult1.xslt"`<br><br>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. |
| C++, C#, Java | Dans le code C++, C# et Java généré, la sortie de mappage est écrite dans la sortie standard de l'application générée.<br><br>Si le mappage contient plusieurs composants cibles, l'application générée concatène la sortie standard de chaque composant cible et le retourne en tant qu'une sortie standard unifiée |

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

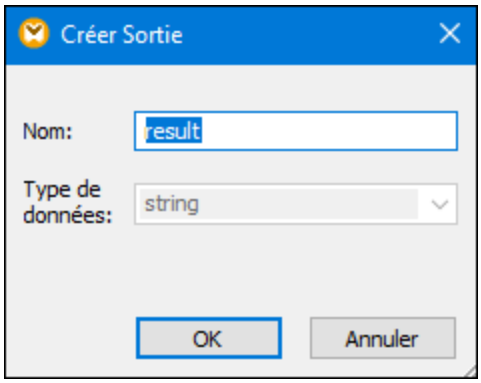
**Pour ajouter un composant de sortie vers la zone de mappage :**

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

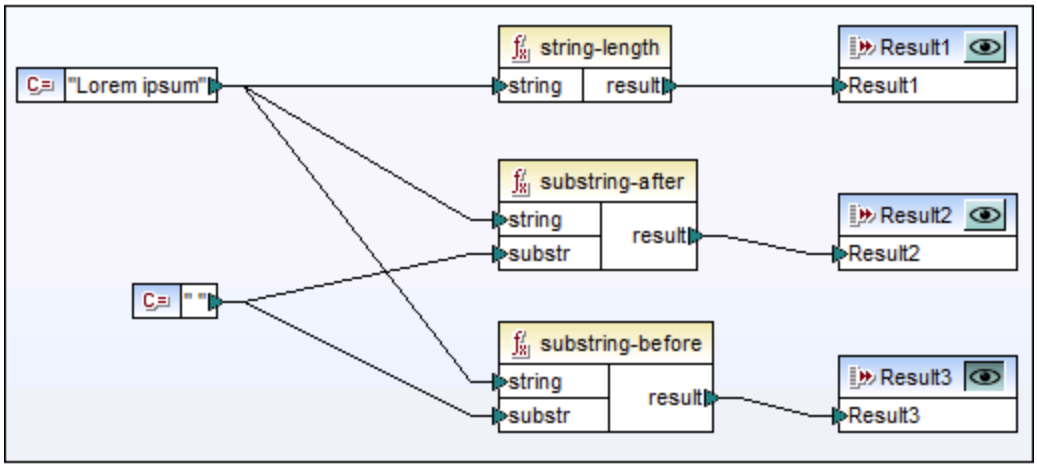
Cet exemple illustre comment consulter la sortie retournée par les fonctions MapForce à l'aide de composants de sortie simples. Vous profiterez au maximum de cet exemple si vous possédez des connaissances de base des fonctions en général et des fonctions MapForce en particulier. Si vous ne connaissez pas les fonctions de MapForce, vous pouvez consulter Utiliser les fonctions 467 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** 632 | 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** 633 | 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** 634 | 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.

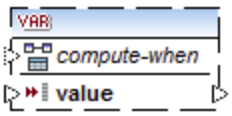
# 5.3 Les variables

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 802. 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 419.

## 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 :*

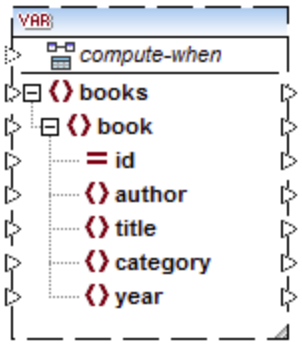
- Structure de schéma XML
- Structure de base de données

*Édition MapForce Enterprise :*

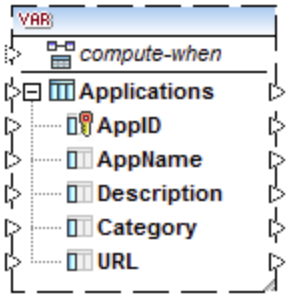
- Structure de schéma XML
- Structure de base de données
- Structure EDI
- Structure FlexText
- Structure de schéma JSON

*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.



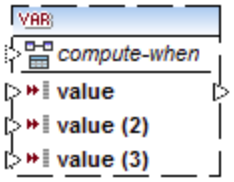

*Exemple 2 : Variable basée sur la base de données (éditions MapForce Professional et Enterprise)*
Si vous choisissez une structure de base de données pour votre variable(*voir la capture d'écran ci-dessous*), vous pouvez choisir une table de base de données spécifique en tant qu'item de racine pour 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 419.

## 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 45 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[105]. 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[419]. |
| 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

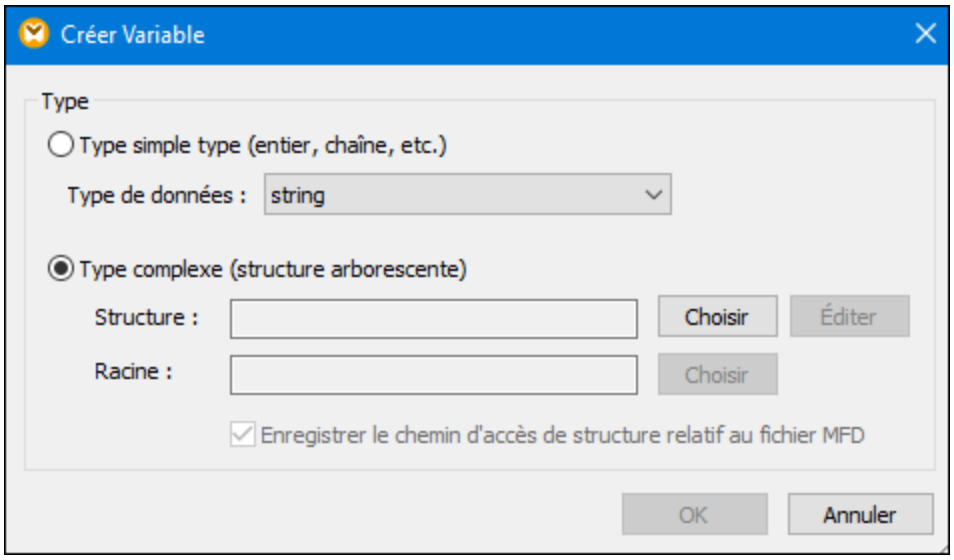
Cette rubrique explique comment ajouter une variable au mappage. La première option est d'ajouter une variable par la commande de menu ou de la barre d'outils. La deuxième option vous permet d'ajouter une variable par le menu contextuel.

### 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 

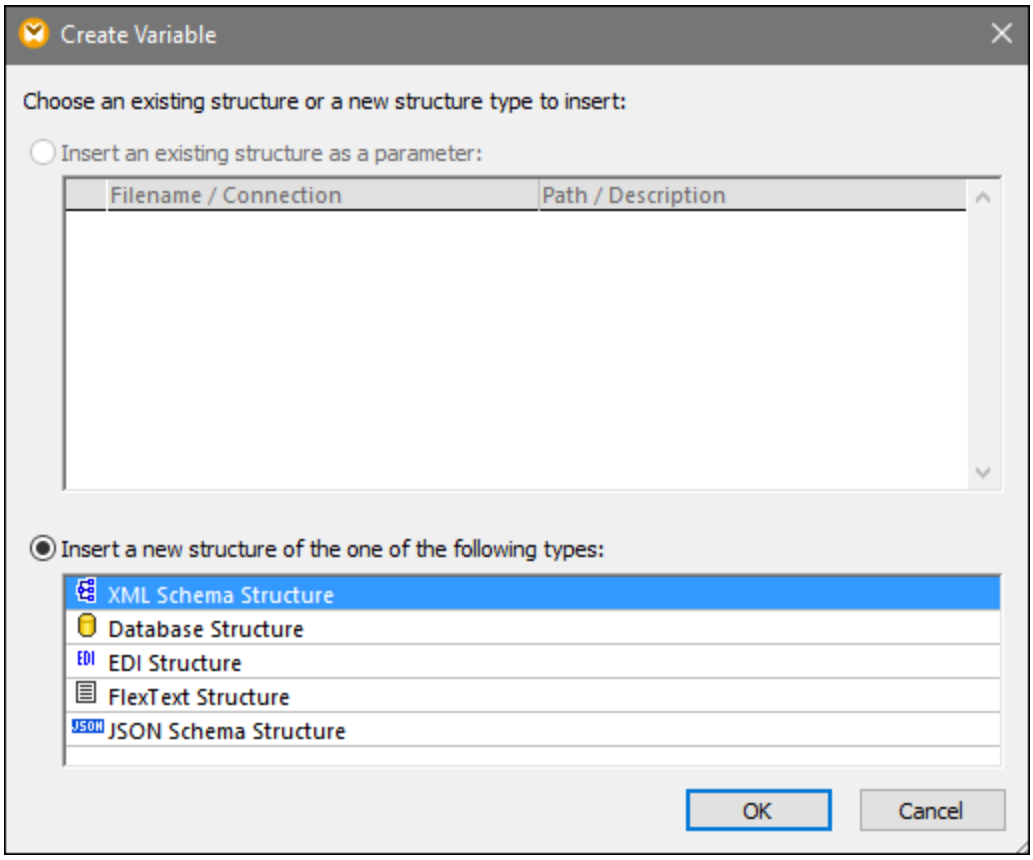
bouton de la barre d'outils (**Variable**).



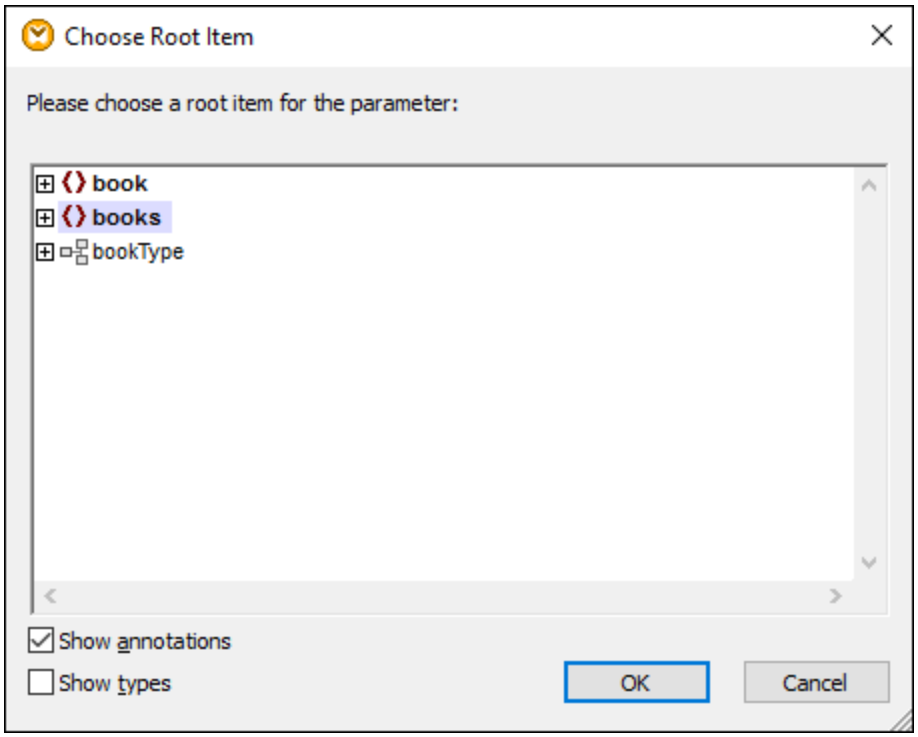

2. Sélectionnez le type de variable que vous souhaitez insérer (type simple ou complexe).

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[413]. 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[413] .




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*).

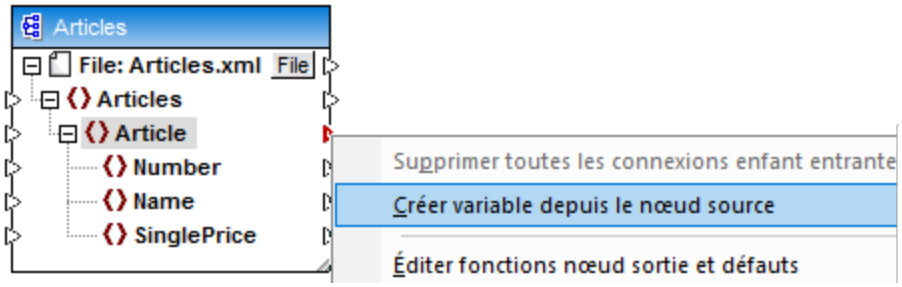



## Option 2 : par le menu contextuel

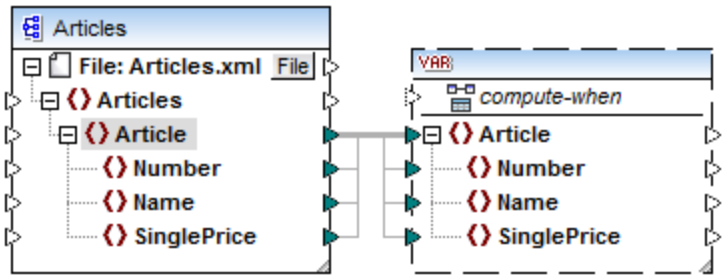
La deuxième option vous permet de créer une variable utilisant le menu contextuel. Les options possibles sont recensées ci-dessous.

*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 [60] (*voir la capture d'écran ci-dessous*).

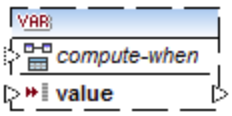



*Variable depuis un nœud cible*
Pour créer une variable depuis un nœud cible, cliquez de la touche droite sur le connecteur d'entrée d'un composant cible et sélectionnez **Créer une variable depuis un nœud cible**. Ceci crée une variable complexe avec le même schéma que pour la cible. Tous les items 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[502] 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.

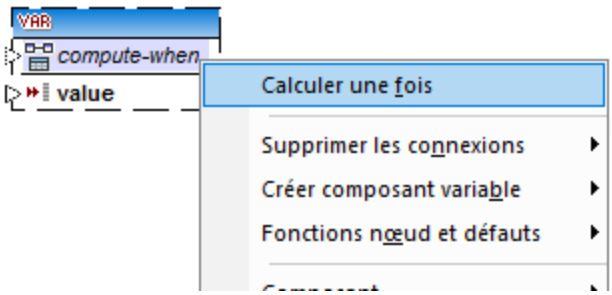
## « Compute-when » est connecté

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[802].

## « 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 :



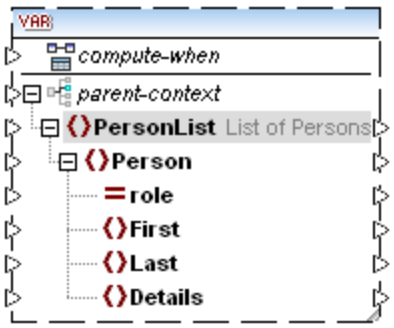

Lorsque vous changez l'étendue d'une variable vers `compute-when=once`, le connecteur d'entrée est supprimé de l'item `compute-when`, puisqu'une telle variable est uniquement évaluée une fois. Dans une fonction définie

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 : Changer le contexte Parent. 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.

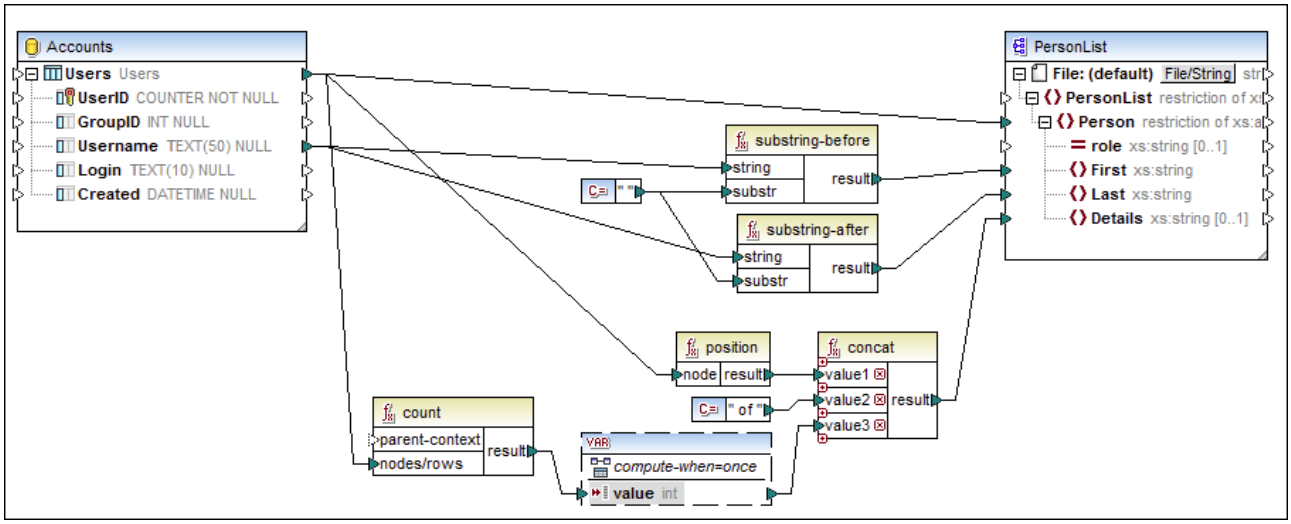



Le contexte parent ajoute un nœud parent virtuel à la hiérarchie dans le composant. Cela vous permet d'itérer sur un nœud supplémentaire dans le même composant de source ou dans un composant de source différent.

## 5.3.3 Exemple : Compter les lignes de table de base de données

Le mappage illustré dans cet exemple est disponible en tant que **DB_UserList.mfd** dans le dossier **<Documents>\Altova\MapForce2024\MapForceExamples**. Ce mappage extrait des enregistrements d'utilisateur depuis une table de base de données appelée "Users" et les écrit dans un fichier XML. La colonne de base de données "Username" contient aussi bien le prénom que le nom d'une personne (par exemple, "Vernon Callaby"). Ce mappage présente les objectifs suivants :

1. Pour chaque enregistrement dans la table "Users", créer un nouvel élément `Person` dans le fichier XML.
2. Partager la valeur extraite depuis le champ de base de données "Username" en deux champs séparés dans le fichier XML ("First" et "Last").
3. Pour chaque enregistrement, trouver le numéro séquentiel comparé au nombre des enregistrements totaux présents dans la base de données (par exemple, "Record 1 of 4") et écrire cette information dans l'élément `Details`.




*DB_UserList.mfd*

Comme illustré ci-dessus, afin d'atteindre le premier objectif, une connexion est établie entre la table de source "Users" et l'élément `Person` du fichier XML cible. Cela permet d'assurer que, pour chaque enregistrement dans la table de source, un nouvel élément `Person` sera créé dans la cible.

La valeur du champ "Username" est fournie dans les fonctions **`substring-before`** [634] et **`substring-after`** [633]. Ces deux fonctions extraient le texte avant et après le caractère d'espace (" "), respectivement, ce qui permet de finaliser le deuxième objectif de mappage.

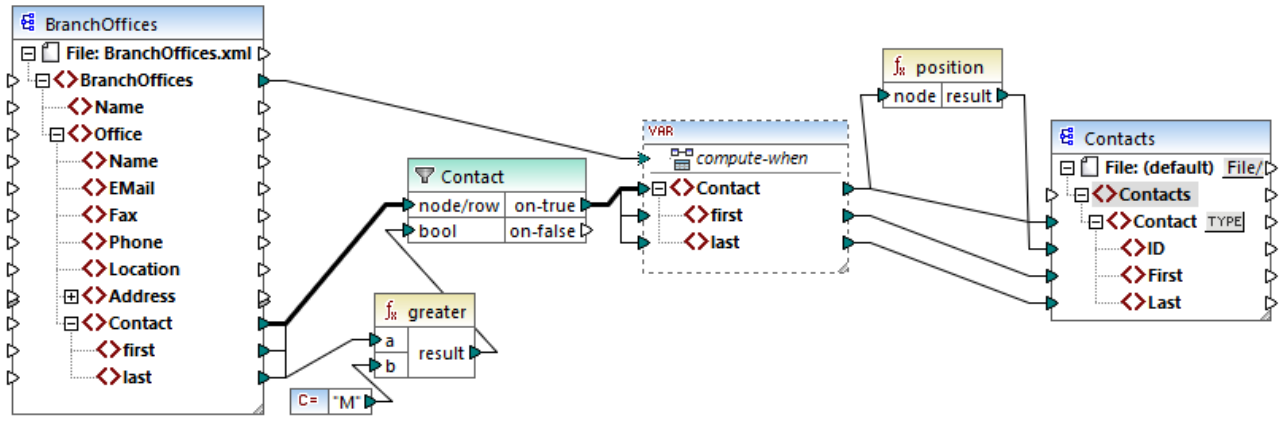
Enfin, pour atteindre le troisième objectif, le mappage utilise la fonction **`count`**. Le résultat de la fonction **`count`** est passé dans la variable. La variable permet d'assurer que ce résultat est stocké dans le mappage et est disponible lors de l'écriture de l'élément "Details" de chaque personne dans le XML cible. Veuillez noter que, pour des raisons d'efficacité, les enregistrements de base de données ne doivent être comptés qu'une seule fois, l'étendue de la variable est donc définie à `compute-when=once` (voir Changer le contexte et l'étendue des variables [419])

## 5.3.4 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 444 )
- Une variable complexe (voir Ajouter des variables 415)
- Les fonctions **greater** 581 et **position** 618 (voir Ajouter une fonction au mappage 468)
- 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** 581 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.

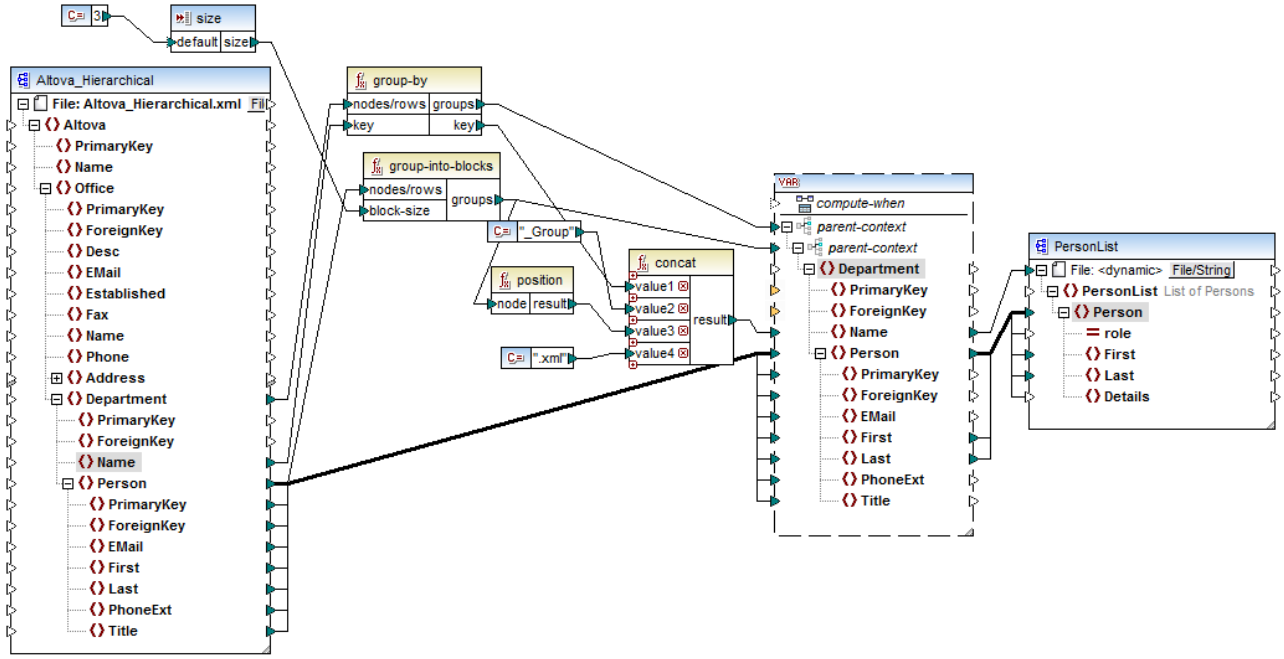
Les fonctions **position** 618 retournent, pour chaque itération de la variable, le nombre de séquences actuelles. Seuls huit contacts remplissent la condition de filtre ; c'est pourquoi, si vous consultez le mappage et regardez la sortie, il est recommandé de noter comment les ID 1 à 8 ont été écrites 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.

## 5.3.5 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 419).

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`** 605 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** 611 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 398.

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** 629 :

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"

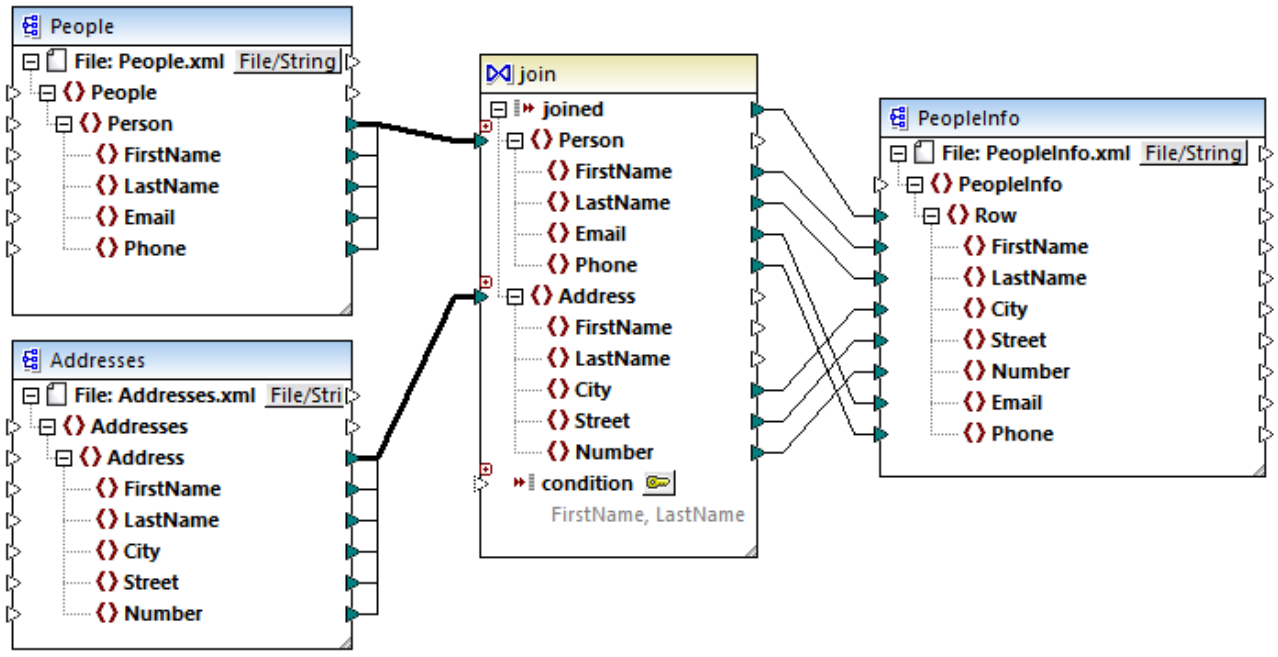
Le résultat de cette concaténation est stocké dans l'item `Name` de la variable, puis est fourni en tant que nom de fichier dynamiques dans le composant cible. Cela entraîne la création d'un nouveau nom de fichier pour chaque valeur reçue. Dans cet exemple, la variable calcule huit groupes au total, 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 787.

## 5.4 Joindre des données

Parfois, vous souhaitez combiner des données depuis deux structures ou plus sur la base de conditions (par exemple, si le champ A dans la première structure a la même valeur que les champ B dans la deuxième structure). Pour ce type de mappage, vous pouvez utiliser un composant Join.

Un composant Join est un composant MapForce qui permet de joindre deux structures ou plus dans le mappage sur la base de conditions définies par l'utilisateur. Il retourne l'association (ensemble joint) d'items qui satisfont à la condition. Les composants Join sont particulièrement utiles pour combiner des données provenant de deux structures qui partagent un champ commun (comme une identité).

Par exemple, dans le mappage illustré ci-dessous, le composant central est un composant "Join". Dans ce mappage, deux structures XML (une liste de personnes et une liste d'adresses) sont jointes. L'objectif est d'obtenir tous les détails de chaque personne dans un fichier XML cible. Les champs `FirstName` et `LastName` agissent en tant que clés de jointure. Si la valeur de `FirstName` et `LastName` (sous `Person`) est la même que celle de `FirstName` et `LastName` (sous `Address`), les détails d'adresse correspondent à une seule et même personne et ils deviennent "joins". Tous les items provenant de la structure jointe peuvent aussi être mappés dans une autre cible (dans ce cas, un fichier XML). La condition Join elle-même est définie dans les propriétés du composant Join, en cliquant sur la touche **Définir condition Join** ( ). Cet exemple s'accompagne d'un échantillon de mappage et est expliqué de manière détaillée dans Exemple : Joindre des structures XML 432.



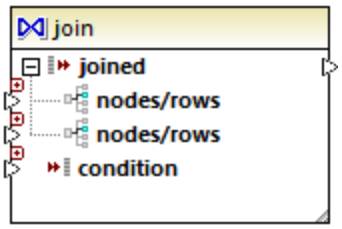

*JoinPeopleInfo.mfd*

Comme illustré ci-dessus, les structures de source et le composant Join sont liées par des connexions "Copier tout", qui réduisent l'encombrement visuel du mappage. En général, ce type de connexion est créé automatiquement par MapForce lorsque le contexte est pertinent (pour plus d'informations, voir Connexions Copier tout 60).

Les structures qui doivent être jointes peuvent provenir soit de composants séparés (comme dans le mappage ci-dessus), soit appartenir au même composant. Les structures à joindre peuvent aussi être de genres différents (par exemple, une structure XML et une table de base de données). Pour plus d'informations concernant des jointures liées aux bases de données, voir Joindre les données de base de données 348.

### Pour ajouter un composant Join :

1. Définir le langage de transformation de mappage sur BUILT-IN (pour ce faire, cliquer soit sur la touche de la barre outils , soit utiliser la commande de menu **Sortie | Moteur d'exécution Built-In**).
2. Dans le menu **Insérer**, cliquer sur **Join**. En alternative, cliquer sur la touche de la barre outils **Join** ( ). Le composant Join apparaît dans le mappage. Par défaut, il accepte des données provenant de deux structures, il a donc deux entrées `nodes/rows`. Si nécessaire, vous pouvez ajouter de nouvelles entrées dans la jointure en cliquant sur la touche **Ajouter entrée** ( ), voir Joindre trois structures ou plus 431.




3. Connecter les structures qui doivent être jointes dans l'item `nodes/rows` du composant join.
4. Ajouter la condition pour la jointure (ou plusieurs conditions). Pour ce faire, cliquer avec la touche de droite dans le composant Join et choisir **Propriétés**. Les conditions Join peuvent aussi être ajoutées directement depuis le mappage, en connectant le résultat Booléen de certaines fonctions à l'item `condition` du composant Join. Dans certains cas. lorsque des tables de base de données sont jointes, la condition (ou les conditions) de jointure peut être créée automatiquement par MapForce. Pour plus d'informations, voir Ajouter des conditions Join 802.

Notes :

- Les composants Join sont pris en charge lorsque la langue cible du mappage est configurée sur BUILT-IN. La génération de code dans C#, C++, ou Java n'est pas prise en charge.
- Lorsqu'une structure n'est pas une source d'entrée valide ou prise en charge pour la jointure, MapForce affiche des indices soit immédiatement et directement dans le mappage, soit dans la fenêtre Messages, lorsque vous validez le mappage (voir Valider des mappages 68).
- Les composants Join ne doivent pas être connectés aux paramètres d'entrée ou aux résultats des fonctions définies par l'utilisateur inline. Si de telles connexions existent, des erreurs de validation se produiront au cours de la validation de mappage.
- Lorsque vous vous connectez à des composants de base de données éligibles (comme des tables ou des modes) directement dans un composant Join, une touche **mode SQL** ( ) apparaît automatiquement en haut à droite du composant Join. Une fois activée, cette touche offre des fonctions SQL spéciales applicables à l'opération de jointure (voir Joins dans le mode SQL 349).
- Il n'est pas possible de connecter la sortie de l'item `joined` à un autre composant Join. Néanmoins, le cas échéant, vous pouvez connecter un résultat partiel à une jointure d'un autre.

### Composants Join comparés à d'autres types de composant

Dans certains cas, des variables complexes ou des filtres peuvent peuvent être utilisés pour obtenir les mêmes résultats (voir Utiliser des variables 413 et Filtres et conditions 444, respectivement). Néanmoins, contrairement à d'autres types de composant, les composants Join permettent d'obtenir un mappage plus aisément compréhensible, étant donné que vous pouvez voir en un coup d'œil les données jointes. De plus, si le mode SQL est activé dans le composant Join, la performance de mappage s'améliore considérablement (cela s'applique aux jointures de base de données, voir Joindre les données de base de données 348).

### Ajouter un contexte parent

Dans certains cas particuliers, afin d'obtenir un résultat de mappage spécifique, vous pouvez fournir explicitement un contexte de mappage (soit-disant "contexte parent") pour les données connectées au composant Join. Pour ajouter un contexte parent, cliquer avec la touche de droite sur l'item `joined` du composant Join, et choisir **Ajouter Contexte Parent** depuis le menu contextuel. Le composant Join change d'apparence pour inclure une entrée `parent-context` supplémentaire dans laquelle vous pouvez connecter l'item de source requis. Pour plus d'informations, voir Exemple : Changer le contexte Parent 810.

`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.

## 5.4.1 Ajouter des conditions Join

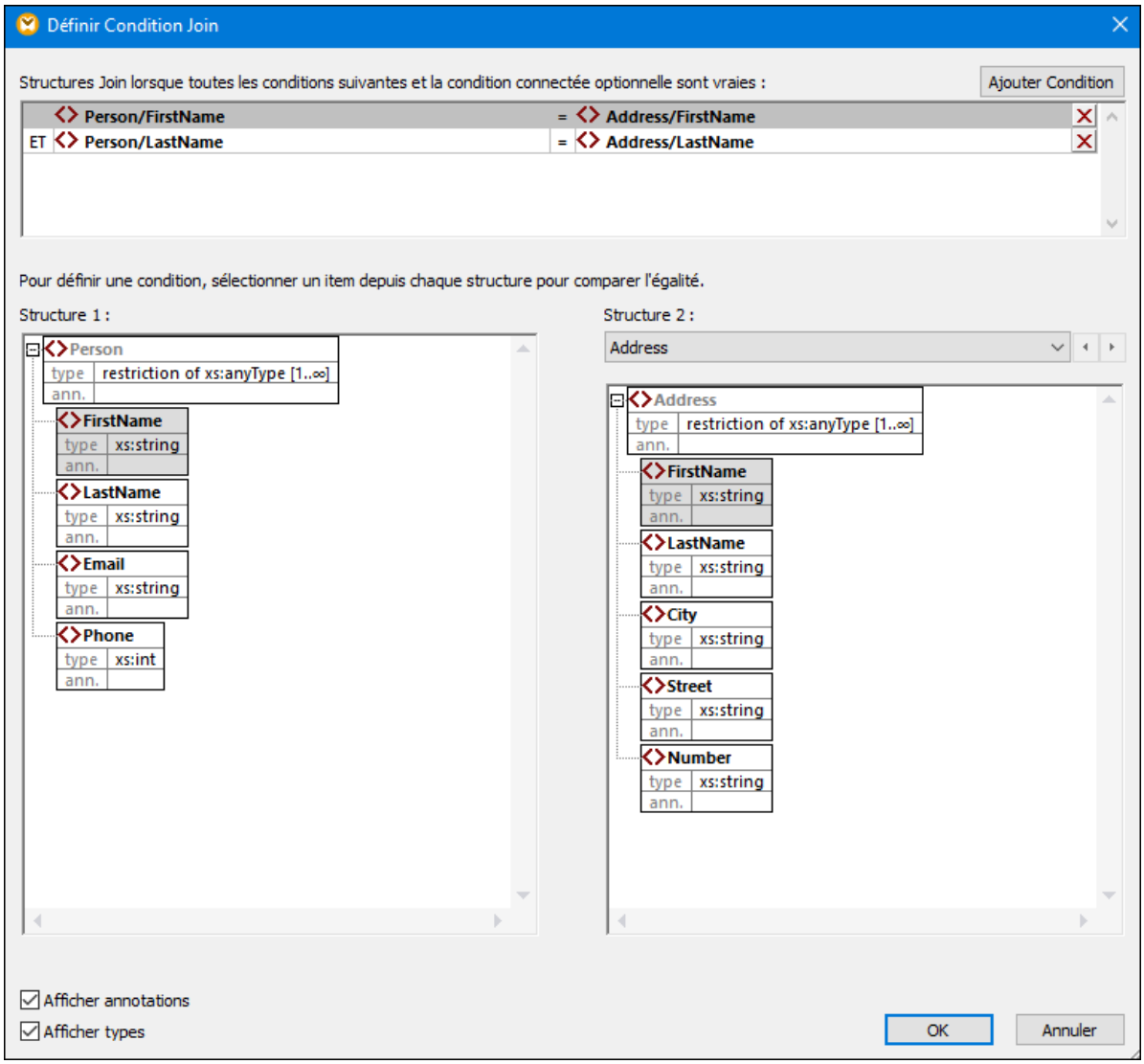
Un "join" fonctionne en combinant des items de deux ou plusieurs structures conformément à une condition, de manière qu'un join exige toujours au moins une condition. Il existe plusieurs moyens d'ajouter des conditions "join", voir ci-dessous.

**Note :** Lorsque des tables de base de données sont jointes dans le mode SQL, MapForce créera la condition Join (ou les conditions) automatiquement, sur la base des relations de clé étrangère détectée entre les tables. Pour que des conditions Join automatiques se produisent, les tables de base de données doivent se trouver dans une relation enfant-parent dans le composant MapForce (c'est à dire, qu'une table doit être "parent" ou "child" d'un autre dans le composant), voir Exemple: Joindre des tables dans le mode SQL 352.

### Approche 1: Ajouter une condition join depuis les propriétés de composant

1. Dans le mappage, veuillez vous assurer qu'au moins deux structures (ou tables de base de données) sont connectées au composant Join. Le composant Join illustré dans cet exemple fait partie du mappage **JoinPeopleInfo.mfd** disponible dans le dossier **<Documents>\Altova\MapForce2024\MapForceExamples\Tutorial**. Ce mappage est discuté en plus grands détails dans Exemple : Joindre des structures XML 432.
2. Dans le composant Join, cliquer sur la touche **Définir condition Join** (  ) (ou cliquer avec la touche de droite sur le composant et sélectionner **Propriétés** depuis le menu contextuel).
3. Choisir un item depuis la structure de gauche et un autre depuis la structure de droite (ainsi, lorsque la comparaison de cette paire retourne vrai, les structures de gauche et de droite sont jointes).




Si vous souhaitez ajouter plusieurs conditions, cliquer sur **Ajouter condition**, puis sélectionner une nouvelle paire d'items. Par exemple, dans l'image ci-dessus, deux conditions join sont définies :

1. `FirstName` dans la Structure 1 doit être égale à `FirstName` dans la structure 2, et
2. `LastName` dans la structure 1 doit être égale à `LastName` dans la structure 2.

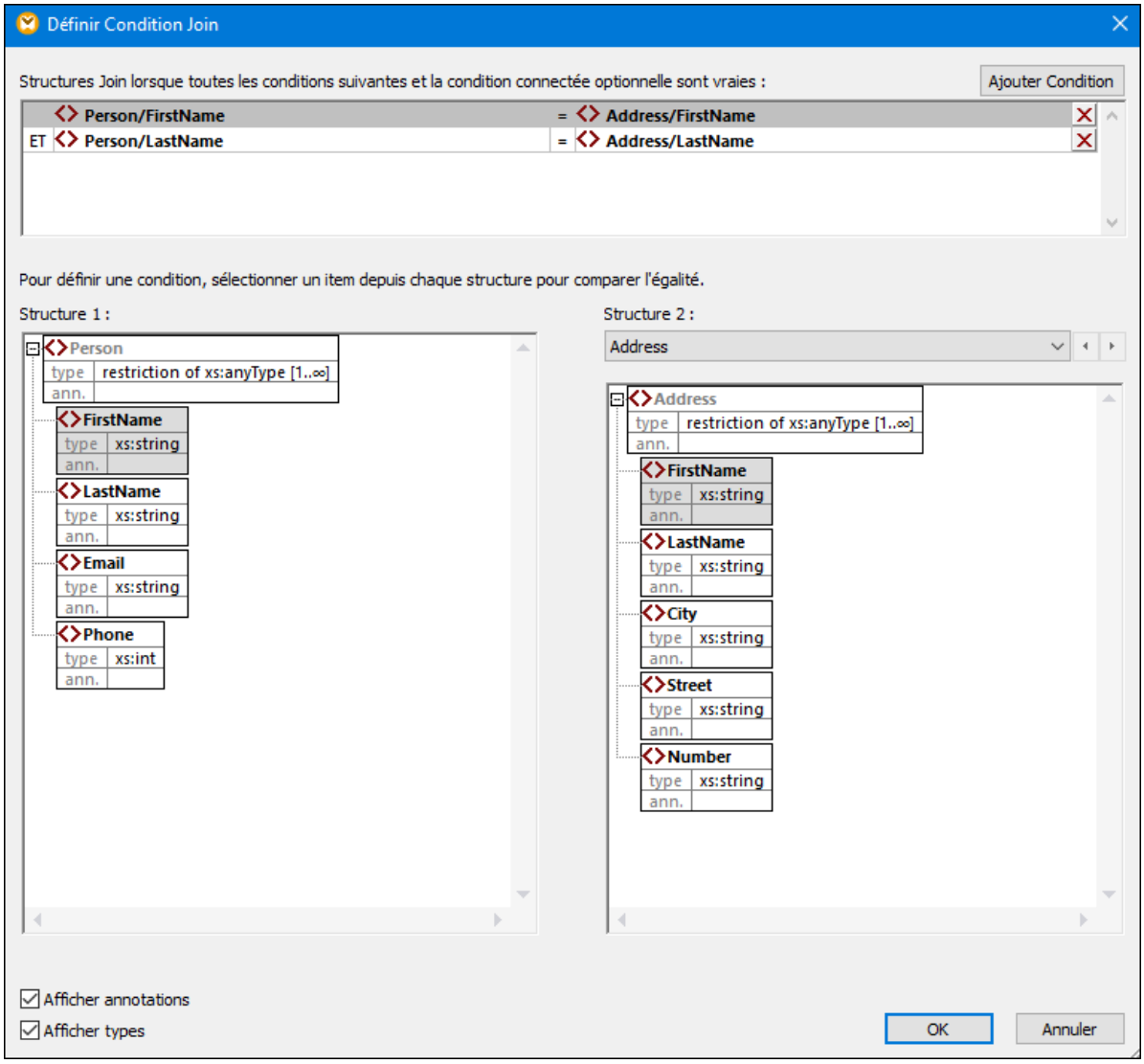
Pour supprimer une condition join, cliquer sur la touche **Supprimer**  située juste à côté.

Notes :

- Si plusieurs conditions join existent, toutes doivent être satisfaites pour que les deux structures soient jointes. En d'autres termes, plusieurs conditions sont jointes par une opération logique AND. Cela comprend également des conditions optionnelles qui ont été ajoutées depuis le mappage (voir Approche 2 ci-dessous).
- Si plus de deux structures sont connectées au composant Join, ce type de structure supplémentaire apparaît dans la liste déroulante ci-dessous "Structure 2". Lorsque vous sélectionnez une telle

structure supplémentaire depuis la liste déroulante, le panneau de gauche affiche toutes les structures qui se produisent *avant* dans le composant Join. Ainsi, vous pouvez définir des conditions "join" se trouvant entre une des structures multiples. Pour consulter un exemple, voir Exemple : Créer un rapport CSV à partir de plusieurs tables 361.

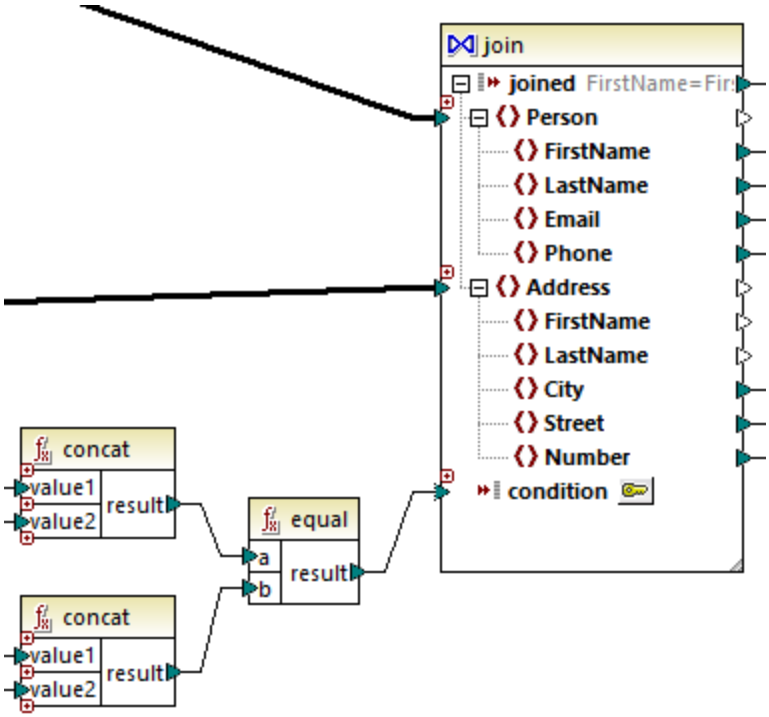
- Pour consulter le type de données des items dans chaque structure, cocher la case **Afficher types**. L'option **Afficher annotations** affiche des informations supplémentaires concernant les items, si de telles informations existent dans le schéma sous-jacent (ou la base de données). Si les deux cases sont cochées, la mise en page change pour accommoder la vue des deux annotations et types, par exemple :




## Approche 2 : Ajouter une condition join depuis le mappage

- Dans le mappage, ajouter des composants qui produisent une valeur booléenne, puis connecter la sortie booléenne à l'entrée de l'item `condition`. Par exemple, la fonction **`equal`** peut comparer une valeur avec des items de mappage, et fournir le résultat booléen en tant qu'entrée dans l'item `condition` du composant join.



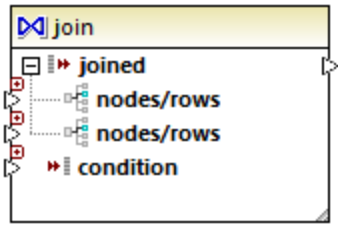

**Note :** Si aucune condition n'est définie depuis les propriétés de composant join (Approche 1), l'item `condition` du composant join doit être connecté (Approche 2).

### Approche 3: Approche mixte

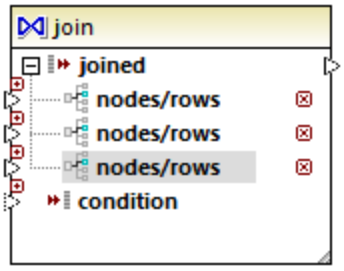
Dans le même mappage, il est possible de définir des conditions join dans les propriétés de composant (Approche 1) et de les combiner avec celles provenant du mappage (Approche 2). Néanmoins, si vous avez l'intention de joindre des tables de base de données dans le mode SQL, les conditions doivent être définies strictement conformément à l'Approche 1, voir aussi Joins dans le mode SQL 349.

## 5.4.2 Joindre trois structures ou plus

Lorsque vous ajoutez un composant Join au mappage en utilisant la commande de menu **Insérer | Join**, il accepte deux structures par défaut (c'est à dire que le composant ne contient que deux entrées `nodes/rows`).




Si vous nécessitez plus de deux structures, cliquer sur la touche **Ajouter entrée** ( ⊡ ) et créer autant de `nodes/rows` que nécessaire. Si vous devez supprimer une entrée `nodes/rows`, cliquer sur la touche **Supprimer entrée** ( ⊠ ). Veuillez noter qu'une jointure nécessite au moins deux structures, la touche ⊠ est uniquement disponible si plus de deux entrées existent.




Lorsqu'une jointure a plusieurs entrées, les conditions de jointure doivent donc prendre en compte chacune des entrées que vous souhaitez joindre, voir Ajouter des condition Join 428. Pour un exemple étape par étape de la manière de joindre plusieurs tables de base de données, voir Exemple : Créer un rapport CSV depuis plusieurs tables 361.

## 5.4.3 Exemple : Joindre des structures XML

Cet exemple illustre comment combiner des données provenant de deux structures XML conditionnellement, en utilisant un composant Join. L'exemple est accompagné par un échantillon de mappage, disponible sous : **<Documents>\Altova\MapForce2024\MapForceExamples\Tutorial\JoinPeopleInfo.mfd**.

L'objectif de ce mappage est de collecter des données personnelles (prénom, nom de famille, adresse, e-mail et téléphone) depuis deux fichiers XML de source dans un seul fichier XML cible.

Le premier fichier XML stocke le prénom et le nom de chaque personne, ainsi que leur adresse e-mail et leur numéro de téléphone, comme indiqué dans la liste de code ci-dessous (veuillez noter que la déclaration XML, les espaces de noms et certains enregistrements ont été omis pour des raisons de simplicité) :

```
<People>
   <Person>
      <FirstName>Marquita</FirstName>
      <LastName>Bailey</LastName>
      <Email>m.bailey@nanonull.com</Email>
      <Phone>555323698</Phone>
   </Person>
   <Person>
      <FirstName>Totie</FirstName>
      <LastName>Rea</LastName>
      <Email>t.rea@nanonull.com</Email>
      <Phone>555598653</Phone>
   </Person>
</People>
```

*People.xml*

Le deuxième fichier XML stocke le prénom et le nom de chaque personne, ainsi que leur adresse e-mail :

```
<Addresses>
   <Address>
      <FirstName>Marquita</FirstName>
```

```
      <LastName>Bailey</LastName>
      <City>Bridgedell</City>
      <Street>Olive Street</Street>
      <Number>4</Number>
   </Address>
   <Address>
      <FirstName>Totie</FirstName>
      <LastName>Rea</LastName>
      <City>Roseford</City>
      <Street>Evergreen Lane</Street>
      <Number>34</Number>
   </Address>
</Addresses>
```

*Addresses.xml*

L'objectif du mappage est de combiner les informations <Person> depuis le premier fichier avec l'information <Address> provenant du second fichier, à chaque fois que les prénom et nom correspondent. Spécifiquement, pour chaque <Person> dans le premier fichier et pour chaque <Address> dans le deuxième fichier, les FirstName et LastName doivent être comparés. Si les deux valeurs sont les mêmes, les enregistrements <Person> et <Address> correspondants se réfèrent à la même personne et doivent être joints. La structure XML de cible doit ressembler à :

```
<PeopleInfo>
   <Row>
      <FirstName>Marquita</FirstName>
      <LastName>Bailey</LastName>
      <City>Bridgedell</City>
      <Street>Olive Street</Street>
      <Number>4</Number>
      <Email>m.bailey@nanonull.com</Email>
      <Phone>555323698</Phone>
   </Row>
   <Row>
      <FirstName>Totie</FirstName>
      <LastName>Rea</LastName>
      <City>Roseford</City>
      <Street>Evergreen Lane</Street>
      <Number>34</Number>
      <Email>t.rea@nanonull.com</Email>
      <Phone>555598653</Phone>
   </Row>
</PeopleInfo>
```

*PeopleInfo.xml*

Le but de ce mapping est facilement atteignable en ajoutant un composant Join dans le mappage. Veuillez noter qu'il est également possible d'atteindre le même résultat en utilisant les autres types de composant ; néanmoins, dans les étapes ci-dessous, vous utiliserez un composant Join, qui est le sujet de cet exemple.

Pour créer le mappage requis, suivre les étapes suivantes.

## Étape 1: Ajouter les fichiers XML de source au mappage

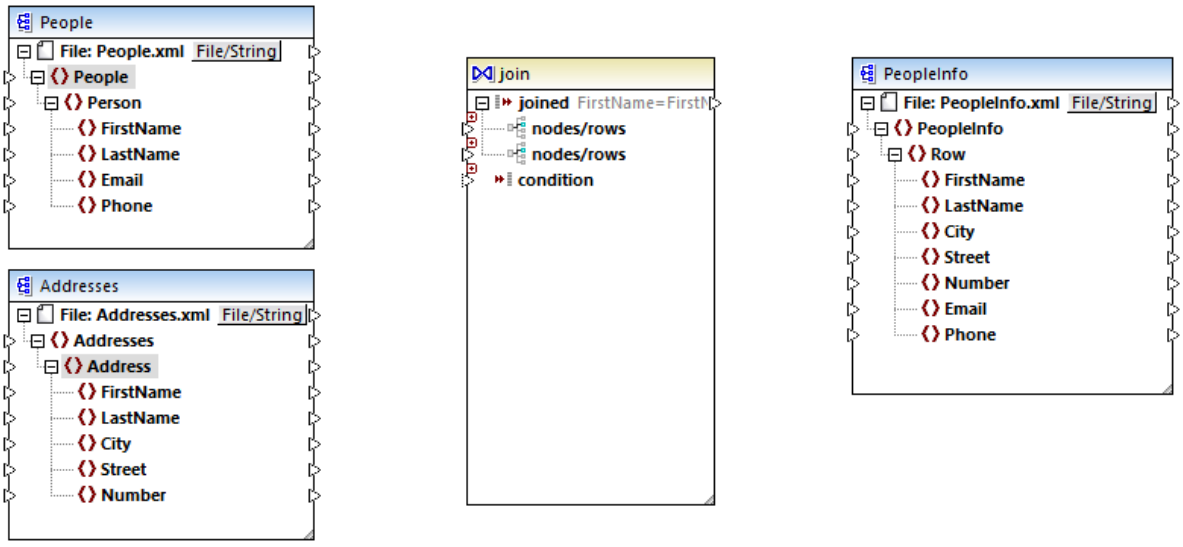
1. Dans le menu **Insérer**, cliquer sur **Schéma XML/Fichier**, et chercher le fichier de source suivant : **<Documents>\Altova\MapForce2024\MapForceExamples\Tutorial\People.xml**.
2. Répéter l'étape ci-dessus pour **Addresses.xml** (le deuxième fichier de source).

## Étape 2: Ajouter le fichier de schéma cible au mappage

- Dans le menu **Insérer**, cliquer sur **Schéma XML/Fichier**, et chercher **<Documents>\Altova\MapForce2024\MapForceExamples\Tutorial\PeopleInfo.xsd** (le fichier de schéma XSD cible). Lorsque vous serez invité à fournir un fichier XML d'échantillon, cliquez sur **Sauter**. Lorsque vous serez invité à choisir un élément racine, choisissez `PeopleInfo` en tant qu'élément racine.

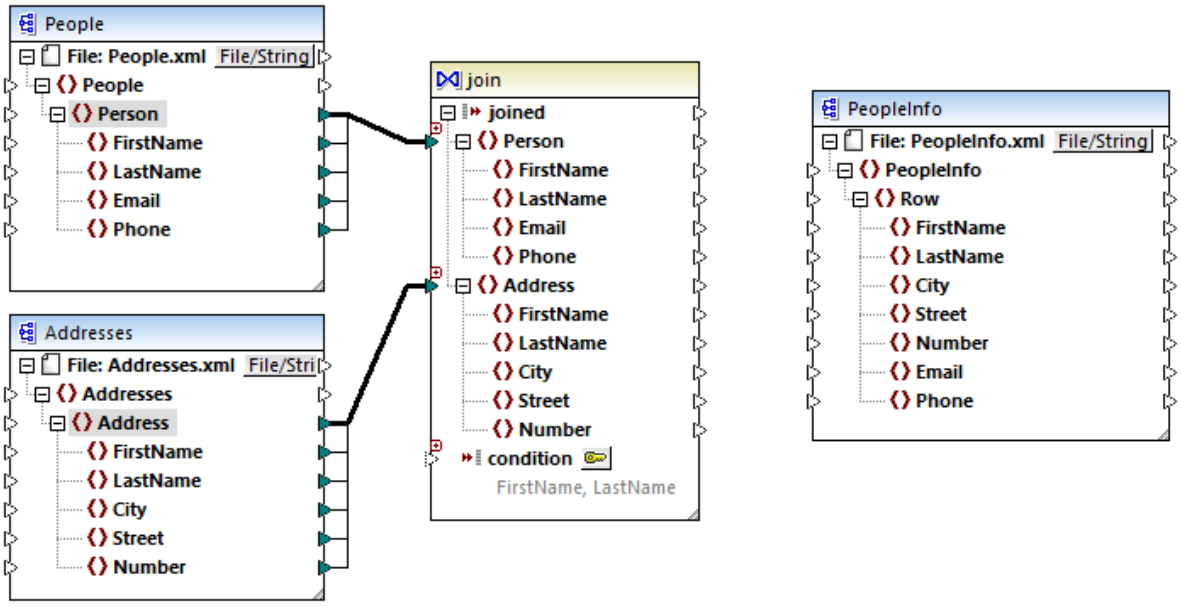
## Étape 3: Ajouter le composant Join

1. Dans le menu **Insérer**, cliquer sur **Join**. (ou bien cliquer sur la touche de barre d'outils **Join** ). À ce niveau, le mappage devrait ressembler à l'exemple suivant (vous devrez glisser et redimensionner les composants pour les faire ressembler au graphique ci-dessous) :

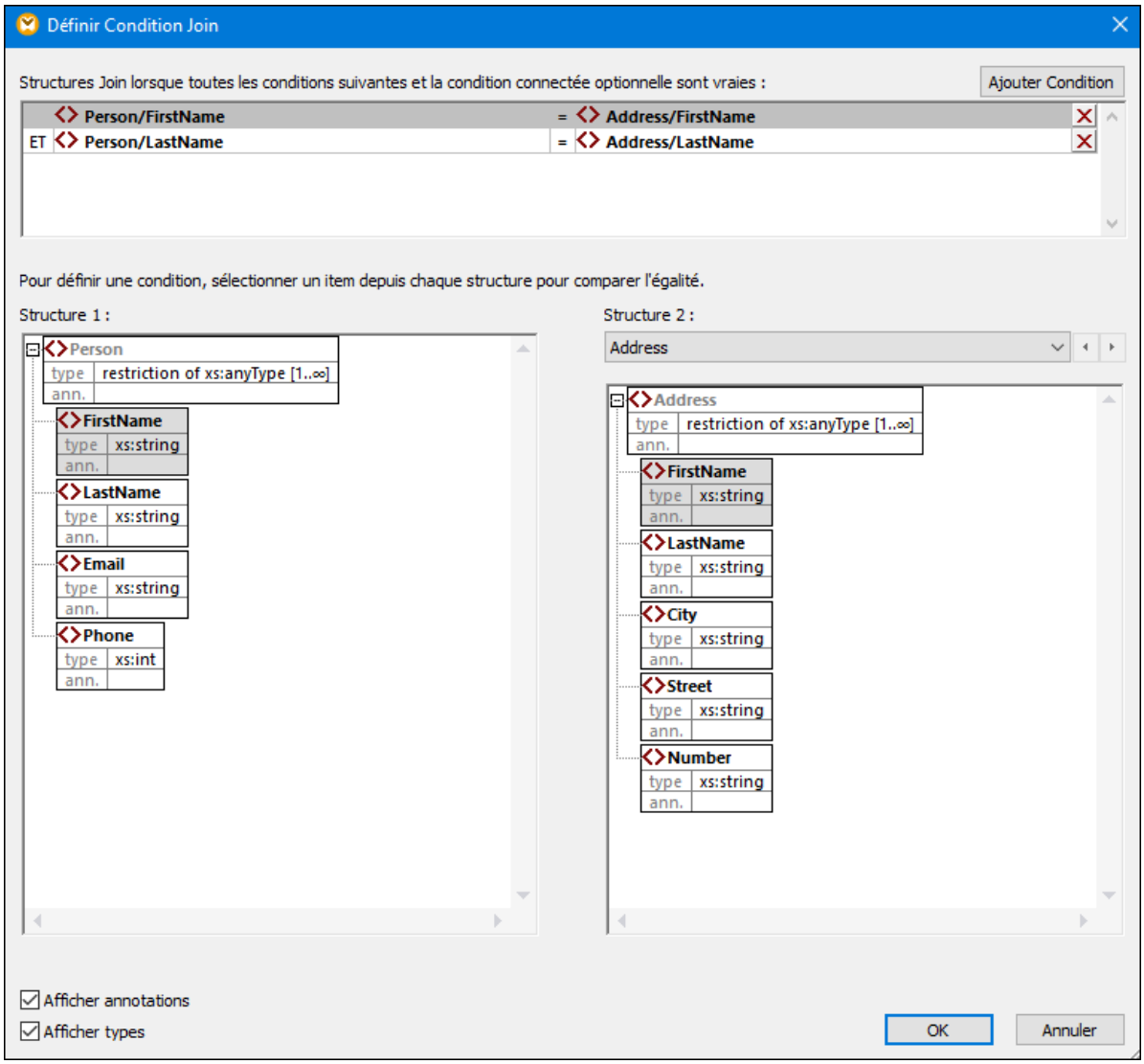



Observez la structure du composant Join. Elle contient deux items `nodes/rows`, qui permettent d'y connecter les deux structures qui doivent être comparées (dans ce cas, les structures `Person` et `Address`).

2. Tracer une connexion depuis `Person` vers le premier item `nodes/rows` du composant Join. De même, connecter `Address` au deuxième item `nodes/rows`.




3. Comme mentionné plus tôt, la jointure doit avoir lieu uniquement si les valeurs `FirstName` et `LastName` sont égales dans les deux structures. Pour définir cette condition, cliquer sur la touche **Définir Condition Join** .
4. Choisir la paire des items qui définit la première condition Join (`FirstName` sous Structure 1, et `FirstName` sous Structure 2).
5. Cliquer **Ajouter condition**, et répéter l'étape ci-dessus pour `LastName`.



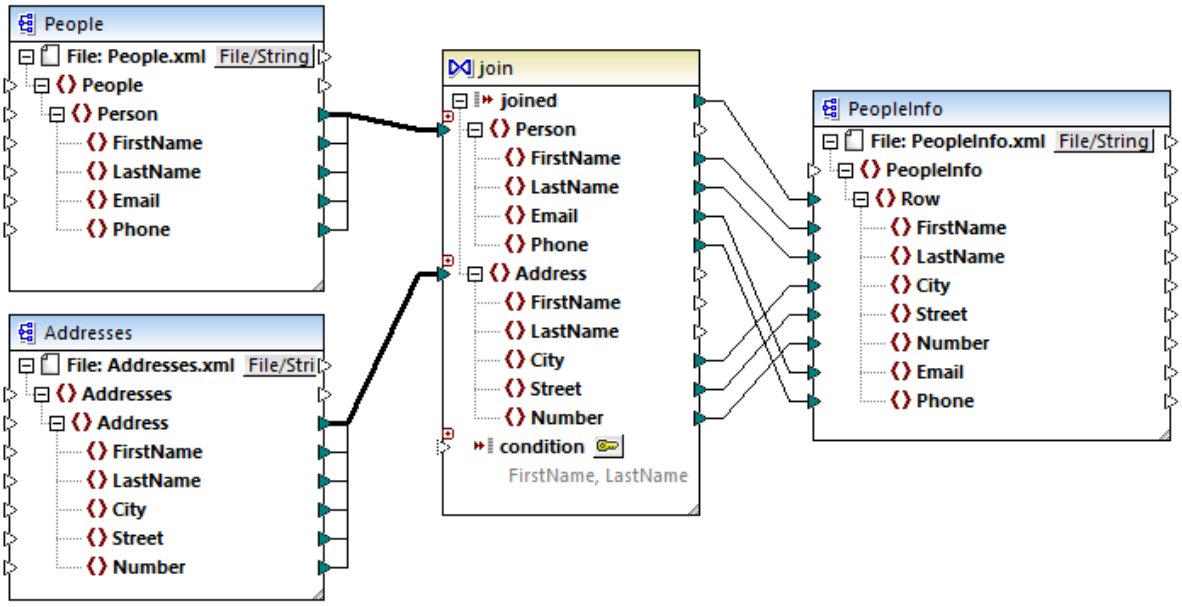

Dans certains mappage, une condition consistant en une comparaison peut suffire à effectuer la jointure. Néanmoins, dans cet exemple, il est important de créer deux comparaisons :

1) `FirstName` dans Structure1 = `FirstName` dans Structure 2
2) `LastName` dans Structure 1 = `LastName` dans Structure 2.

Si plusieurs conditions sont définies, *elles doivent toutes être vraies pour que la jointure puisse avoir lieu*. C'est pourquoi, dans cet exemple, une jointure n'aura lieu que si les deux comparaisons sont vraies (ce qui est le comportement souhaité). Si une seule des comparaisons ci-dessus a été définie, une jointure peut se produire pour des personnes ayant le même prénom mais des noms de famille différents.

## Étape 4: Mapper le composant Join vers le schéma de cible

Maintenant que les deux structures sont jointes, vous pouvez définir quels items de la structure jointe doivent être mappés dans la cible. Pour ce faire, créer des connexions depuis les items dans les deux structures jointes vers le composant de cible, comme indiqué ci-dessous. La connexion entre `joined` et `Row` a l'objectif suivant : lorsque la condition Join est satisfaite, elle crée un nouvel item `Row` dans la cible.



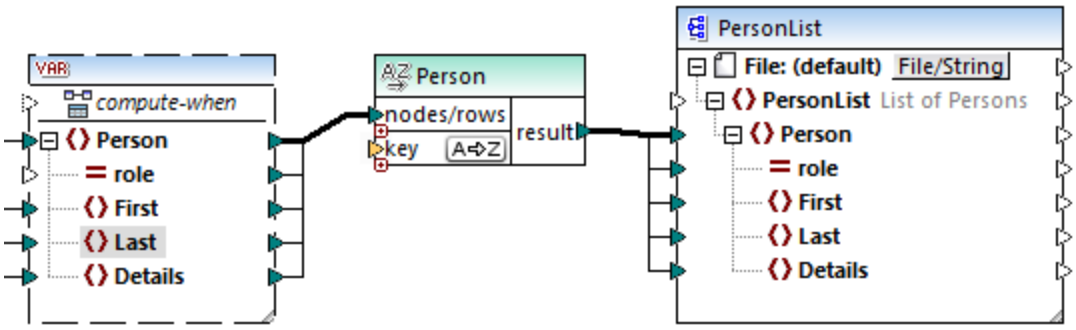

Pour consulter la sortie de mappage, cliquer sur l'onglet **Sortie**. Comme prévu, chaque enregistrement de personne (<Row>) comprend maintenant tous les détails d'adresse, joints depuis deux sources différentes.

# 5.5 Trier les données

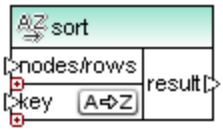
Afin de trier les données basées sur une clé de tri spécifique, utilisez un composant Sort (Tri). Le composant Tri prend en charge les langues cible suivantes : XSLT2, XQuery, et Built-in. Lorsque le langage de transformation est "Built-in", le composant Tri peut être utilisé pour trier les données de table de la base de données. Néanmoins, il est possible d'obtenir une meilleure performance par le biais d'un composant SQL-WHERE/ORDER. Pour plus de détails, voir Filtrer et trier des données de base de données (SQL WHERE/ORDER) 365.

**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).



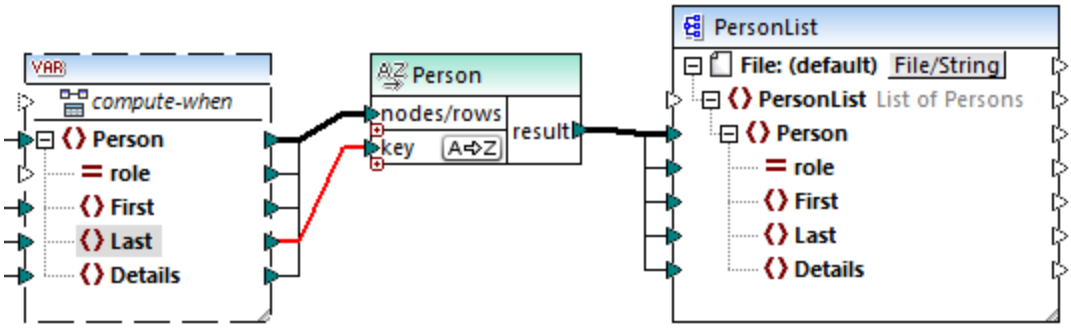

- Dans le menu **Insérer**, cliquer sur **Sort** (ou bien cliquer sur la touche de la barre d'outils **Sort** ). Le composant Sort est inséré dans sa forme "non-connectée".




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 `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`.

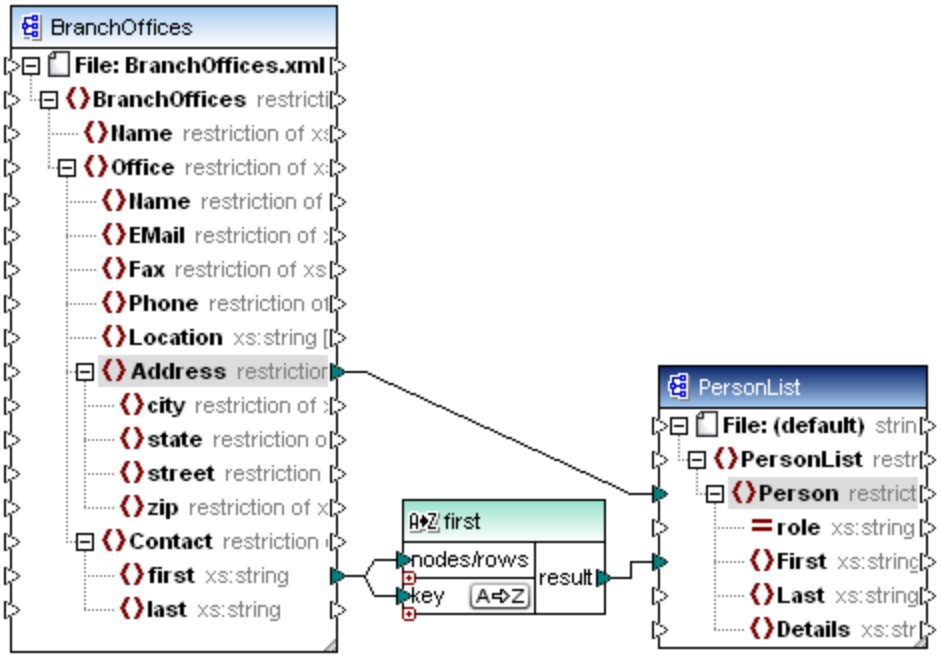



**Pour modifier l'ordre de tri :**

- 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.

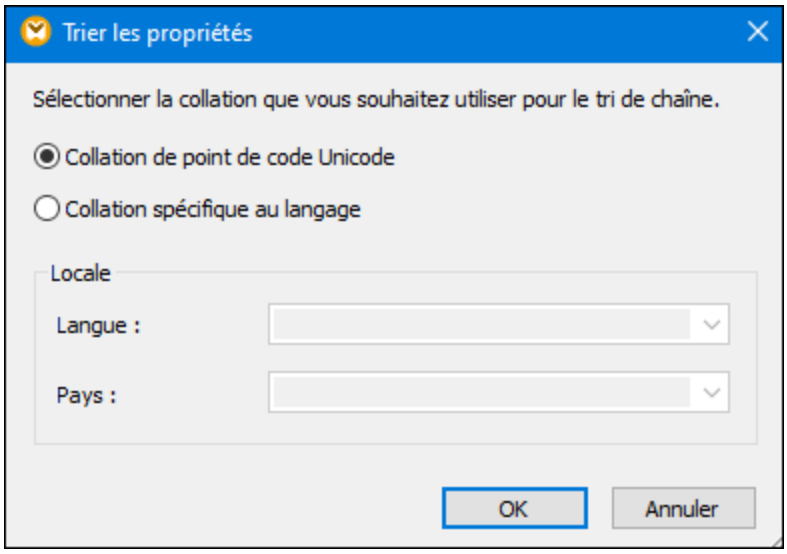
**Pour trier des données d'entrée consistant en des items de type simple :**

- 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.

## 5.5.1 Trier par clés multiples

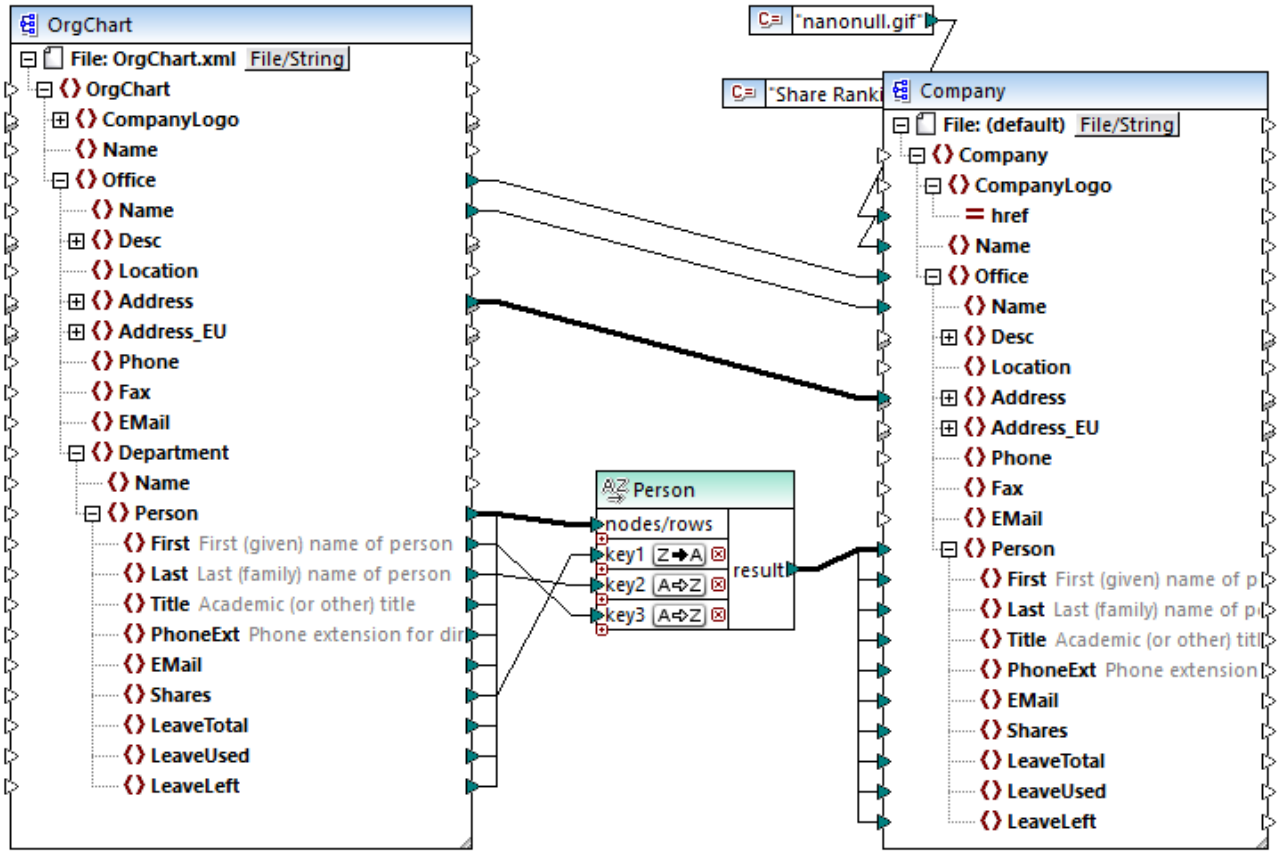
Une fois avoir ajouté un composant Sort au mappage, une clé de tri appelée `key` sera créée par défaut.


*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)

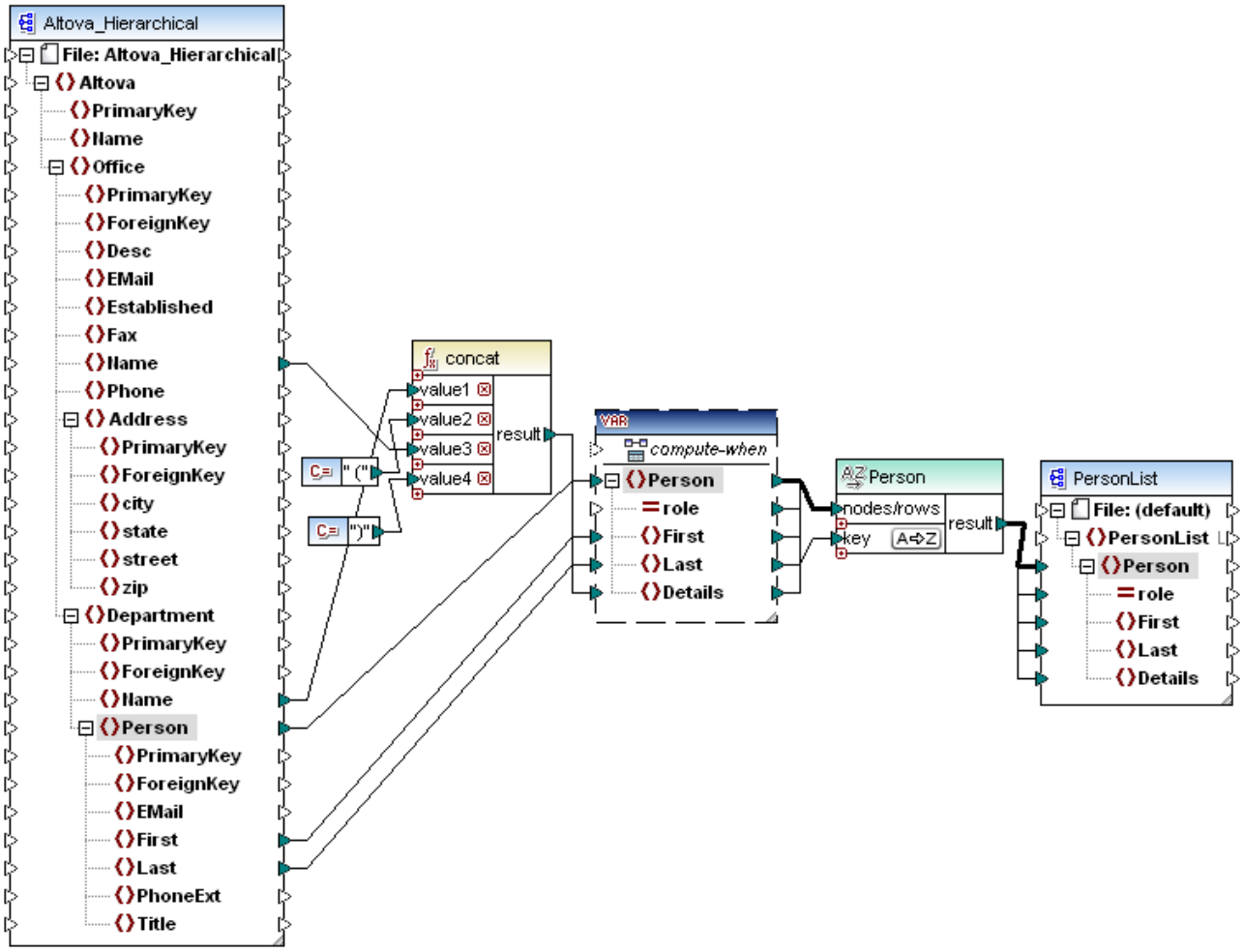
Veuillez noter que la position de la clé de tri dans le composant Sort détermine sa priorité de tri. Par exemple, dans le mappage ci-dessus, les enregistrements sont tout d'abord triés par le nombre de parts. Il s'agit de la clé de tri avec la priorité la plus élevée. Si le nombre de parts est le même, 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).

## 5.5.2 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[468] .

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[802]). 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[413] et Connexions Copier tout[60].
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.

# 5.6 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 ( )
- SQL WHERE/ORDER ( )
- 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 ou des lignes CSV, 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 :

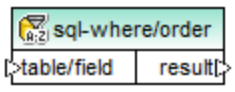

Dans la structure ci-dessus, la condition connectée à **bool** détermine si le **nœud/ligne** connecté va bien vers la sortie **on-true** ou **on-false**. Concrètement, si la condition est vraie, le **nœud/ligne** sera redirigé vers la sortie **on-true**. Ainsi, si la condition est fausse, le **nœud/ligne** 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. Si vous souhaitez ajouter une exception lorsque la condition du filtre n'est pas remplie, il est obligatoire de connecter la sortie **on-false** (voir Ajouter des exceptions 463).

Pour un exemple de mappage étape par étape, voir Exemple : Filtrer des nœuds 446.

## Filtrer les données de base de données

Les composants **Filtrer nœuds/lignes** peuvent filtrer des données depuis toute autre structure de composant prise en charge par MapForce, y compris des bases de données. Néanmoins, si vous souhaitez filtrer des données depuis une base de données, il est recommandé d'utiliser un composant **SQL WHERE/ORDER** à la place. Ce composant **SQL WHERE/ORDER** est optimisé pour travailler avec des bases de données et fournit une meilleure performance qu'un composant **Filtrer nœuds/lignes**.




Pour plus d'informations concernant ces composants, voir Composant SQL WHERE / ORDER 365.

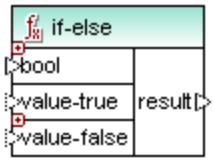
## Retourner une valeur par condition

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>
```

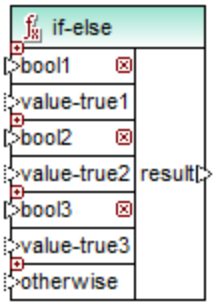
Une **Condition If-Else** vous permettrait de retourner, pour chaque item dans la liste, la valeur "high" si la température dépasse 20 degrés Celsius, et la valeur "low" si la température est inférieure à 5 degrés Celsius.

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.



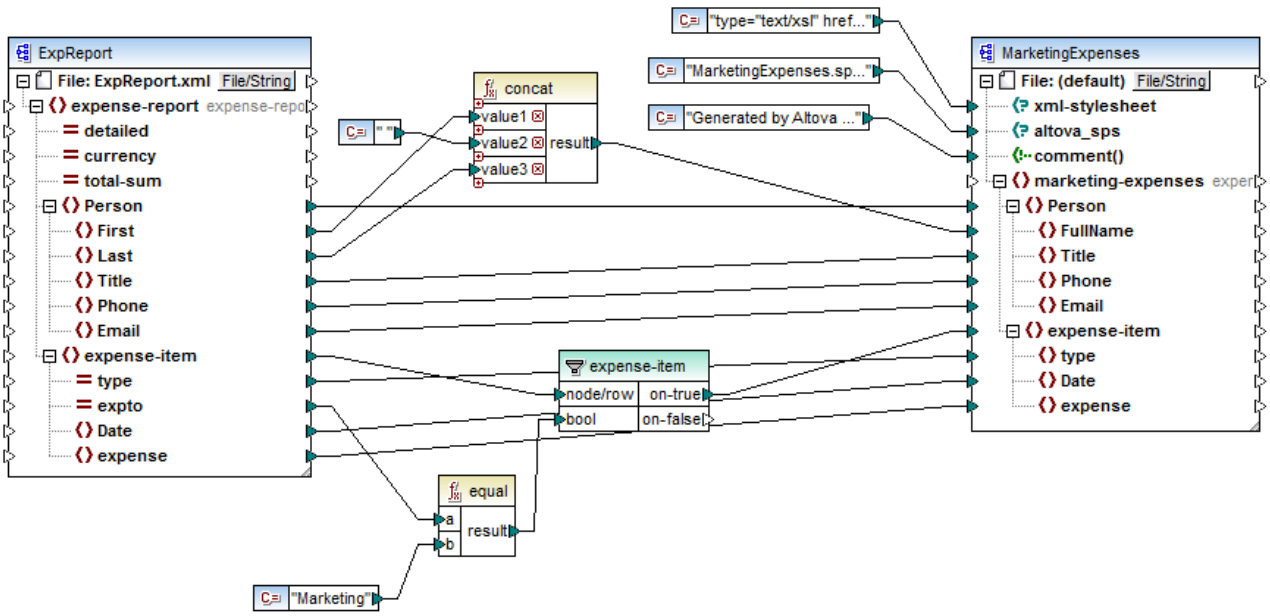

Les **Conditions If-Else** étendues sont évaluées du haut en bas (la première condition est contrôlée en premier, puis la deuxième, etc.). Si vous souhaitez retourner une valeur si aucune des conditions ne sont vraies, connectez-vous à **otherwise**.

Pour un exemple de mappage étape par étape, voir Exemple: Retourner une valeur par condition 448.

## 5.6.1 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. La technique illustrée dans cet exemple fonctionne de la même manière non seulement pour XML, mais aussi pour d'autres types de composant, comme CSV ou texte. Dans le cas des bases de données, bien que vous pouvez utiliser un filtre, il est recommandé d'utiliser à la place un composant SQL WHERE/ORDER, pour une meilleure performance (voir Composant SQL WHERE / ORDER 365).

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 468). 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 :

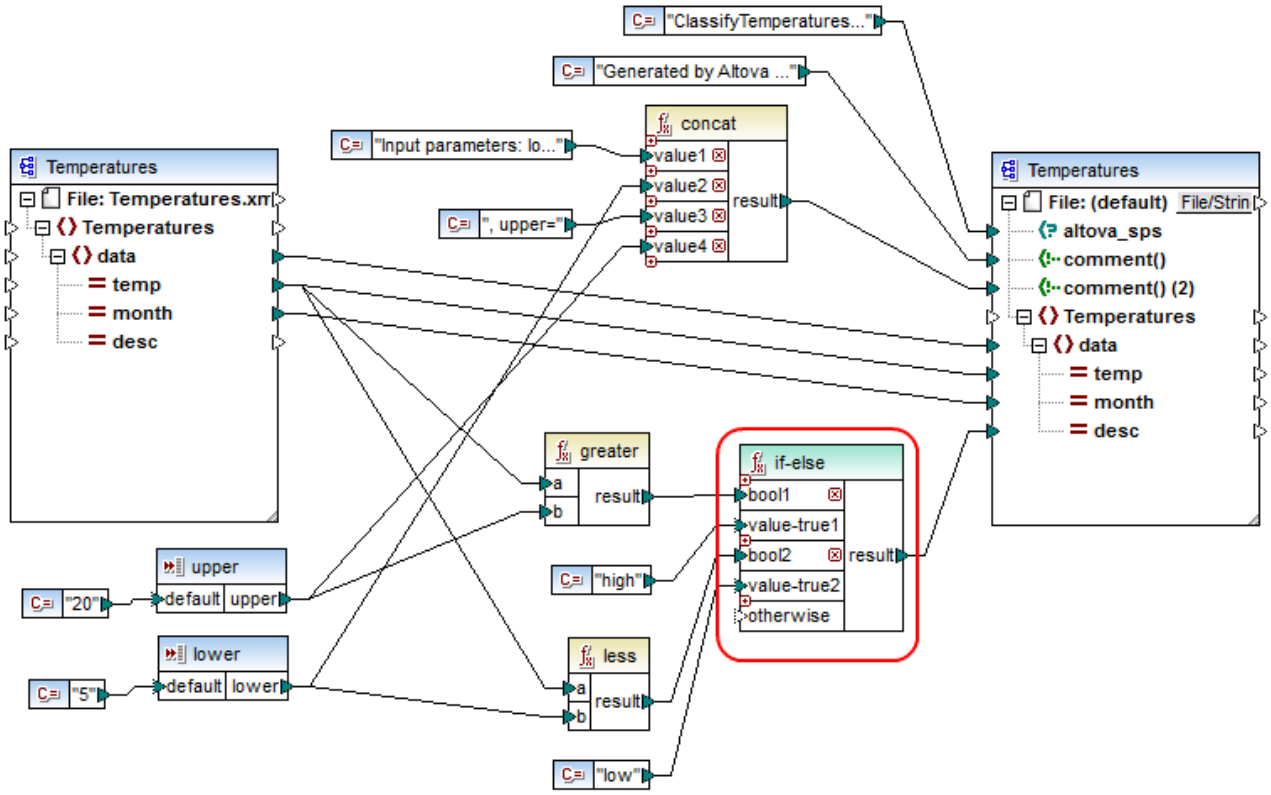
```
...
   <expense-item>
      <type>Meal</type>
      <Date>2003-01-01</Date>
      <expense>122.11</expense>
   </expense-item>
   <expense-item>
      <type>Lodging</type>
      <Date>2003-01-02</Date>
      <expense>299.45</expense>
   </expense-item>
...
```

*Sortie XML après exécution du mappage*

## 5.6.2 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 446).

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 468). 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 398.

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.

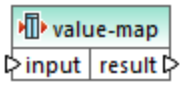
Une fois que vous êtes prêt à consulter le résultat de mappage, cliquer sur l'onglet **Sortie**. Veuillez noter que la sortie XML résultante contient maintenant l'attribut "desc", que la température soit supérieure à 20 ou inférieure à 5.

```
...
   <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*

# 5.7 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 444 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.

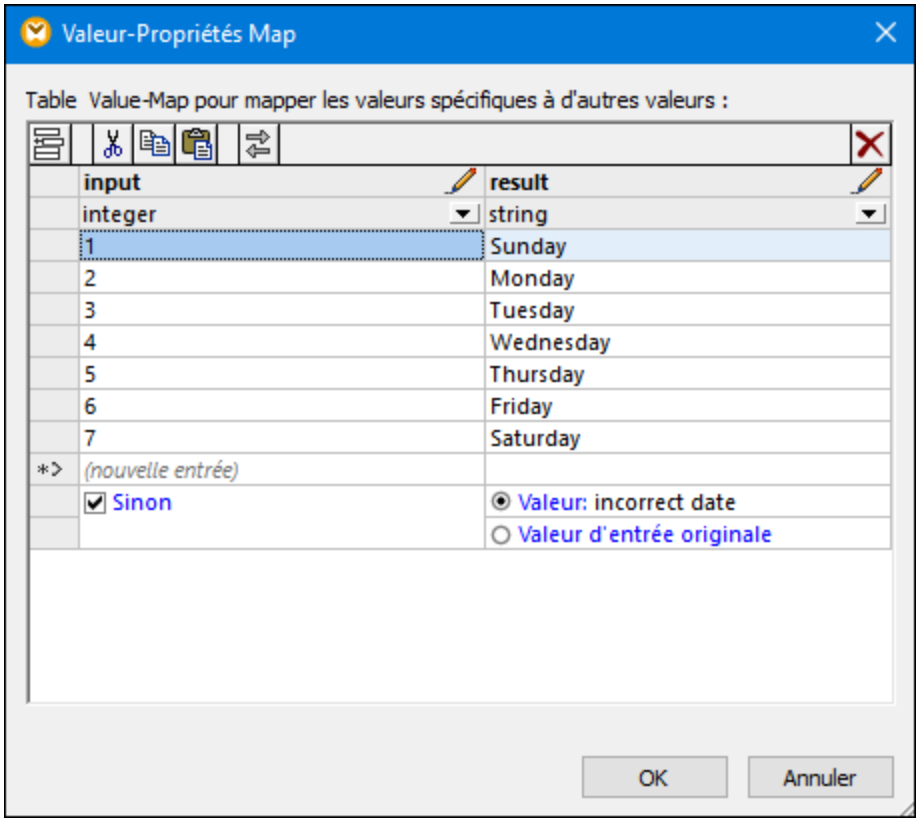
Si la table de consultation est très volumineuse, l'exécution de mappage sera ralentie par la table de consultation. Dans ce cas, il est recommandé d'utiliser un composant de base de données avec SQL-Where 365 à la place. Les bases de données sont de bons candidats pour cela, en raison de leur portabilité. 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`**[625].

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[459].

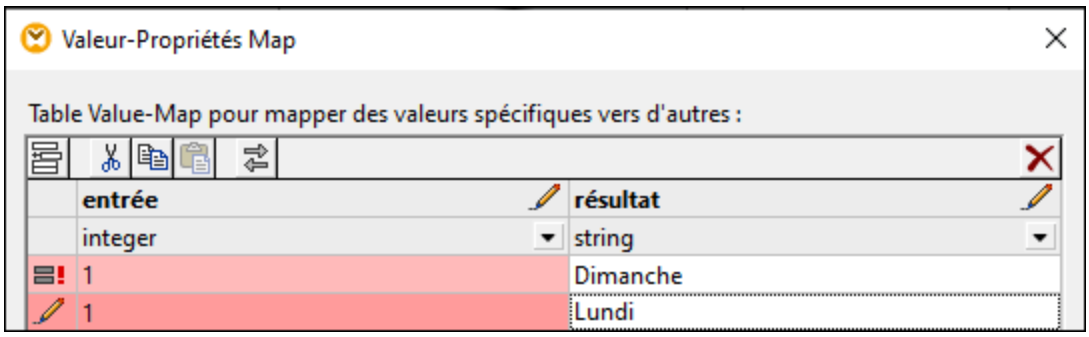
## Remplir un Value-Map

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. Vous pouvez aussi coller des données depuis la grille de la base de données dans le volet de Requête BD 300. 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 456.

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.

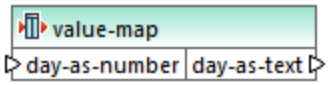
Pour copier ou couper des lignes pour les coller ensuite sur une autre position, sélectionner tout d'abord la ligne, puis cliquer sur la touche **Copier** (ou sur la touche **Couper** , respectivement). Vous pouvez aussi copier ou couper plusieurs lignes qui ne sont pas forcément consécutives. Pour sélectionner 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** .

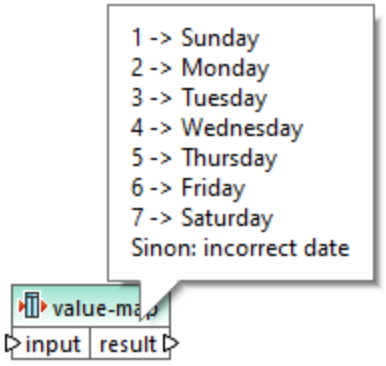
## Renommer des paramètres de Value-Map

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 :

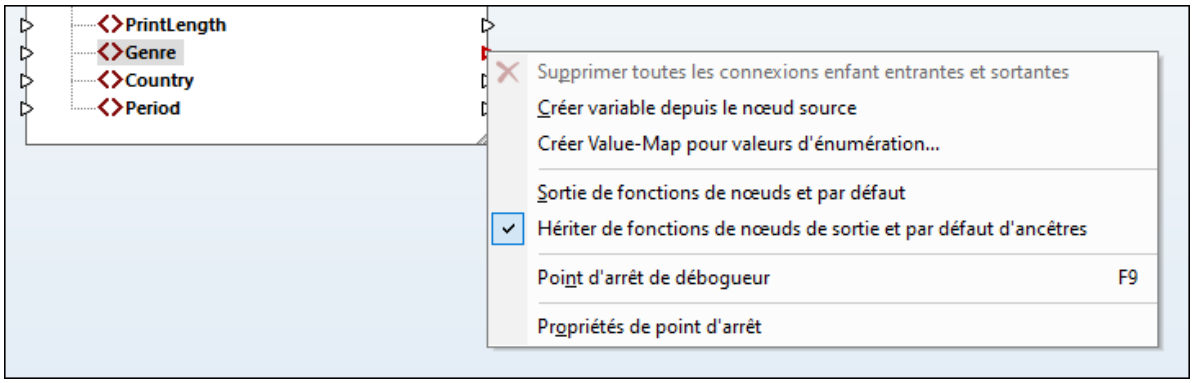



## Créer un Value-Map depuis le type d'énumération

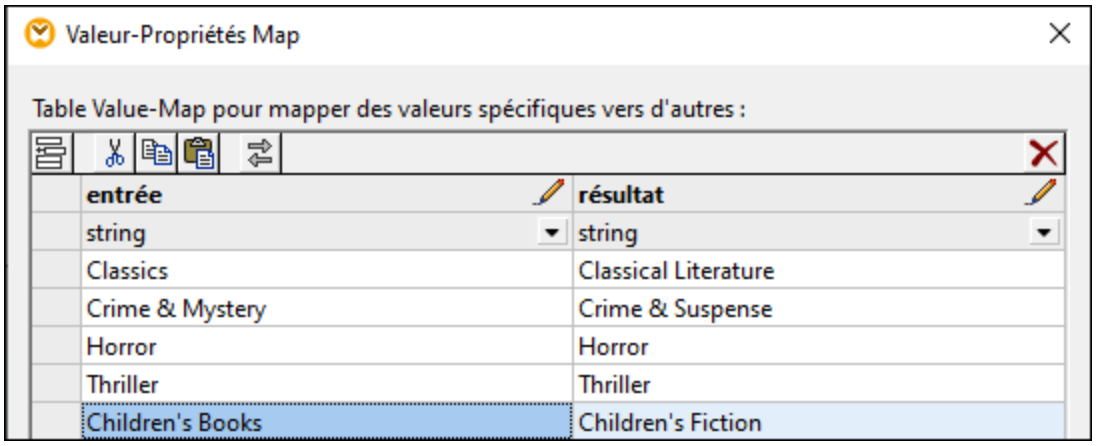
MapForce vous permet de créer un Value-Map depuis de nœuds avec des valeurs d'énumération. Cette fonction est actuellement prise en charge pour les composants XML dont les nœuds ont des facettes d'énumération (*toutes éditions*) et composants EDI dont les nœuds ont des listes de code EDI (*Enterprise 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`.



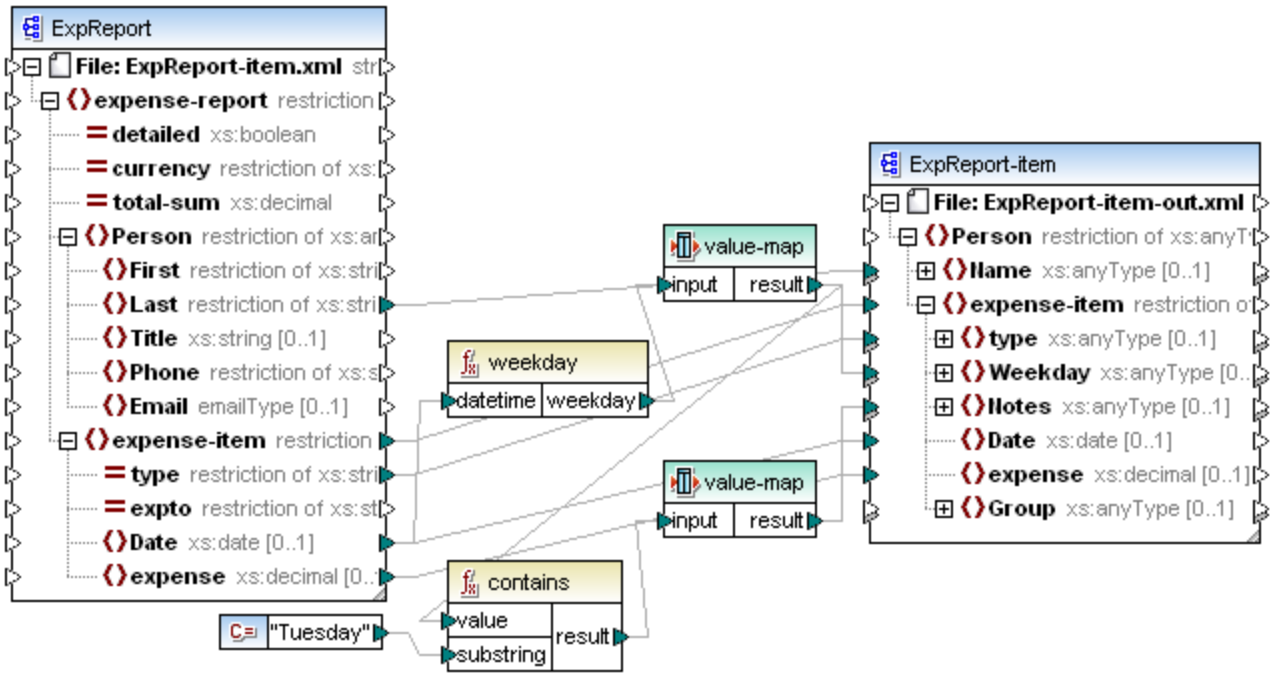

2. Sélectionnez **Créer Value-Map pour Valeurs d'énumération** depuis le menu contextuel.
3. Le dialogue **Propriétés Value-Map** apparaît. Les deux parties `input` et `result` de Value-Map sont pré-remplies avec les mêmes valeurs d'énumération. Pour pouvez revoir et éditer les valeurs tel que requis. 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.

## 5.7.1 Exemple : Remplacer les jours de la semaine

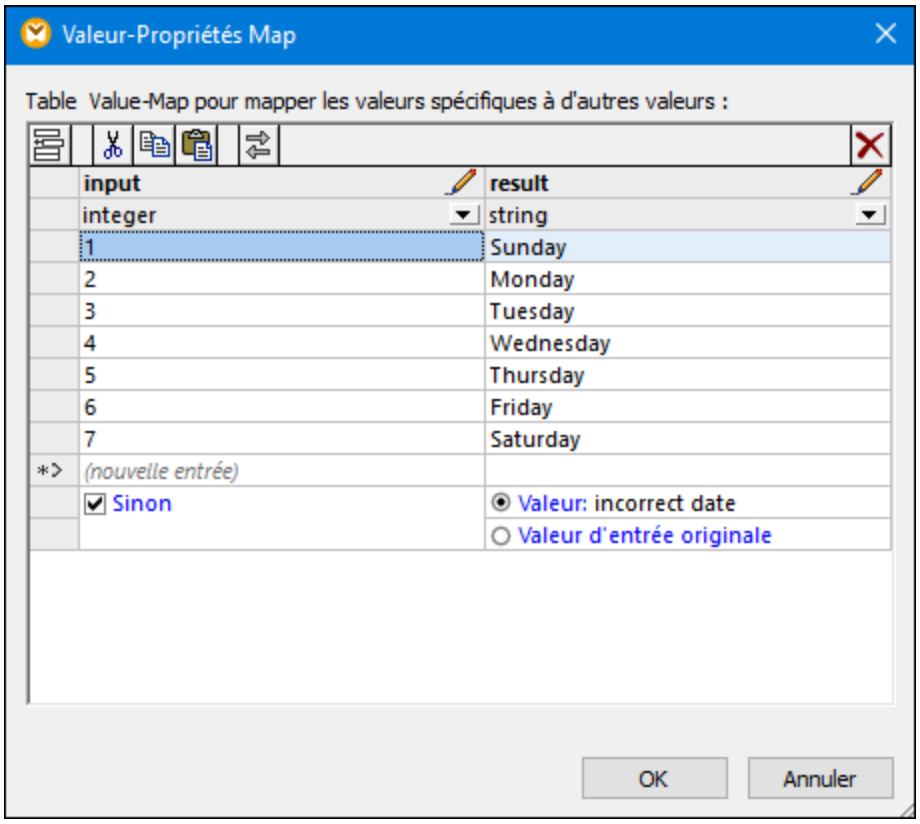
Cet exemple illustre un Value-Map qui remplace des valeurs entières avec les noms du jour de la semaine (1 = Dimanche, 2 = Lundi, etc.). Cet exemple est accompagné par un mappage qui est disponible dans le chemin suivant : **<Documents>\Altova\MapForce2024\MapForceExamples\Tutorial\Expense-valmap.mfd**.



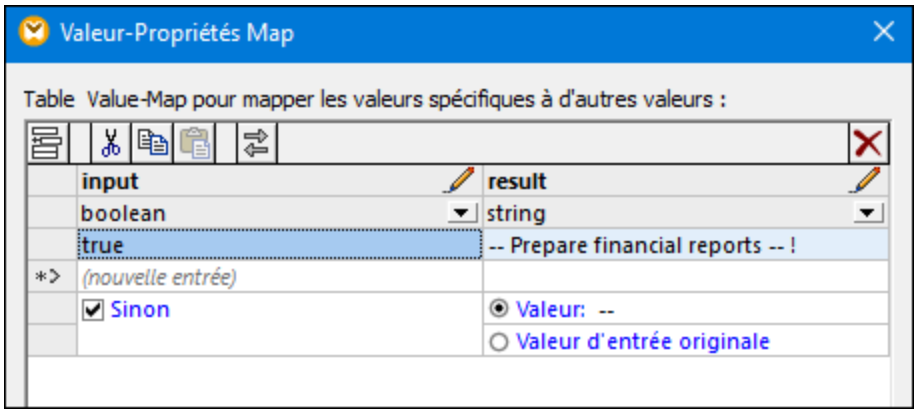

*Expense-valmap.mfd*

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.

## 5.7.2 Exemple : Remplacer des titres de tâche

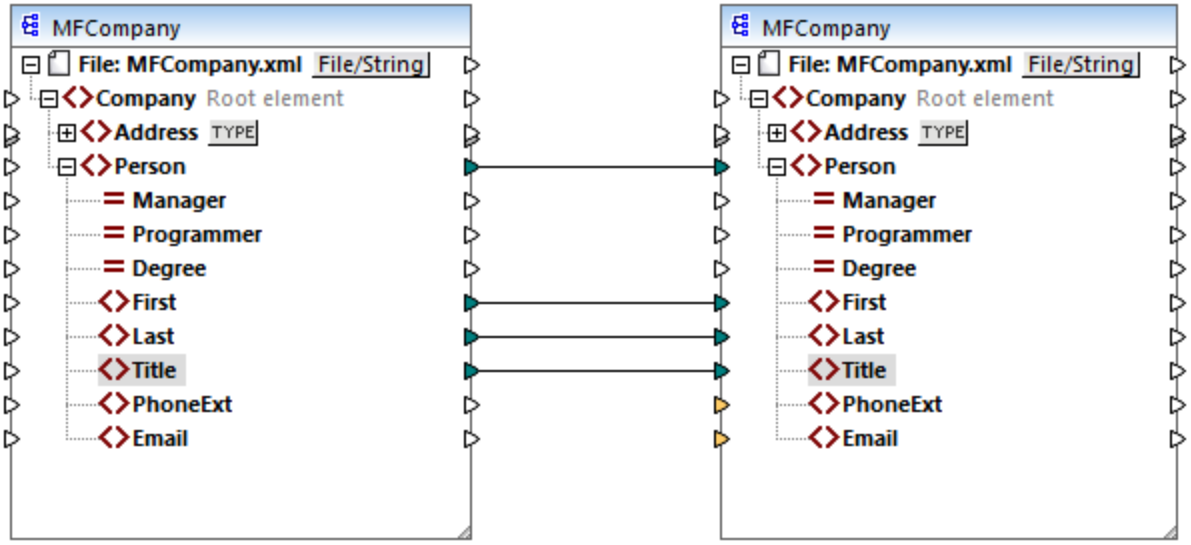
Cet exemple vous montre comment remplacer des valeurs d'éléments spécifiques dans un fichier XML à l'aide de composants Value-Map (c'est à dire, en utilisant une table de consultation prédéfinie).

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>
```

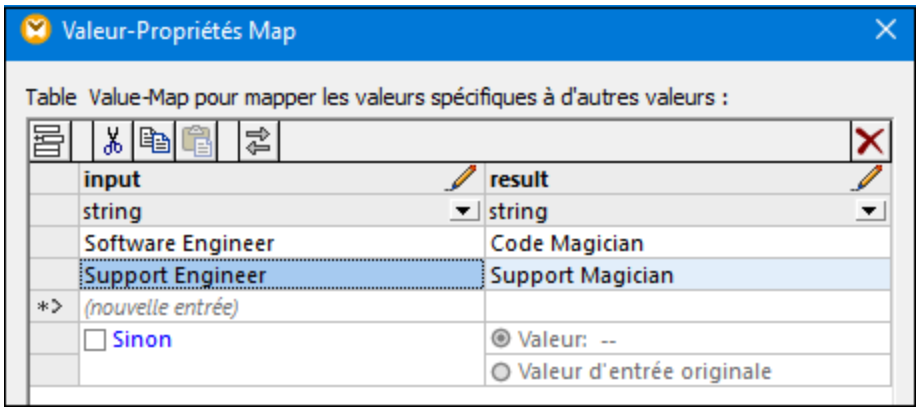
Partons du principe que vous souhaitez remplacer certains des titres de poste dans le fichier XML ci-dessus. En particulier, le titre "Software Engineer" doit être remplacé par "Code Magician". De même, le titre "Support Engineer" doit être remplacé par "Support Magician". Les autres titres de poste 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 :

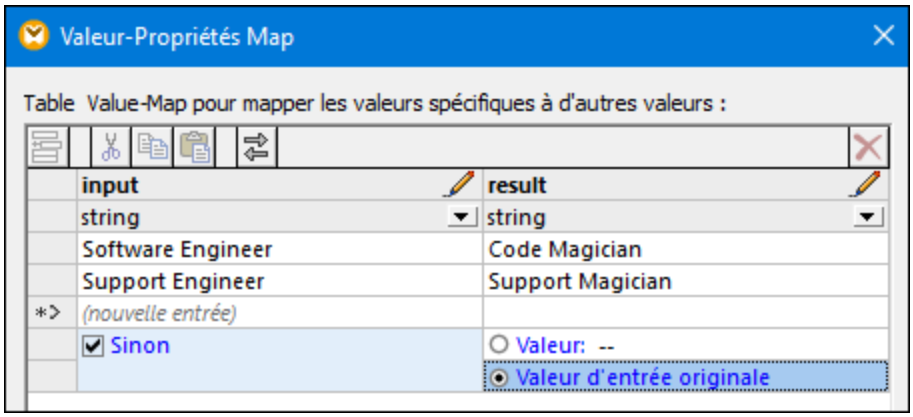



Conformément à la configuration ci-dessus, chaque occurrence de "Software Engineer" sera remplacée par "Code Magician", et chaque occurrence de "Support Engineer" sera remplacée par "Support Magician". Veuillez noter que la condition **Otherwise** n'est pas encore spécifiée. C'est pour cette raison que 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>
```

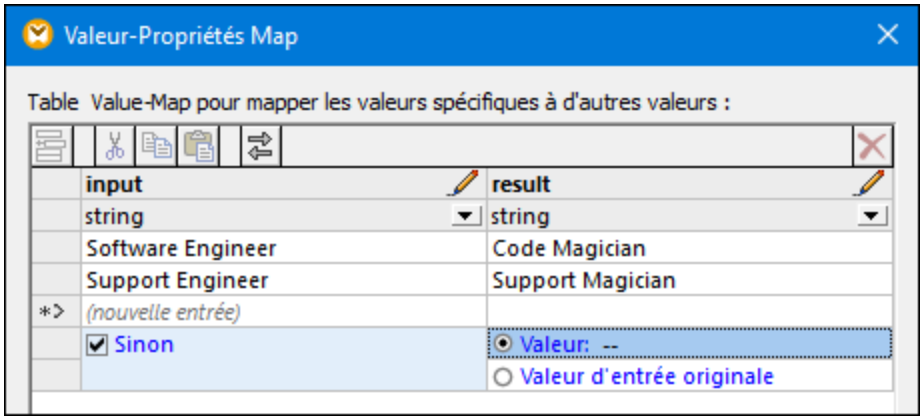
Comme indiqué auparavant, des nœuds vides entraînent des entrées manquantes dans la sortie générée ; c'est pourquoi, dans le fragment XML ci-dessus, seul le titre de Michelle Butler a été remplacé, étant donné que son titre était présent dans la table de consultation. La configuration créée jusqu'à présent 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.

# 5.8 Exceptions

Une exception est un type de composant spécial qui vous permet d'arrêter le processus de mappage et d'afficher une erreur lorsqu'une condition retournée par un filtre se produit. Vous pouvez ajouter une exception lorsque votre mappage inclut un filtre qui vérifie une condition vrai/faux (voir Filtres et Conditions 444). Par exemple, vous pouvez vouloir lancer une exception si la valeur de certains item de mappage est supérieure à un seuil personnalisé.

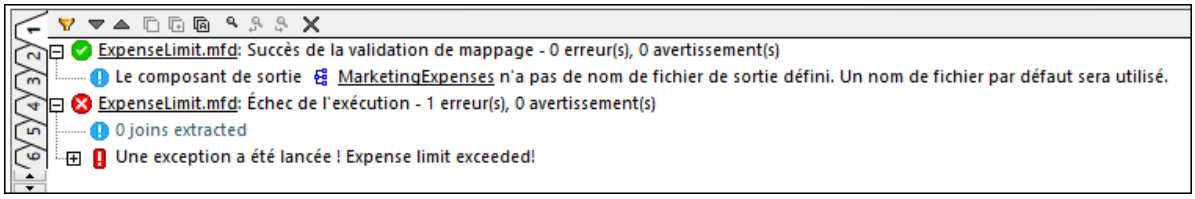
### Pour ajouter une exception dans le mappage :

1. Dans le menu **Insérer**, cliquer sur **Exception**.
2. Cliquer sur la touche de barre d'outils **Insérer Exception** ( ).
3. Connecter l'entrée **throw** de l'exception soit dans une sortie **on-true** ou **on-false** d'un filtre.
4. En option, connecter l'entrée **error-text** de l'exception à un autre composant (généralement, une constante) qui fournit le texte de l'erreur lorsque cette exception est lancée.

**Note :** Les deux sorties **on-true** et **on-false** du filtre doit être connecté. Spécifiquement, une de ces sorties doit être connectée directement à l'exception (sans fonctions ou composants intermédiaires). L'autre sortie doit être connectée au composant de cible, soit directement ou par le biais des composants intermédiaires.

Lorsque le mappage rencontre une exception, vous serez prévenus comme ci-dessous :

- Dans MapForce, la fenêtre Messages affiche une erreur, et le texte d'exception (dans ce cas, "Expense limit exceeded" (La limite de dépense est dépassée)).




  Si la langue de mappage est XSLT 2.0 ou XQuery, une erreur "Execution failed" (Échec d'exécution) apparaît dans la fenêtre Messages, et l'onglet XSLT2 ou XQuery respectif est ouvert. La ligne d'erreur est souligné dans la fenêtre Messages.

- Si vous exécutez le mappage avec MapForce Server, l'erreur "Exception was thrown!" (L'exception a été lancée !) est retournée, suivie par le texte d'exception personnalisé que vous avez défini dans MapForce.

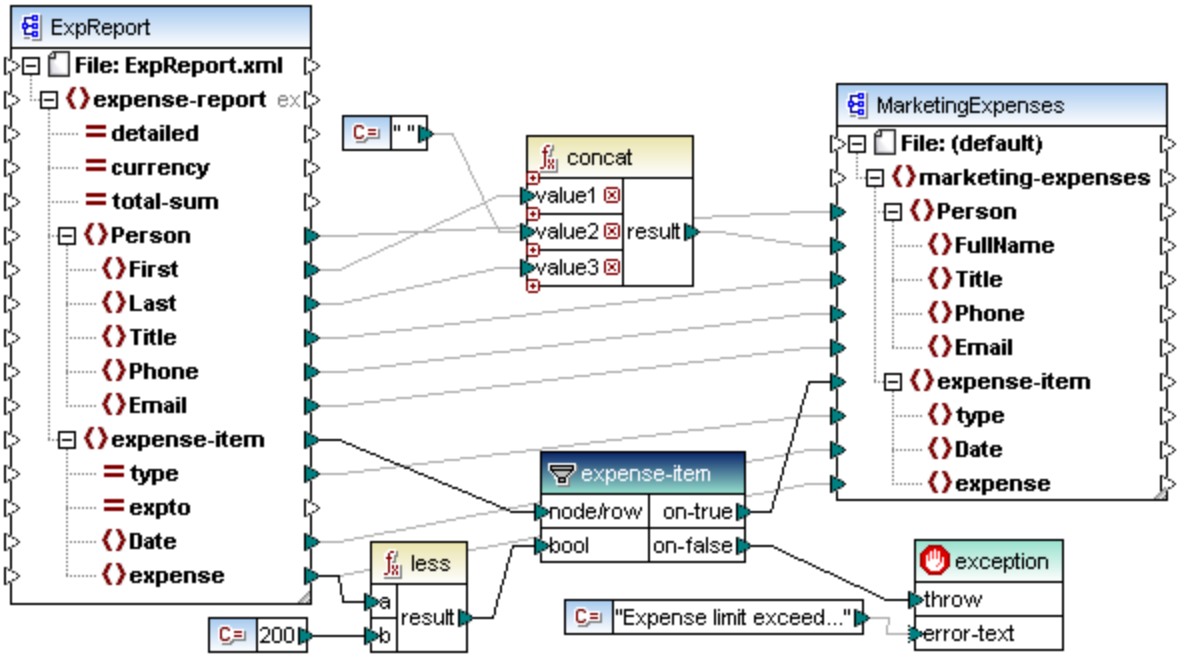
```
c:\codegen\mfx\ExpenseLimit>"C:\Program Files (x86)\Altova\MapForceServer2016\bi
n\MapForceServer.exe" run ExpenseLimit.mfx
Exception was thrown! Expense limit exceeded!
```

- Si vous exécutez le mappage depuis le code C#, C++ ou Java généré, l'erreur "USER EXCEPTION" est retournée, suivie par le texte d'exception personnalisé que vous avez défini dans MapForce.

```
C:\codegen\cs\ExpenseLimit\Mapping\bin\Debug>Mapping.exe
Mapping Application
USER EXCEPTION: Expense limit exceeded!
```

## 5.8.1 Exemple : Exception sur condition "Greater Than" (Supérieur à)

Cet exemple illustre un mappage qui lance une exception lorsqu'une condition "Greater Than" (Supérieur à) se produit. Le mappage échantillon qui accompagne cet exemple peut être consulté sous : **<Documents>\Altova\MapForce2024\MapForceExamples\ExpenseLimit.mfd**.




Ce mappage lance une exception à chaque fois que l'item **expense** dans l'instance XML de source a une valeur supérieure à 200. La valeur "200" est fournie par une constante. La fonction `less` est utilisée pour comparer les deux valeurs. Si la valeur de **expense** est inférieure à 200, alors son parent, l'**expense-item**, est transmis sur le filtre, et aucune exception n'est lancée. Sinon, une exception est lancée, avec le texte personnalisé "Expense limit exceed" (La limite de dépense est dépassée).

Comme affiché ci-dessus, l'exception est identifiée par l'icône et elle consiste en deux items : **throw** et **error-text**. L'item **throw** doit être connecté dans la sortie **on-false** ou **on-true** d'un filtre. L'**error-text** est connecté à une constante qui fournit le texte personnalisé de l'exception.

Surtout, les deux sorties du filtre sont connectées ; sinon, l'exception ne sera pas lancé. Dans cet exemple particulier, la sortie **on-false** est connectée à l'exception, alors que la sortie **on-true** est connectée au composant cible.

## 5.8.2 Exemple : Exception lorsque le nœud n'existe pas

Cet exemple illustrate comment lancer une exception lorsqu'un nœud dans le schéma XML de source n'existe pas. Pour des raisons de simplicité, cet exemple utilise le même schéma XML aussi bien en tant que composant source et cible.

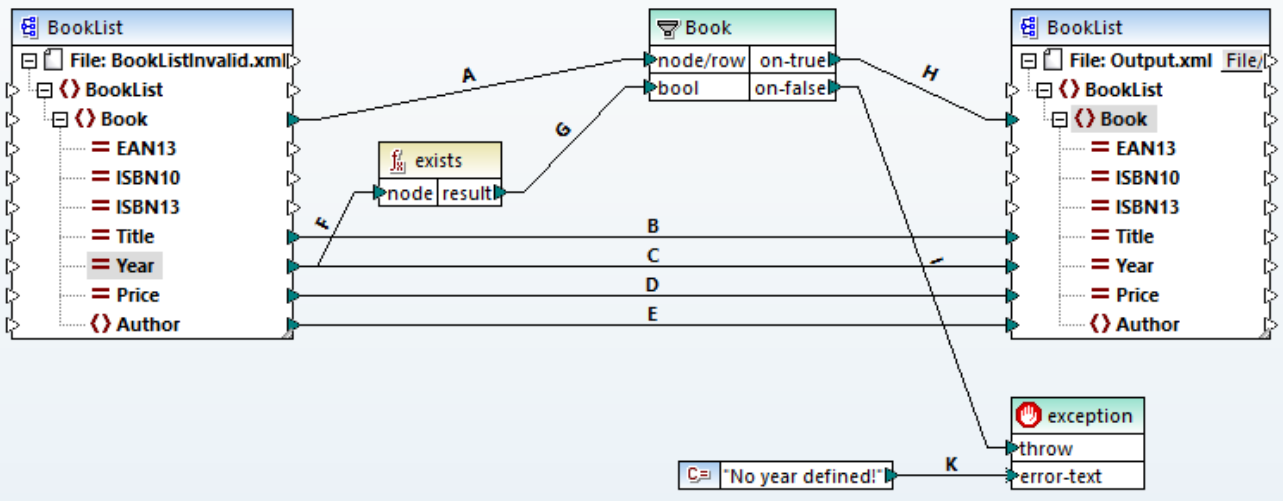
### Pour ajouter le schéma de source dans le mappage :

1. Dans le menu **Insérer**, cliquez sur **Schéma XML/Fichier**, et cherchez **<Documents>\Altova\MapForce2024\MapForceExamples\ BookList.xsd**.
2. Lorsque vous serez invité à fournir un fichier d'instance, cliquez sur **Sauter**.
3. Lorsque vous serez invité à sélectionner un élément de racine de schéma, sélectionnez **BookList** en tant qu'élément de racine.

Pour ajouter le schéma de cible, suivre les mêmes étapes. Ensuite, en utilisant les commandes correspondantes depuis le menu **Insérer** (ou les touches de barre d'outils correspondantes), ajoutez les éléments suivants :

- Un composant **Filtre : Nœuds/Lignes** (voir aussi Filtres et conditions 444)
- Une constante avec le texte "No year defined!" (Aucune année définie !)
- Une exception

Enfin, tirer la fonction `exists` depuis la fenêtre **Bibliothèques** dans la zone de mappage, et établir les connexions comme illustré ci-dessous.




Conformément au schéma XML, tous les attributs de l'élément **Book** sont optionnels, sauf le titre du livre. C'est pourquoi, l'attribut "Year" peut ou pet ne pas exister dans une instance XML valide. L'objectif du mappage est de traiter avec succès une instance XML où l'attribut "Year" existe pour chaque livre. Sinon, le mappage doit lancer une exception.

### Pour tester l'exécution réussie du mappage :

1. Double-cliquer sur l'en-tête du composant de source et, à côté de **Fichier XML d'entrée**, chercher le fichier suivant : **<Documents>\Altova\MapForce2024\MapForceExamples\ BookList.xml**.
2. Cliquer sur la touche **Sortie** pour exécuter le mappage.

**Pour tester l'exception :**

1. Créer, dans le même répertoire, une copie du fichier **BookList.xml** appelée **BookListInvalid.xml**.
2. La modifier de manière à supprimer l'attribut "Year" d'un élément Book.
3. Double-cliquer sur l'en-tête du composant de source, et, à côté du **fichier XML d'entrée**, chercher le fichier **BookListInvalid.xml**.
4. Cliquer sur la touche **Sortie** pour exécuter le mappage.

Examinons à présent de plus près le fonctionnement d'un mappage.

La connexion **A** veille à ce qu'un livre soit créé dans l'instance cible pour chaque livre dans l'instance de source. Les connexions **B, C, D, E** garantissent que "Title", "Year", "Price" et "Author" sont copiés depuis la source vers la cible, pour chaque livre.

La connexion **F** déclenche la fonction **exists** pour contrôler l'existence de l'attribut "Year". La connexion **G** fait passer le résultat de fonction (`true` ou `false`) dans le filtre. Si le résultat est `true`, l'attribut "Year" existe, et le livre passe dans le filtre, et ensuite dans la cible par le biais de la connexion **H**.

> Veuillez noter que le filtre n'a pas été connecté directement à la sortie **Year** du composant de source. Si nous l'avions connecté de cette manière, le filtre filtrerait **Year** du fait de sa propre existence, ce qui n'est pas significatif, et l'exception ne serait jamais lancée.

La connexion **I** est là parce que l'exception doit être connectée soit à une sortie **on-false** ou **on-true** d'un filtre, conformément aux règles. Enfin, la connexion **K** fait passer le texte d'erreur personnalisé de la constante au composant d'exception.

# 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 547.
- **Fonctions et défauts de nœuds** — il s'agit de fonctions plus spécialisées qui vous permettent de créer et d'appliquer une logique de traitement personnalisée à un ou plusieurs nœuds descendants dans un composant de mappage. Elles permettent de traiter soit *avant* d'atteindre un nœud d'une structure de mappage, ou *immédiatement après* qu'elle quitte un nœud. Pour plus de détails, voir Fonctions par défaut et de nœud 476.
- **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 491.
- **Fonctions personnalisées** — il s'agit des fonctions que vous pouvez importer depuis des sources externes comme des bibliothèques XSLT, des modules de bibliothèque XQuery, des fichiers .class Java, des fichiers .NET .dll, 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 508, Importer des fonctions XQuery 1.0 personnalisées 515, et Importer des bibliothèques Java et .NET personnalisées 520.

**Note :** Vous pouvez importer des bibliothèques externes de fonctions soit directement (aucune configuration nécessaire) soit en configurant un MFF (MapForce Function File) reconnu par MapForce. Si vous utilisez la deuxième possibilité, vous pouvez aussi importer des bibliothèques C++, en plus des classes Java et des .NET assemblies. Veuillez noter que les bibliothèques importées via des fichiers .mff doivent remplir les conditions préalables mentionnées dans Référencer les bibliothèques Java, C# et C++ manuellement 528.

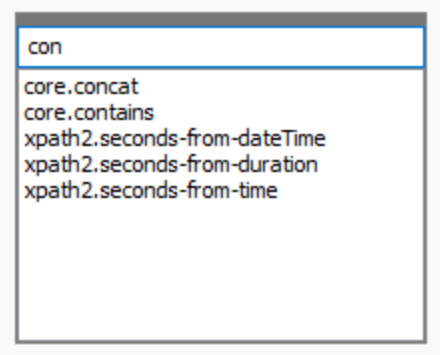
# 6.1 Notions fondamentales de base

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 22.

## 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 547. 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.

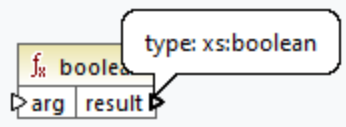
## Chercher une fonction

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 ✖.

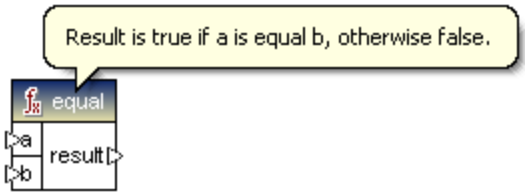
Pour trouver toutes les occurrences d'une fonction dans le mappage actif actuellement, cliquez avec la touche de droite sur le nom de la fonction dans la fenêtre **Bibliothèques** et choisissez **Trouver tous les appels** depuis le menu contextuel. Les résultats de recherche sont affichés dans la fenêtre **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.

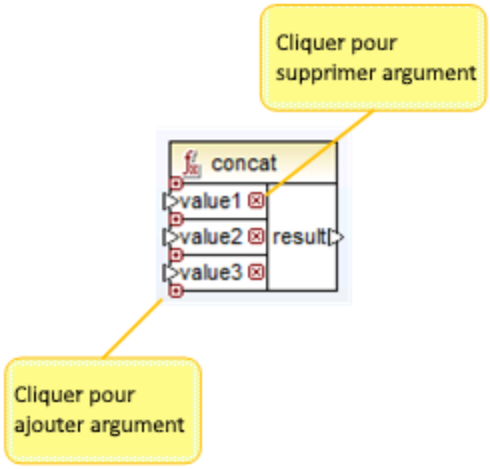



Pour afficher la description d'une fonction, déplacez le curseur au-dessus de l'en-tête de la fonction (*voir la capture d'écran ci-dessous*). Veuillez vous assurer que la touche de la barre d'outils (**Afficher astuces** est activée.




## 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** 629. 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.



# 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 492.
- 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 2.0 508.
- Les fichiers XQuery 1.0 personnalisés qui contiennent des fonctions. Cela réfère aux fonctions XQuery écrites en dehors de MapForce qui qualifient pour l'importation dans MapForce comme décrit dans Importer des fonctions personnalisées XQuery 1.0 515.
- Les fichiers Java .class et bibliothèques .NET .dll qui qualifient pour l'importation dans MapForce tel que décrit dans Importer des bibliothèques Java et .NET personnalisées 520.

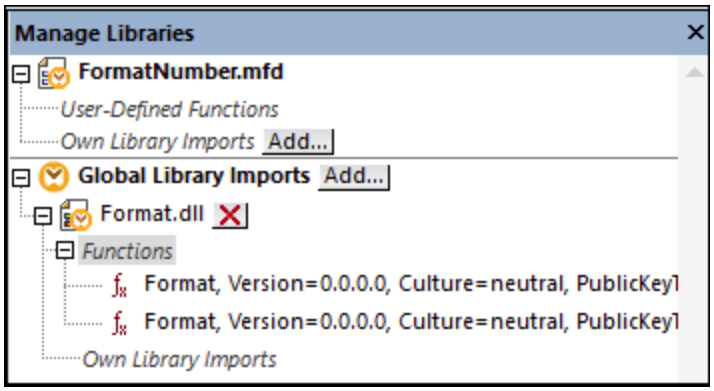
**Note :** Vous pouvez importer des bibliothèques externes de fonctions soit directement (aucune configuration nécessaire) soit en configurant un MFF (MapForce Function File) reconnu par MapForce. Si vous utilisez la deuxième possibilité, vous pouvez aussi importer des bibliothèques C++, en plus des classes Java et des .NET assemblies. Veuillez noter que les bibliothèques importées via des fichiers .mff doivent remplir les conditions préalables mentionnées dans Référencer les bibliothèques Java, C# et C++ manuellement 528.

## 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 |

| Commande | Description | Applicable pour |
|---|---|---|
| **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 |
| **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 26. 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.

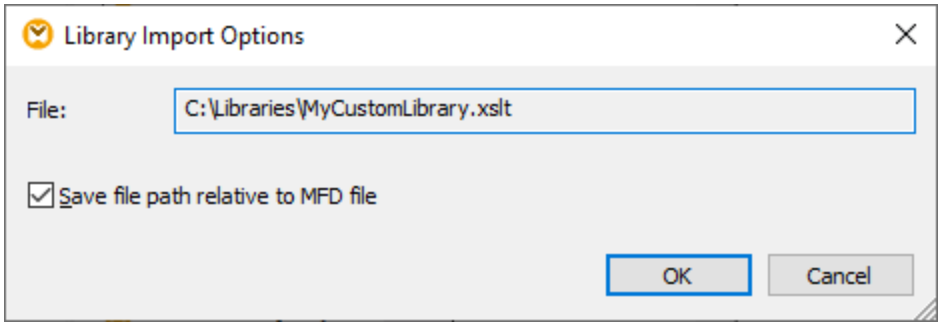
## 6.2.2 Chemins de bibliothèque relatifs

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 473.

> 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 [28] 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 [47].

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 [49].

# 6.3 Valeurs par défaut et fonctions de nœud

Les fonctions par défaut et les fonctions de nœud sont particulièrement utiles lorsque vous souhaitez appliquer la même logique de traitement à plusieurs items descendants dans une structure, par exemple : Normalement, pour ce faire, vous devriez copier-coller la même fonction plusieurs fois sur le mappage. Néanmoins, cela encombrerait le mappage et le rendrait plus compliqué à lire. Les fonctions par défaut et fonctions de nœud peuvent être appliqués à un seul item ou à plusieurs items à la fois. Les défauts remplacent les séquences vides. Si la connexion transporte une valeur, le défaut sera ignoré.

Les défauts et fonctions de nœud sont idéales pour presque la plupart des composants qui ont une arborescence avec des nœuds (par ex., XML, EDI, composants Join, variables).

Les défauts et fonctions de nœud sont compatibles uniquement avec le langage de transformation Built-In. L'exécution de tels mappages provenant du code de programme C#, C++, Java, ou avec les transformations XSLT/XQuery générées n'est pas prise en charge. Du côté du serveur, vous pouvez exécuter ces mappages avec MapForce Server Advanced Edition.

## Les avantages de fonctions de nœud et les défauts

Créer des fonctions de nœud et des défauts signifient définir une *règle*. Les règles ont les caractéristiques importantes suivantes qui les rendent particulièrement flexibles et faciles à utiliser :

- *Héritage.* Lorsque vous définissez une règle sur un item qui a des descendants, la règle sera héritée par des descendants par défaut sauf si vous avez choisi de désactiver cette option. Si l'item dans lequel vous définissez la fonction est doté de plusieurs niveaux d'items enfants imbriqués sous lui, vous pouvez choisir d'appliquer la règle uniquement aux items d'enfant directs, ou à tous les items descendants.
- *Filtrer* MapForce applique les règles de manière conditionnelle, basées sur le type de données de chaque item. Cela permet par exemple, d'appliquer certaines valeurs par défaut ou une fonction pour tous les items de type `string`, et un défaut différent ou une fonction pour tous les items de type `décimal` Pour les détails, voir le *Scénario 2* dans Scénarios Use-Case 481. Vous pouvez également définir des options de filtrage plus avancées : Par exemple, vous pouvez spécifier un type de données à laquelle votre fonction doit correspondre (ceci pourrait être une catégorie de types de données tels que `numérique`), puis filtrer les nœuds de ce type de données basé sur le nom de nœud ou le type de nœud (par ex., `integer`). Pour plus de détails, voir le *Scénario 5* 485.

Par exemple, vous pouvez demander à MapForce de faire la chose suivante :

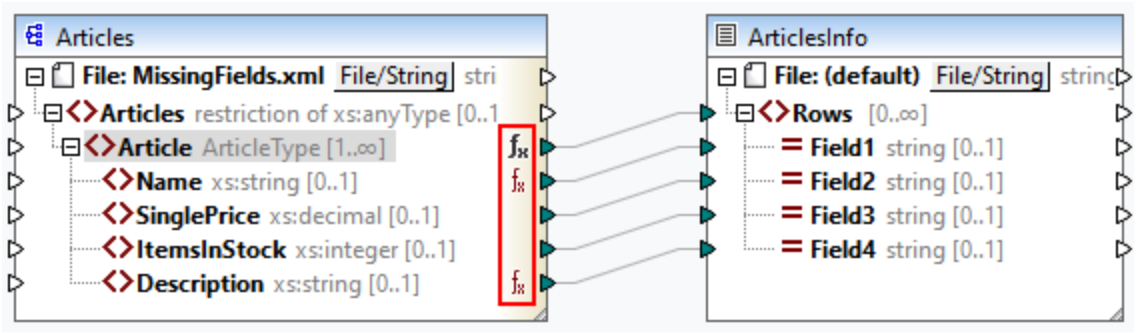
- Chaque fois qu'une valeur vide ou nulle survient, la remplacer avec une autre valeur et le faire récursivement pour tous les items descendants.
- Chaque fois qu'une valeur spécifique survient, la remplacer avec une autre valeur (ou avec un string vide), et le faire récursivement pour tous les items descendants.
- Remplacer toutes les valeurs null de la base de données avec des strings vides et du texte personnalisé.
- Ajouter un préfixe ou un suffixe à toutes les valeurs qui sont rédigées dans un fichier cible ou une base de données

## Côté sortie vs. entrée

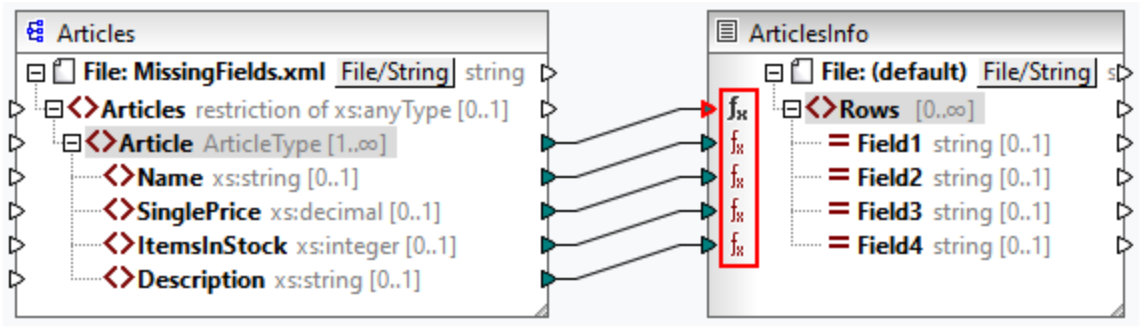
Vous pouvez définir des fonctions de nœud et des défauts du côté entrée, du côté sortie, ou des deux côtés d'un composant, dépendant de vos besoins. Dans MapForce, un mappage fonctionne de la manière suivante :

(i) d'abord, il lit les données du composant source (par ex., un fichier XML), (ii) puis, en option, il les traite d'une manière (par ex., en utilisant une fonction), et (iii) finalement, il écrit les données dans un composant cible (par ex., une base de données). En tenant compte de ce principe de base, vous pouvez définir les fonctions de nœud et les défauts à des niveaux différents :

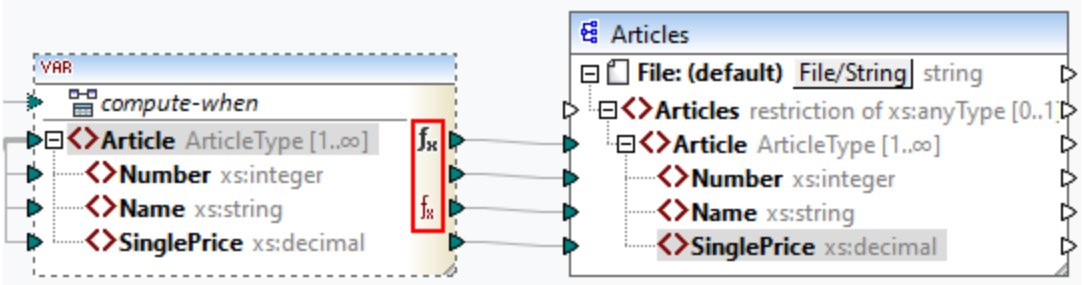
- Immédiatement *après* la lecture des données depuis la source, mais avant qu'elles ne soient traitées par votre mappage, ce qui signifie que la fonction/le défaut a été défini du côté de la sortie du composant source (*voir l'exemple ci-dessous*).




- Immédiatement *avant* l'écriture des données au composant cible (et une fois tous les traitements intermédiaires terminés), ce qui signifie que la . fonction/le défaut a été défini du côté de l'entrée du composant cible (*voir l'exemple ci-dessous*).

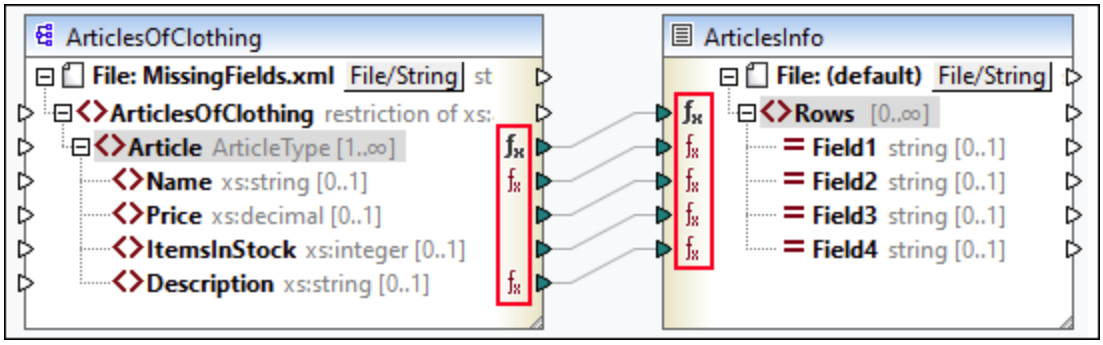



- À un stage intermédiaire du processus de mappage. Par exemple, si le mappage contient une variable intermédiaire de type complexe (par ex., une structure XML), vous pourriez réduire toutes les valeurs avant qu'elles soient fournies à la structure XML, ou juste après qu'elles soient retournées par la structure XML (*voir l'exemple ci-dessous*).




- Du côté sortie d'un composant de source et du côté entrée du composant cible. Dans l'exemple ci-dessous, un défaut a été défini pour tous les nœuds de type `string` dans le composant

`ArticlesOfClothing`. Dans le composant `ArticlesInfo`, nous avons défini une fonction de nœud qui transformera toutes les valeurs de nœud e string en caractères majuscules.




### Dans cette section

Cette section explique comment configurer une règle, décrit les scénarios de vie réelle dans lesquels les défauts et fonctions de nœud peuvent être utiles, et montre comment ajouter des métadonnées de nœud aux fonctions de nœud. La section est organisée en rubriques suivantes :

- Configuration de la règle 478
- Scénarios Use-Case 481
- Métadonnées de nœud dans les fonctions de nœud 487

## 6.3.1 Configuration de la règle

Vous pouvez créer des fonctions de nœud et des défauts pour presque chaque item dans le mappage. Créer des fonctions de nœud et des défauts signifient définir une *règle*. Afin de créer une règle, sélectionnez un item (nœud ou champ) dans lequel vous souhaitez définir une règle. Il peut s'agir d'un seul nœud ou d'un nœud avec des nœuds enfant. Si vous choisissez de créer une règle pour un nœud avec enfants, la règle s'appliquera à tous les enfants sauf si vous désactivez explicitement cette option.

### Aspects importants de défauts et fonctions de nœud

Les défauts et fonctions de nœud ont les aspects importants suivants :

- Vous pouvez créer des défauts ou des fonctions de nœud soit du côté d'entrée d'un composant cible, soit du côté sortie d'un composant source. Afin d'établir quel côté est celui qui convient à vos besoins, voir Choisir le côté d'entrée ou de sortie 476.

- Lorsque plusieurs règles existent pour un seul et même item, MapForce appliquera une règle à un item qui est plus proche de cet item. Pour savoir comment écraser les règles, voir le *Scénario 4* dans Scénarios Use-Case 484.

- Les défauts et les fonctions de nœud peuvent être créés si le type de connexion entre la source et la cible est Orienté vers la source ou Orienté vers la cible. Toutefois, les connexions copier-tout ne sont pas prises en charge. En particulier, les fonctions de nœud et les défauts ne sont pas appliquées aux descendants des connexions copier-tout. Le nœud parent avec une connexion copier-tout peut avoir des fonctions de nœud et des défauts, mais uniquement s'il a une valeur simple, par exemple, un

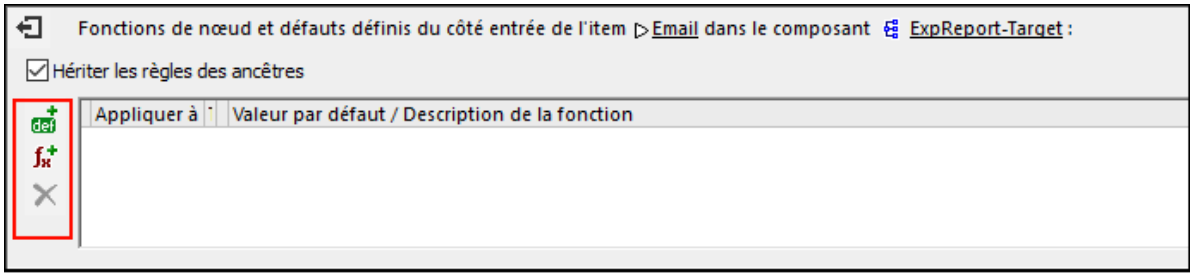
élément XML avec un contenu de type simple et des attributs. Pour plus d'informations sur les types de connexion, voir Types de connexion 54.

- Créer des fonctions par défaut et fonctions de nœud n'est pas prise en charge pour le nœud `File`.
- À l'intérieur d'une fonction de nœud, seuls certains composants importants dans ce contexte sont pris en charge (par ex., des fonctions built-in, des variables, des conditions if-else). Les structures complexes telles que XML, JSON, EDI ou des bases de données ne sont pas prises en charge. Ajouter des fonctions définies par l'utilisateur inline et des composants join à une fonction de nœud n'est pas pris en charge.
- Une fonction de nœud peut avoir un paramètre d'entrée maximum, ou aucun paramètre. Le paramètre d'entrée est toujours appelé `raw_value`. Ne jamais supprimer le paramètre d'entrée, même si vous n'avez pas besoin d'entrée pour votre fonction ; sinon, des erreurs de validation apparaîtront lorsque vous exécutez le mappage. La même chose s'applique à la sortie de la fonction. Si vous devez rétablir un composant d'entrée supprimé par accident, exécutez la commande de menu **Fonction | Insérer entrée**.
- MapForce vous permet de fournir des métadonnées de nœud (par ex., un nom de nœud, une annotation) à une fonction de nœud. Pour les détails, voir Métadonnées de nœud dans les fonctions de nœud 487.
- La boîte de dialogue **Filtrer les fonctions de nœud et Défauts** vous permet de choisir d'une large gamme de types de données, dont quelques unes représentent des catégories de types. Ceci signifie qu'ils correspondront à une plus grande sélection de types. Par exemple, le type `string` correspond à d'autres types de données variés depuis le `string`, tel que `normalizedString`, `jeton`, `NCName`, `NMTOKEN`, `IDREF`, `ENTITY` et d'autres. De même, le type `decimal` correspondra aux types dérivés `integer`, `long`, `short` et autres. La hiérarchie des types est spécifiée dans la Recommandation XML Schema W3C.

## Créer une règle

Pour créer une règle, suivez les étapes suivantes :

1. Cliquez avec la touche de droite sur le nœud d'intérêt et sélectionnez **Fonctions de nœud et défauts | Fonctions de nœud de sortie et défauts** depuis le menu contextuel. Qu'il s'agisse de fonctions de nœud et défauts *Entrée* ou *Sortie* dépend du côté du composant sur lequel vous devez créer une fonction de nœud ou un défaut. En alternative, cliquez avec la touche de droite sur un connecteur et sélectionnez une commande de fonction de nœud pour ce côté. Le volet **Mappage** affichera la fenêtre de fonctions de nœud, dans laquelle vous pouvez définir des défauts et fonctions de nœud (*rectangle rouge dans la capture d'écran ci-dessous*).



Si un nœud parent a des règles définies et que vous voulez transférer ces règles du parent à un enfant, sélectionnez la case à cocher *Hériter les règles des ancêtres* (*voir la capture d'écran ci-dessus*). Pour plus d'informations sur l'héritage, voir Scénarios Use-Case 481.

2. La prochaine étape est de définir si vous voulez ajouter un défaut ( ) ou une fonction ( ). Dès que vous cliquez sur l'option pertinente, une nouvelle règle sera créée (une ligne dans la grille de la fenêtre de fonction de nœud). Pour des informations sur la configuration des règles, voir *Configuration de la règle* ci-dessous. Si vous définissez une fonction, la zone de mappage changera pour afficher l'entrée de la fonction et les paramètres de sortie.

*Éditer/supprimer les règles*

Pour voir, modifier ou supprimer une règle, cliquez sur l'icône (noire ou rouge) à côté d'un nœud d'intérêt. La fenêtre des fonctions de nœud affichera toutes les règles définies pour ce nœud. Si vous avez ajouté une fonction de nœud, vous verrez le bouton Edit dans la grille, qui vous permet de modifier la mise en œuvre de la fonction. Si la touche Edit n'est pas présente, alors la fonction est probablement définie sur un ancêtre. Dans ce cas, cliquez sur l'icône à côté de l'item pour lequel la règle a été définie.

Pour supprimer une règle, sélectionnez-la depuis la grille dans la fenêtre de fonction de nœud, puis cliquez sur le bouton .

## Configurer une règle

Dès que vous sélectionnez **Fonctions de nœud et Défauts | Entrée/Sortie Fonctions de nœud et Défauts** depuis le menu contextuel (*voir les instructions ci-dessus*), vous verrez la fenêtre des fonctions de nœud dans laquelle vous pouvez configurer des fonctions de nœud et des défauts. Les paramètres disponibles sont décrits ci-dessous.

- *Appliquer à :* Une règle peut être appliquée à l'item actuel, ses items enfant directs ou à tous les items descendants. Si l'item que vous avez sélectionné n'a pas de descendant, seule l'option *Item actuel* sera disponible.

- *Type de données :* Cliquez sur la touche ... et choisissez un type de données depuis le dialogue. La règle ne s'appliquera qu'aux items qui ont ce type de données (ou un type de données dérivé). Si l'item que vous avez sélectionné n'a pas de descendants, alors le type de données de l'item est le seul choix possible.

- *Valeur par défaut/Description de la fonction :* Si vous définissez un défaut, saisissez ici la valeur par défaut que vous souhaitez définir pour l'item sélectionné (et tous les descendants, le cas échéant). Pour définir un string vide comme défaut, laissez ce champ vide. Si vous définissez une fonction, ce champ est réservé uniquement à des informations : Il affiche un résumé de la fonction.

Pour quitter la fenêtre de fonction de nœud, cliquez sur le bouton dans le coin supérieur gauche de la fenêtre et appuyez sur **Escape**.

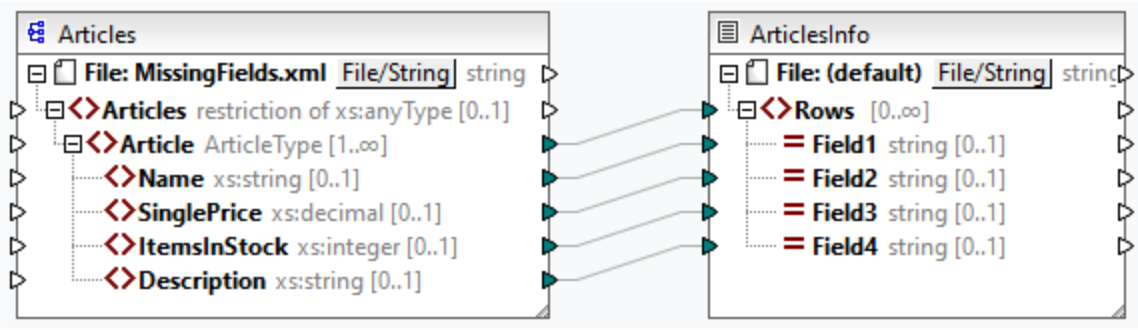
## Indice visuel

MapForce affiche différents indices visuels pour aider à comprendre quelles règles sont définies et si elles s'appliquent à un nœud spécifique (*voir la table ci-dessous*).

| Icône | Description |
|---|---|
| | Cet icône indique qu'une règle est définie pour son item et peut affecter tous ses descendants. Cliquez sur l'icône pour modifier ou supprimer la règle. |
| | Cet icône indique que l'item hérite la règle définie au niveau d'ancêtre. |
| | Cet icône indique que la règle est définie pour et s'applique à cet élément. Cette icône apparaît généralement lorsqu'une fonction ou un défaut est défini pour un seul nœud. |
| | Cette icône indique que même si une règle s'applique à cet item, elle est délibérément bloquée. Cette icône est affichée uniquement si l'héritage est bloqué et aucune autre règle n'est définie pour ce nœud. Si une règle provenant d'un ancêtre s'applique, l'icône a une priorité. |
| | Cette icône indique qu'une règle définie pour cet item est inactive. Par exemple, cette icône peut apparaître pour des items enfants qui ne sont pas encore connectés. |
| | Cette icône indique que la règle est actuellement inactive, car aucune correspondance et nœuds connectés n'ont été identifiés. Le terme *matching* signifie la chose suivante : Avant d'appliquer une fonction ou un défaut, MapForce recherche le type de données que vous avez défini ; si vous avez également spécifié un filtre qui doit correspondre à un nom de nœud spécifique, par exemple, MapForce recherchera également le nom de nœud. |

## 6.3.2 Scénarios Use-Case

Cette rubrique décrit les différents scénarios dans lesquels les défauts et fonctions de nœud peuvent être utiles. Pour tester ces scénarios, vous aurez besoin du mappage d'échantillon suivant : **<Documents>\Altova\MapForce2024\MapForceExamples\Tutorial\MissingFields.mfd**. Notre point de départ est le mappage illustré ci-dessous.




Certains des nœuds dans le fichier source sont vides. À cette étape, le fichier de sortie a l'aspect suivant :

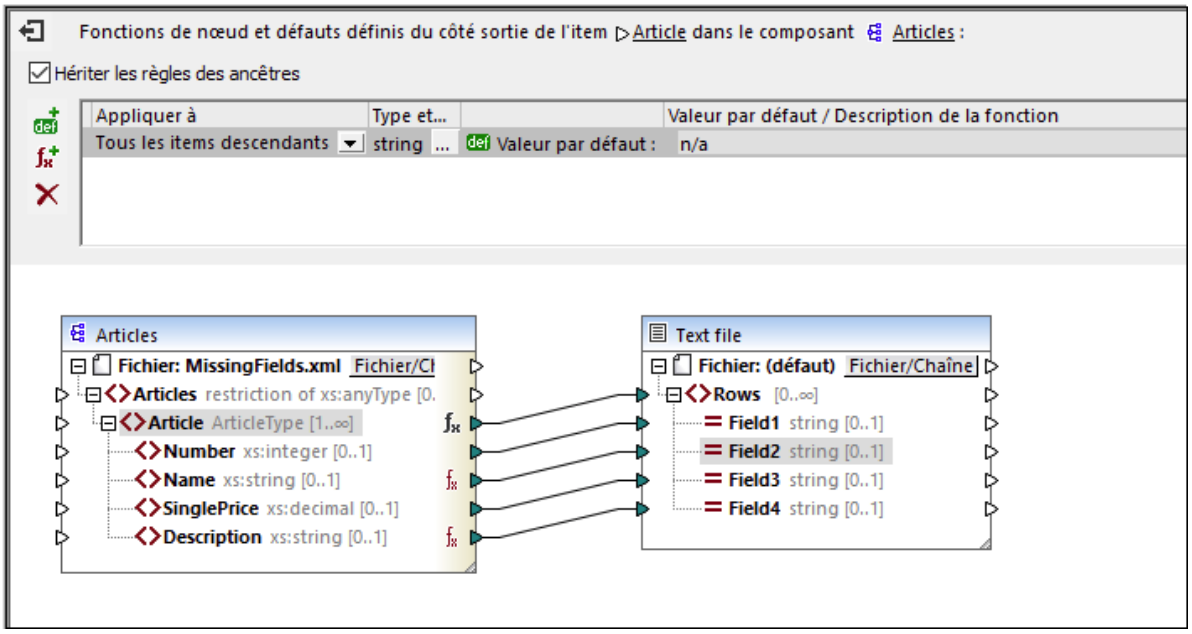
```
T-Shirt                 25.3      20        Disponible en noir, bleu et rouge
Chemise                 70.3      60
Pantalons                                   Stock limité
Veste                   57.5      40
```

Les scénarios décrits ci-dessous vous montreront comment définir un défaut pour tous les nœuds de type `string`, définir les défauts pour les différents types de données, bloquer les règles pour des nœuds spécifiques, écraser les règles héritées, et définir une fonction de nœud, et appliquer cette fonction de manière conditionnelle aux nœuds pertinents.

## Scénario 1 : Définir un défaut pour tous les nœuds de string

Dans notre mappage, quelques uns des nœuds du type `string` sont vides. Notre objectif est de définir une valeur par défaut `n/a` à la place de strings vides. Suivez les étapes suivantes :

1. Cliquez avec la touche de droite sur l'élément `Article` et sélectionnez **Fonctions de nœud et défauts | Fonctions de nœud de sortie et défauts** depuis le menu contextuel.
2. Cliquez sur l'option dans la fenêtre des Fonctions de nœuds et saisissez `n/a` dans le champ *Valeur par défaut* (*voir la capture d'écran ci-dessous*).



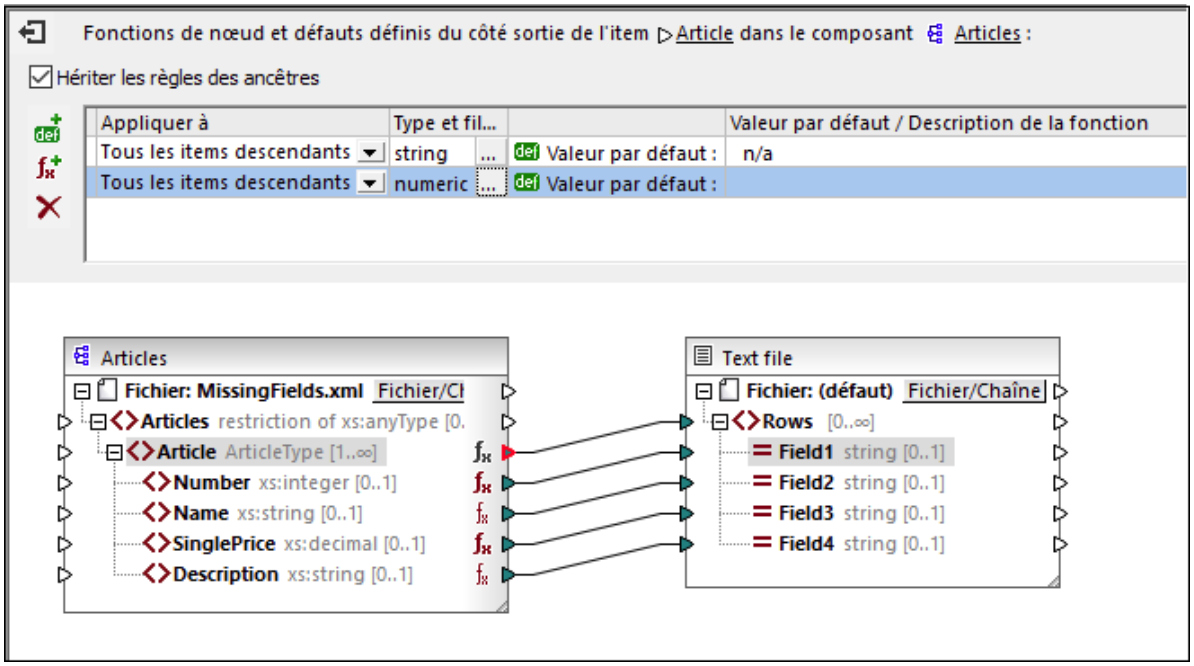

Dans le composant `ArticlesOfClothing`, les éléments `Nom` et `Description` sont de type `string`, ce qui rend la valeur par défaut appropriée pour les deux. Pour cette raison, l'icône apparaît à côté de ces nœuds. Puisque les valeurs par défaut s'appliquent uniquement aux valeurs vides, seuls deux nœuds `Field4` ont reçu les valeurs `n/a` dans la sortie (*en surbrillance jaune ci-dessous*).

```
T-Shirt             25.3      20        Disponible en noir, bleu et rouge
Chemise             70.3      60        n/a
Pantalons                               Stock limité
Veste               57.5      40        n/a
```

## Scénario 2 : Définir les défauts pour différents types de données

Outre la définition d'un défaut pour des nœuds d'un type spécifique, vous pouvez en plus définir un autre défaut pour un autre type de données. Par exemple, un des articles a des valeurs vides dans `Prix` et `ItemsInStock`, dont les deux sont de type numérique. Maintenant, notre objectif est de remplacer ces valeurs vides avec `0`. Suivez les étapes ci-dessous :

1. Répétez les étapes depuis le *Scénario 1*. Ceci ajoutera une valeur par défaut pour tous les nœuds de type `string` qui ont des valeurs vides.
2. Ajoutez un autre défaut et sélectionnez `numérique` comme type de données.
3. Tapez `0` dans le champ *Valeur par défaut* (*voir la capture d'écran ci-dessous*).



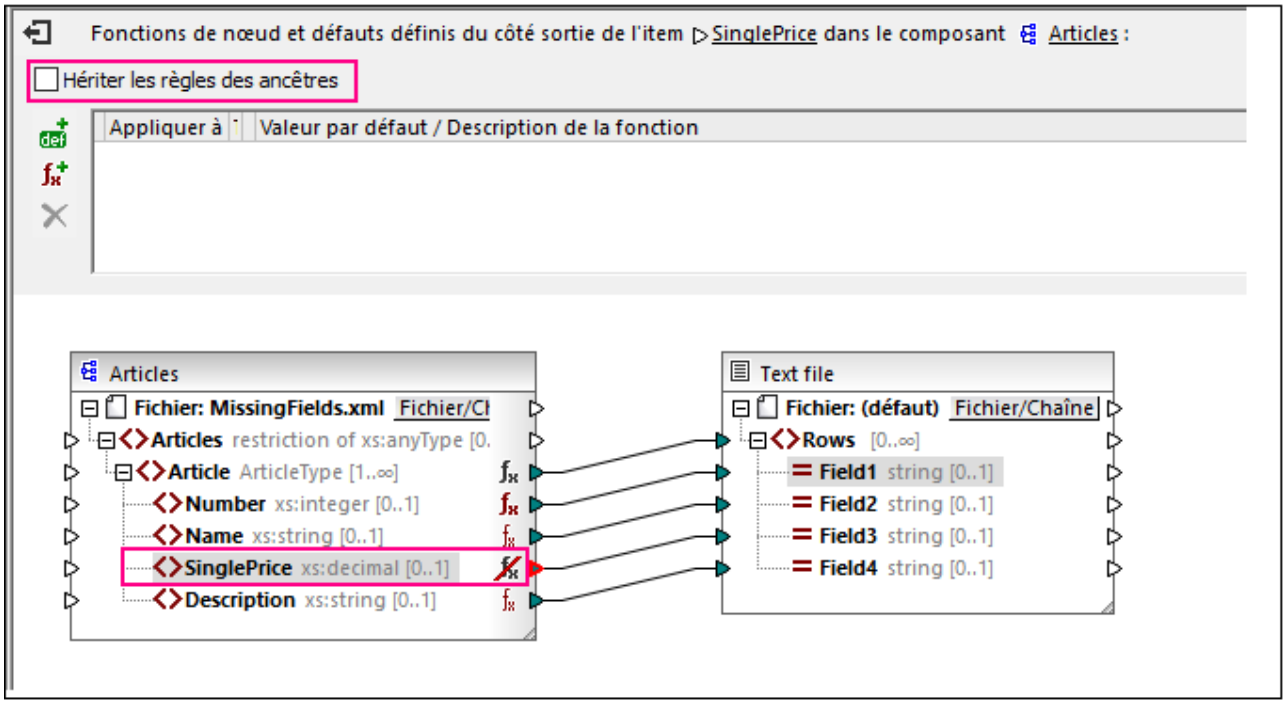

La sortie a maintenant l'air de ceci (*notez les parties en surbrillance*) :

```
T-Shirt          25.3    20          Disponible en noir, bleu et rouge
Chemise          70.3    60          n/a
Pantalons        0       0           Stock limité
Veste            57.5    40          n/a
```

**Note :** Conformément à la Spécification de schéma XML , `integer` est dérivé de `décimal`, et les deux appartiennent à un type de données numérique. Dans notre exemple, la règle s'appliquera aux deux éléments `ItemsInStock` et `Prix` si vous sélectionnez `numeric` comme type de données dans la fenêtre de fonctions de nœud. Si vous sélectionnez `decimal` comme type de données, la règle continuera à s'appliquer aux deux éléments. Toutefois, si vous sélectionnez `integer` comme type de données, la règle s'appliquera uniquement à l'élément `ItemsInStock`.

## Scénario 3 : Bloquer la règle pour des nœuds spécifiques

Dans ce scénario, nous voulons appliquer les défauts pour tous les nœuds string et numeric, mais nous ne voulons pas définir un défaut dans l'élément `Prix`. Pour atteindre cet objectif, reproduisez les étapes du *Scénario 2*, cliquez sur l'élément `Prix` et désactivez la case à cocher *Hériter des règles des ancêtres*. Dans le mappage ci-dessus, l'item `Prix` n'hérite plus des règles depuis son parent, `Article`. Pour cette raison, l'icône apparaît à côté de l'élément `Prix` (*voir la capture d'écran ci-dessous*).

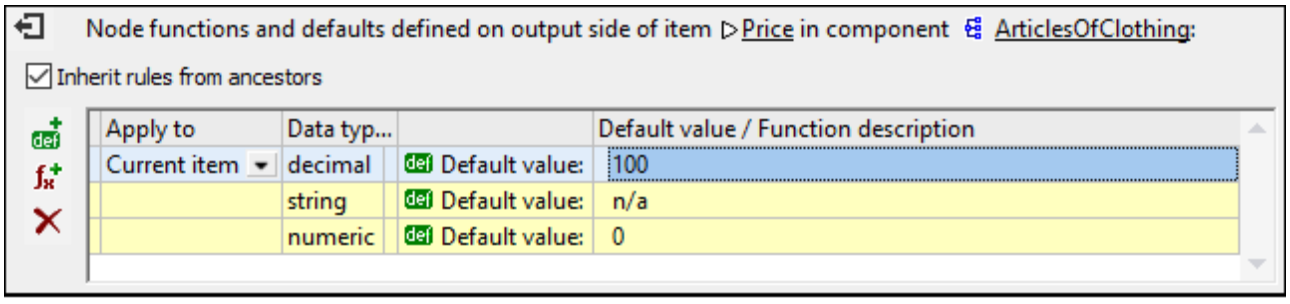



La sortie ci-dessous montre que la valeur vide de l'élément `Prix`, pour laquelle la règle a été bloquée, a été mappée vers le `Champ2` (*notez les parties en surbrillance*).

```
T-Shirt                 25.3      20        Disponible en noir, bleu et rouge
Chemise                 70.3      60        n/a
Pantalons                         0         Stock limité
Veste                   57.5      40         n/a
```

## Scénario 4 : Écraser les règles héritées

Dans ce scénario, nous voulons définir les défauts pour tous les nœuds string et numériques ; toutefois, pour l'élément `Prix` exclusivement, nous voulons définir `100` comme valeur par défaut. Pour atteindre cet objectif, répétez les étapes décrites dans le *Scénario 2*, cliquez sur `Prix` dans le composant `ArticlesOfClothing` et ajoutez un nouveau défaut, tel qu'affiché dans la capture d'écran ci-dessous. Les règles héritées ont un fond jaune.



La règle définie directement dans un nœud a la priorité sur les règles créées au niveau parent. Pour cette raison, la sortie a maintenant l'aspect suivant (*notez les parties surlignées*) :

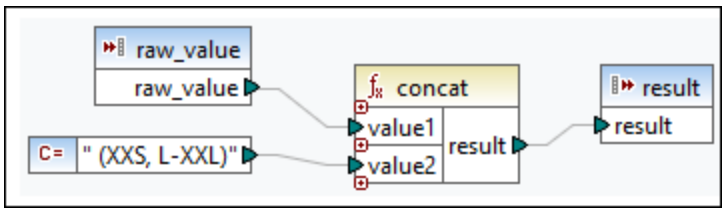
```
T-Shirt                    25.3      20    Disponible en noir, bleu et rouge
Chemise                    70.3      60    n/a
Pantalons                  100       0     Stock limité
Veste                      57.5      40    n/a
```

Si vous avez défini plus d'une règle pour le même nœud, la règle au niveau supérieur de la grille s'applique au nœud actuel. Vous pouvez permuter l'ordre de ces règles en glissant manuellement les règles dans la grille.

## Scénario 5 : Créer une fonction de nœud et l'appliquer aux conditions

Ce scénario expliquera comment créer une fonction de nœud et l'appliquer aux conditions. Nous utiliserons la fonction **concat** pour ajouter du texte aux nœuds du type `string`. Suivez les instructions ci-dessous :

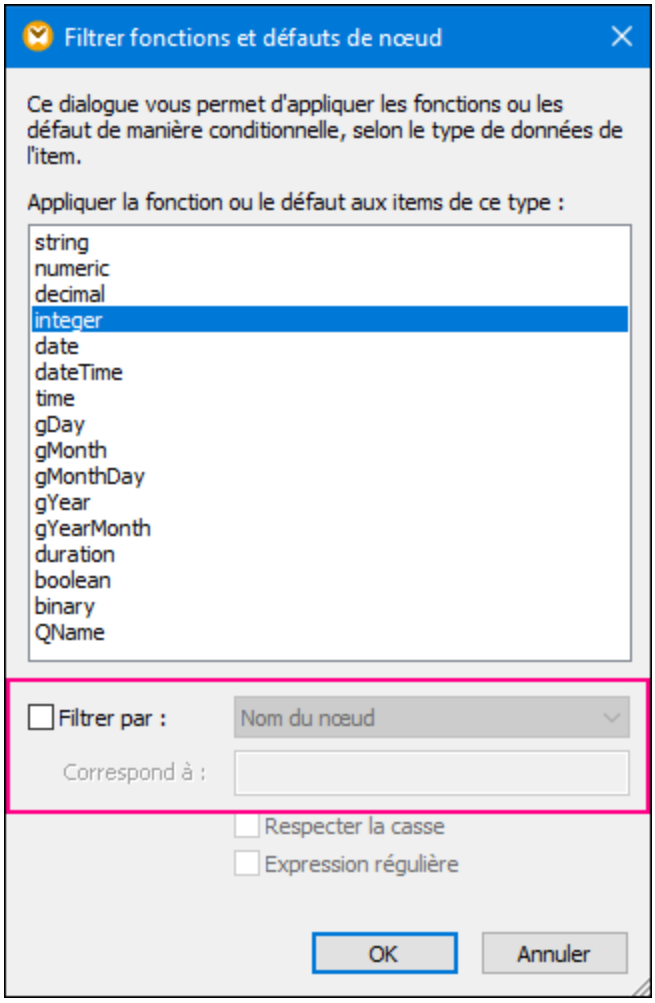
1. Cliquez avec la touche de droite sur l'élément `Article` et sélectionnez **Fonctions de nœud et défauts | Fonctions de nœud de sortie et défauts** depuis le menu contextuel.
2. Ajoutez une fonction de nœud en cliquant sur l'icône et créez la fonction affichée ci-dessous.




À cette étape, nous n'avons pas filtré les nœuds et la sortie a l'aspect suivant (*notez les parties en surbrillance*) :

```
T-Shirt    (XXS, L-XXL)       25.3      20    Disponible en noir, bleu et rouge (XXS, L-
XXL)
Chemise    (XXS, L-XXL)       70.3      60

Pantalons  (XXS, L-XXL)                       Stock limité (XXS, L-XXL)

Veste      (XXS, L-XXL)       57.5      40
```

La sortie affiche que notre règle a été appliquée aux deux éléments de type `string` : `Nom` et `Description`. Maintenant, nous voulons que la règle s'applique uniquement à l'élément `Nom`. À cette fin, allez à la fenêtre des fonctions de nœud et cliquez sur la touche [...] dans la colonne *Type et filtre de données*. Ceci ouvre la boîte de dialogue **Filtrer Fonctions de nœud et Défauts**, dans laquelle vous pouvez changer les types de données et spécifier les options de filtrage. Définissez le filtre tel qu'affiché dans la capture d'écran ci-dessous.




La sortie a désormais l'aspect suivant :

```
T-Shirt    (XXS, L-XXL)        25.3      20    Disponible en noir, bleu et rouge
Chemise    (XXS, L-XXL)        70.3      60

Pantalons  (XXS, L-XXL)                        Stock limité
Veste      (XXS, L-XXL)        57.5      40
```

*Filtrer par type de nœud*
Cette option vous permet de réduire le type de données que vous avez sélectionné dans le dialogue **Filtrer Fonctions de nœud et Défauts**. Par exemple, vous avez choisi d'appliquer votre fonction de nœud aux nœuds numériques. Ensuite, vous pouvez spécifier un sous-type (par ex., *Filtrer par/nom de nœud :* `xs:decimal`).

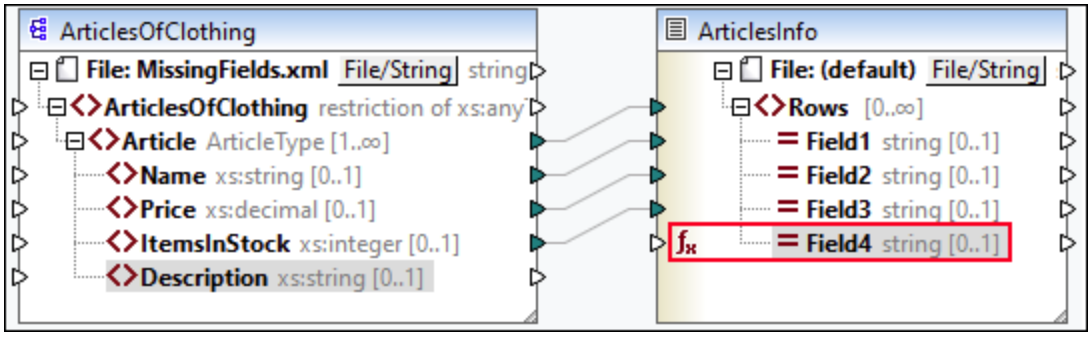
*Filtrer avec des expressions régulières*
Pour plus d'options de filtrage avancées, vous pouvez utiliser des expressions régulières (par ex., pour faire correspondre des noms de nœud multiples). Pour plus d'informations, voir Expressions régulières 543. Veuillez prendre note des points suivants :

- Les ancres `^` et `$` sont implicites et ne doivent pas être saisies dans la case *Correspond à*.
- Le respect de la casse peut être spécifié dans la case á cocher *Correspond à* . Pour cette raison, l'indicateur `i` n'est pas pris en charge.
- La correspondance sur plusieurs lignes n'est pas pertinente pour le filtrage de nœud. Pour cette raison, l'indicateur `m` n'est pas pris en charge.

### Scénario 6 : Créer des défauts pour des nœuds non connectés

MapForce vous permet d'appliquer les défauts aux nœuds non connectés, mais uniquement si le nœud parent et/ou un frère sont connectés. Considérez l'exemple suivant :




Dans le mappage ci-dessus, nous avons défini une valeur par défaut pour le nœud `Champ4` dans le composant `ArticlesInfo`. À cette étape, `Champ4` n'a pas de connexion d'entrée. Les nœuds parent et frère sont connectés. Si le `Champ4` reste non connecté, la valeur par défaut écrasera toutes les valeurs de `Description` dans la sortie (*voir ci-dessous*). Si vous connectez `Description` et `Champ4`, la valeur par défaut sera appliquée aux séquences vides.

```
T-Shirt                    25.5    20    n/a
Chemise                    70.3    60    n/a
Pantalons                                n/a
Veste                      57.5    40    n/a
```

**Note :** les défauts ne sont pas appliqués aux nœuds non connectés dans des composants JSON.

## 6.3.3 Métadonnées de nœud dans des fonctions de nœud

Dans certains cas, il se peut que vous souhaitiez que la fonction de nœud fasse quelque chose sur la base d'informations concernant le nœud actuel, à savoir les *métadonnées de nœud*. Le terme « métadonnées de nœud » signifie des informations diverses concernant le nœud auquel la fonction s'applique (par ex., le nom de nœud, la longueur de valeur et la précision), qui est stocké dans le schéma. La table ci-dessous recense tous les paramètres de métadonnées possibles que vous pouvez utiliser dans une fonction de nœud. Notez que certains paramètres de métadonnées s'appliquent uniquement à des types de données spécifiques. MapForce affichera un avertissement lorsque vous tentez d'utiliser des métadonnées qui sont incompatibles avec le nœud actuel.

| Paramètres de métadonnées | Description |
|---|---|
| node_name | Fournit le nom du nœud actuel tel qu'affiché dans le composant. Ces métadonnées sont prise en charge pour tous les nœuds. Dans les fichiers XML, node_name fait référence au nom de l'élément ou à l'attribut actuel. Dans les composants CSV, node_name fait référence au nom d'un champ CSV. Dans les base de données, node_name fait référence au nom d'une colonne de table. |
| node_annotation | Affiche un texte d'annotation à côté de la valeur du nœud dans la sortie. Ces métadonnées sont prises en charge pour tous les nœuds. |
| node_minLength | Fournit la valeur de la facette minLength du type de données du nœud. Applicable aux nœuds XML et de texte avec des types appropriés. Pour plus d'information sur les facettes de contrainte Length, voir la Recommandation de schéma XML. |
| node_maxLength | Fournit la valeur de la facette maxLength du type de données du nœud. Applicable aux nœuds XML et de texte avec des types appropriés. Pour plus d'information sur les facettes de contrainte Length, voir la Recommandation de schéma XML. |
| node_totalDigits | Fournit la valeur de la facette totalDigits du type de données du nœud. Applicable aux nœuds XML avec des types appropriés. Pour les détails sur la facette totalDigits, voir la Recommandation de schéma XML. |
| node_fractionDigits | Fournit la valeur de la facette fractionDigits du type de données du nœud. Applicable aux nœuds XML avec des types appropriés. Pour les détails sur la facette fractionDigits, voir la Recommandation de schéma XML. |
| node_length | Fournit la length (en octets) d'un nombre. Applicable aux champs de base de données avec des types appropriés. Pour en savoir plus sur length, voir la documentation Microsoft. |
| node_precision | Fournit le nombre de tous les chiffres d'un nombre. Applicable aux champs de base de données avec des types appropriés. Pour plus d'informations, voir la documentation Microsoft. |
| node_scale | Fournit les chiffres du numéro à droite du point décimal d'un nombre. Applicable aux champs de base de données avec des types appropriés. Pour plus d'informations, voir la documentation Microsoft. |

**Note :** le type de données d'un paramètre de métadonnées doit coïncider avec le type de données du nœud pertinent. Autrement, l'exécution de la fonction échouera.

*Ajouter les métadonnées à fonction de nœud*

Pour ajouter un paramètre de métadonnées à une fonction de nœud, vous devez créer une fonction de nœud[479], puis cliquer sur l'option **Ajouter des spécificités du nœud**, qui est située sous la fenêtre des fonctions de nœud. Ceci ouvrira le dialogue **Insérer entrée avec spécificité de nœud**, dans laquelle vous pouvez sélectionner un paramètre de métadonnées pertinent. À la place de l'option **Ajouter les spécificités du nœud**, vous pouvez aussi choisir une des options suivantes :

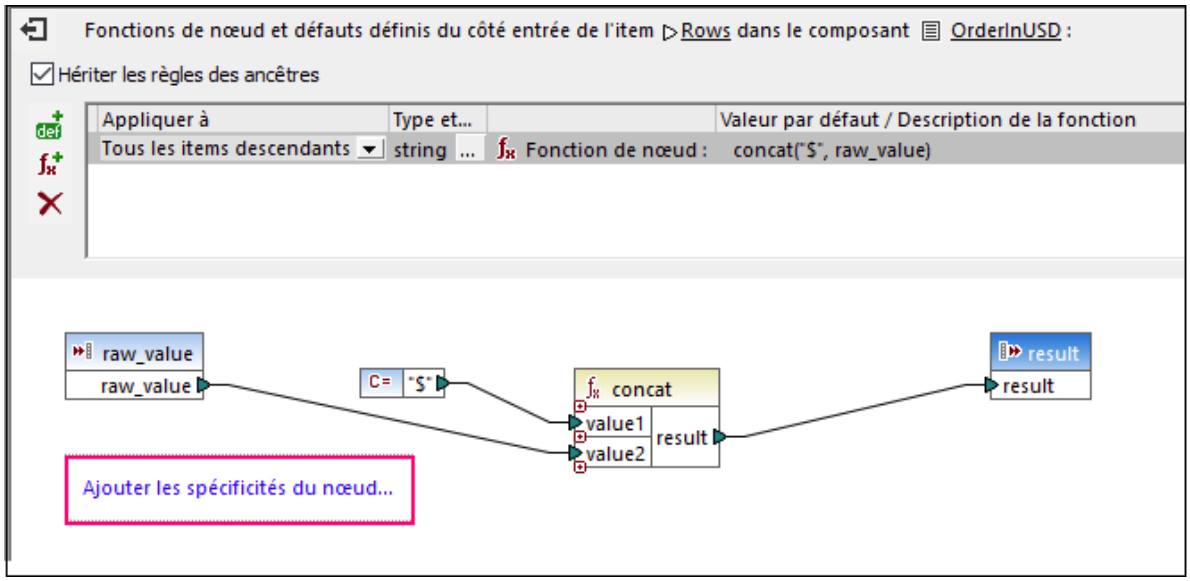
- Cliquez sur la touche de droite dans une zone vide dans le mappage et sélectionnez **Insérer entrée** depuis le menu contextuel.
- Cliquez sur la commande de la barre d'outils.
- Cliquez **Insérer entrée** dans le menu **Fonction**.

*Lorsque les métadonnées ne sont pas prises en charge par un nœud*
Lorsqu'un paramètre de métadonnées n'est pas pris en charge par un nœud, vous pouvez donner l'instruction à MapForce de renvoyer une séquence vide ou de ne pas appliquer la fonction de nœud (les deux options sont disponibles dans le dialogue **Insérer entrée avec la spécificité de nœud**). La séquence vide doit être gérée ; autrement, la fonction nœud peut ne retourner aucune valeur. Généralement, vous devez utiliser les fonctions de séquence (comme `substitute-missing`[625] et `exists`[599]), ou d'autres types de composant pour le traiter ultérieurement.

## Exemple

Cet exemple ci-dessous explique comment fournir le texte d'annotation à une fonction de nœud. Pour cet exemple, vous allez avoir besoin du mappage suivant : **`<Documents>\Altova\MapForce2024\MapForceExamples\Tutorial\MissingFields.mfd`**. Dans **`MissingFields.xsd`** (sur lequel **`MissingFields.xml`** est basé), nous avons ajouté l'annotation suivante pour l'élément `Name` : `Nom de l'article du catalogue Printemps 2022`. Notre objectif est de fournir cette annotation à une fonction de nœud et de voir l'annotation dans la sortie. Suivez les instructions ci-dessous :

1. Créer une fonction de nœud[479] pour l'élément `Name` dans le composant `ArticlesOfClothing`.
2. Répliquer la fonction affichée ci-dessous. Pour ajouter un paramètre `node_annotation`, cliquez sur **Ajouter des spécifiques de nœud** (*voir le rectangle rouge ci-dessous*) et sélectionnez ce paramètre depuis la liste. Pour savoir comment ajouter et manipuler des fonctions, voir Notions fondamentales de bases[468].




**Note :** par défaut, des structures imbriquées profondément ne sont pas entièrement scannées, afin de pouvoir préserver la mémoire et d'améliorer l'expérience de l'utilisateur. Si le composant dans lequel vous appliquez la fonction de nœud possède de telles structures imbriquées, vous pouvez agrandir les nœuds pertinents sur le mappage afin de rendre MapForce sensible aux nœuds. Dans ce cas,

MapForce prendra en compte les nœuds agrandis lorsque vous ajoutez un nouveau paramètre de métadonnées, et l'avertissement peut disparaître. La fonction de nœud doit être connectée pour être appliquée ; élargir des éléments non connectés n'est pas pertinent.

*Sortie*

La liste ci-dessous montre que l'annotation a été ajoutée aux valeurs des éléments `Name`.

```
T-Shirt (nom de l'article du catalogue Printemps 2022)   25.5  20    Dispo en noir, bleu
et rouge Chemise (nom de l'article du catalogue Printemps 2022)   70.3  60
                  Pantalons (nom de l'article du catalogue Printemps 2022)
Stock limité
Veste (nom de l'article du catalogue Printemps 2022)     57.5  40
```

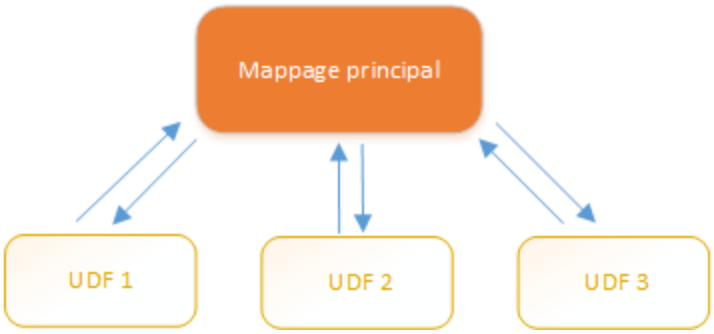
# 6.4 Fonctions définies par l'utilisateur

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 :

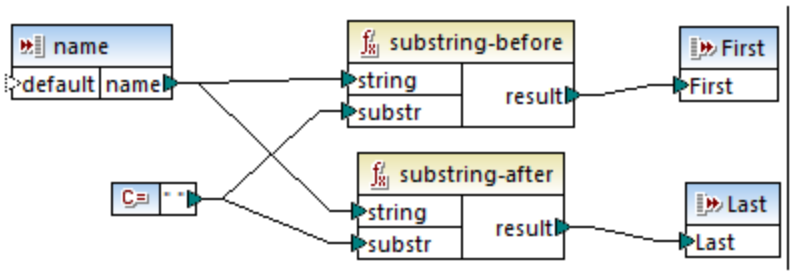
- Les FDU sont réutilisables à l'intérieur d'un mappage ou dans de multiples mappages.
- Les FDU peuvent faciliter la lecture de votre mappage : Par exemple, vous pouvez mettre en packages des parties du mappage en composants plus petits et abstraire des détails de mise en œuvre. Le diagramme ci-dessous 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 504.
- 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) 494. Les FDU récursives 502 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 492
- Paramètres FDU 497
- FDU récursives 502
- Implémentation de la consultation 504

## 6.4.1 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

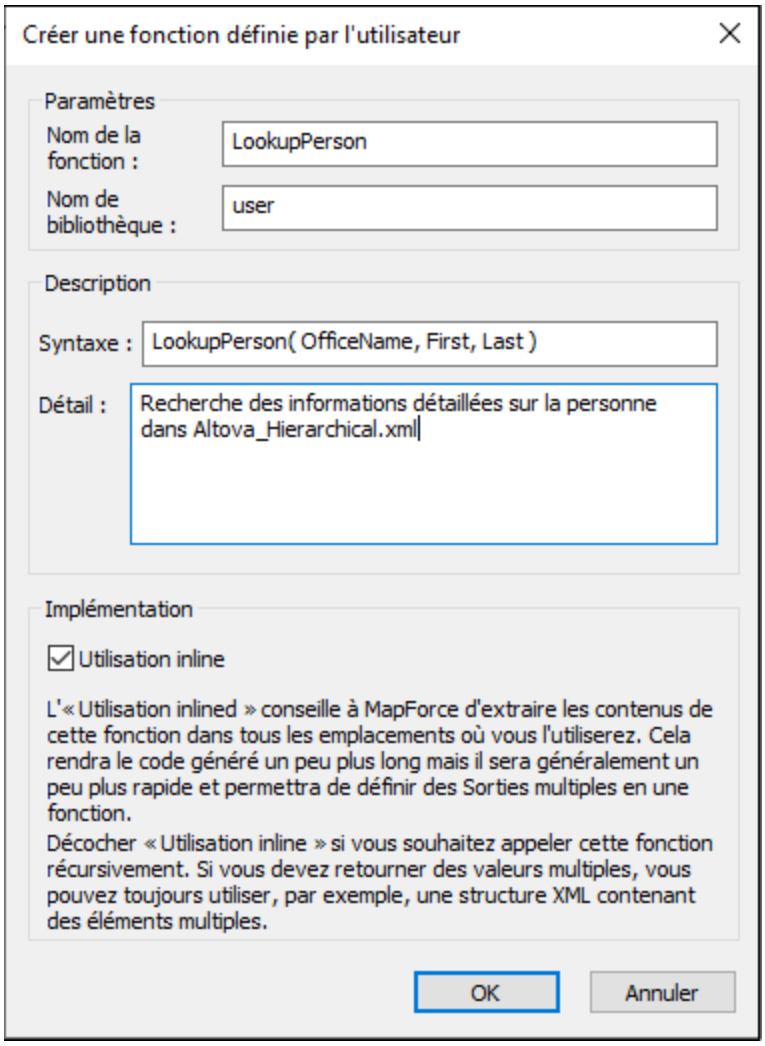
Cette sous-section explique comment créer une FDU de zéro et de composants déjà existants. L'exigence minimum est un compostant de sortie auquel quelques données sont connectées. Pour les paramètres d'entrée, une fonction peut avoir zéro, une ou plusieurs entrées. Les paramètres d'entrée et de sortie 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 497.

*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 26) 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 814 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 804, et ceci pourrait amener le mappage à produire un résultat différent.

## Appeler et importer les UDF

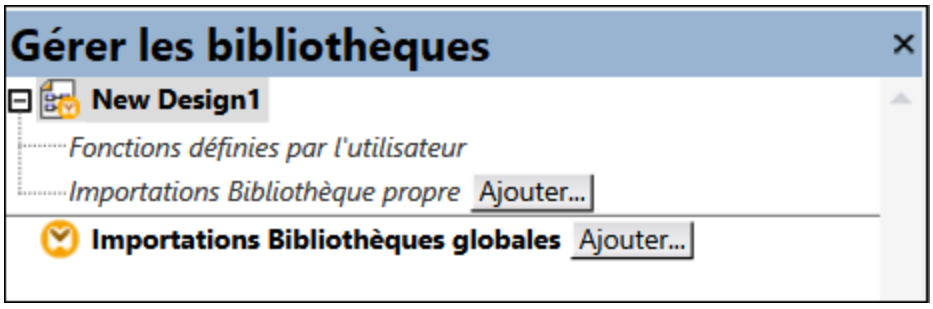
Après avoir créé une FDU, vous pouvez l'appeler du même mappage dans lequel vous l'avez créé ou depuis tout autre mappage.

*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 26. 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 471.

*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.

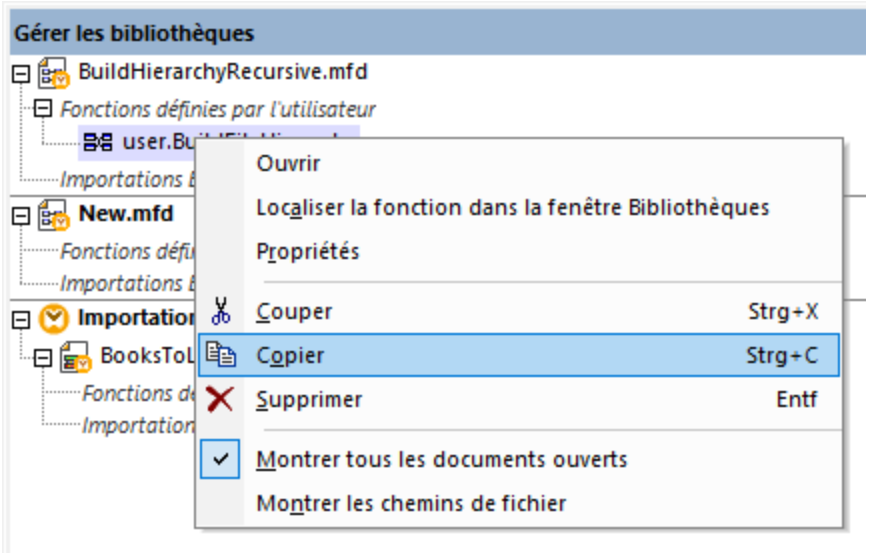
Pour retourner au mappage principal, cliquer sur la touche dans le coin supérieur gauche de la fenêtre de mappage.

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 471.
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 47.
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

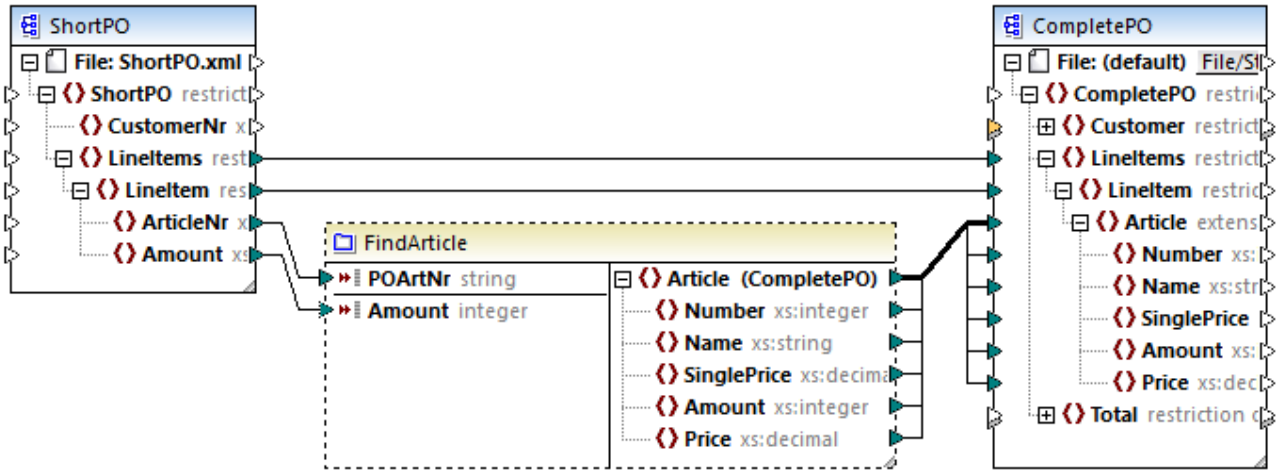
Pour supprimer une FDU, suivez les étapes ci-dessous :

1. Double-cliquer sur la barre de titre de la FDU dans le mappage.
2. Cliquer sur la touche ERASE 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.

## 6.4.2 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 498. 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

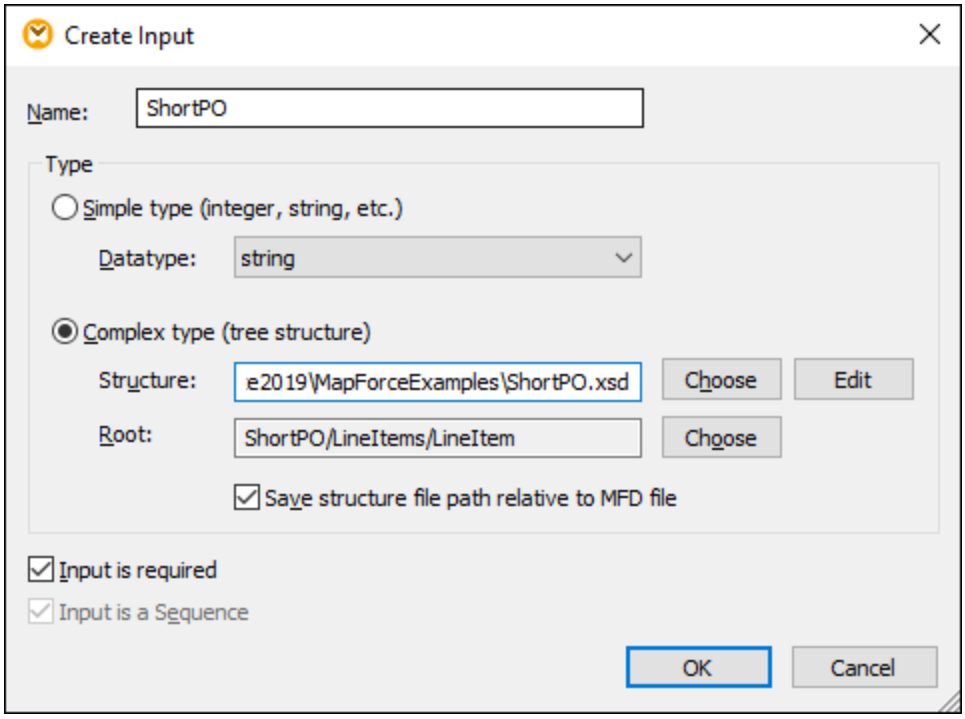
*UDF basées sur les structures de base de données (éditions Professional et Enterprise)*
MapForce vous permet de créer des paramètres UDF basés sur BD avec une arborescence de tables associées. L'arborescence de tables associées représente une structure in-memory qui n'a pas de connexion

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[492] ou ouvrir une tâche existante[495].
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.



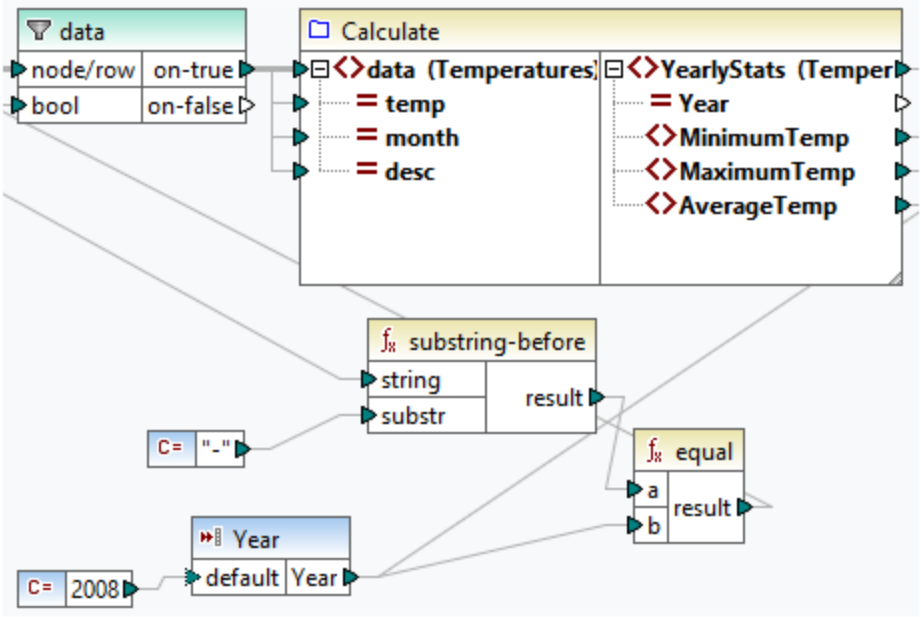

3. Choisissez si les paramètres d'entrée ou de sortie doivent être de type simple ou complexe (*voir la boite de dialogue ci-dessus*). Voir la liste des structures complexes disponibles ci-dessous. Par exemple, pour créer un paramètre qui est de type XML complexe, cliquez sur **Choisir** à côté de *Structure* 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[47]. Les cases à cocher *Entrée requise* et *Entrée est une séquence* sont expliquées dans les sous-sections suivantes.

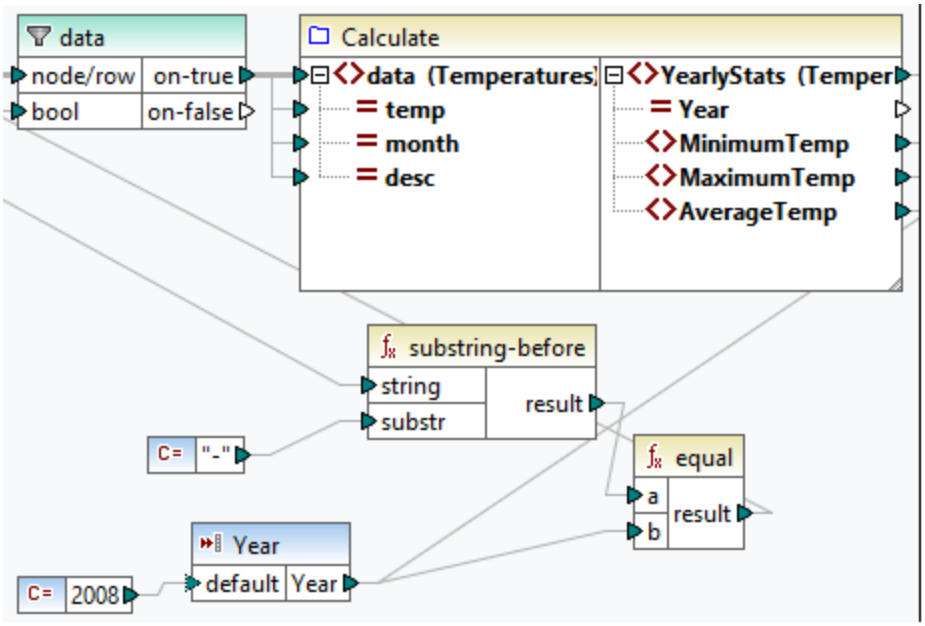
*L'entrée est requise*
Pour rendre un paramètre obligatoire dans une FDU, sélectionnez la case *Entrée est requise* (*voir la boîte de dialogue ci-dessus*). Si vous effacez la case à cocher *Entrée requise*, le paramètre deviendra optionnel et aura une bordure en pointillés dans le mappage.

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.

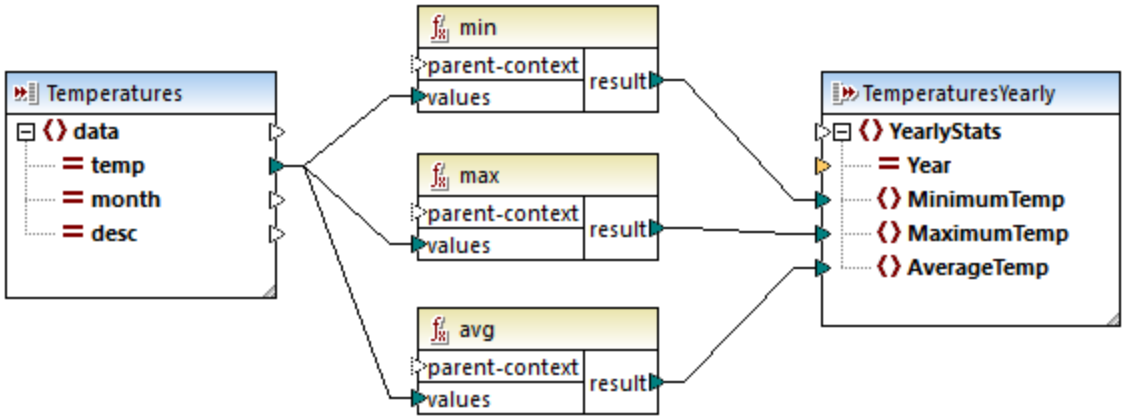



*L'entrée est une séquence*
En option, vous pouvez définir si un paramètre de fonction doit être traité en tant que valeur simple (option par défaut) ou en tant que séquence. Une séquence est une plage de zéro ou de plusieurs valeurs. Une séquence peut être utile quand votre fonction définie par l'utilisateur 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[494].

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

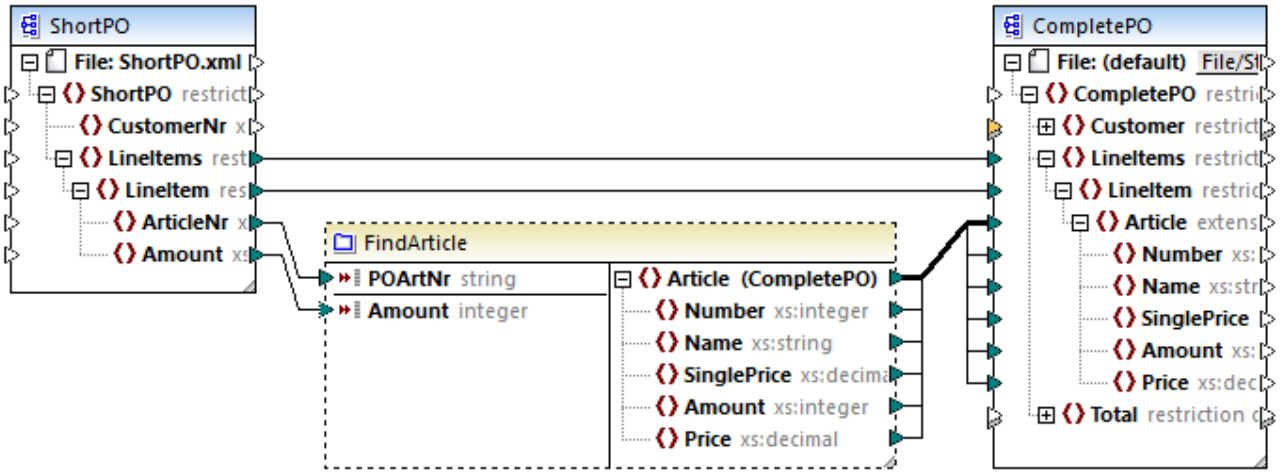
`substitute-missing` 625 avant l'entrée de fonction pour assurer que la séquence n'est jamais vide. En alternative, spécifiez le paramètre comme a séquence et ajoutez la gestion pour la séquence vide à l'intérieur de la fonction.

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.

## 6.4.3 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.

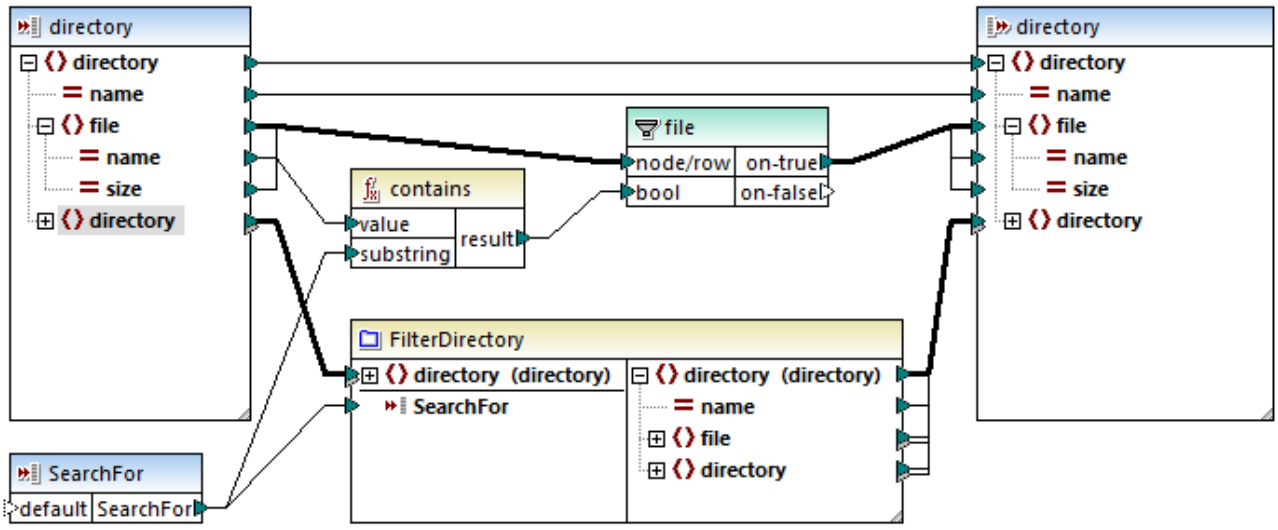
### Fichier source

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[494].




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.

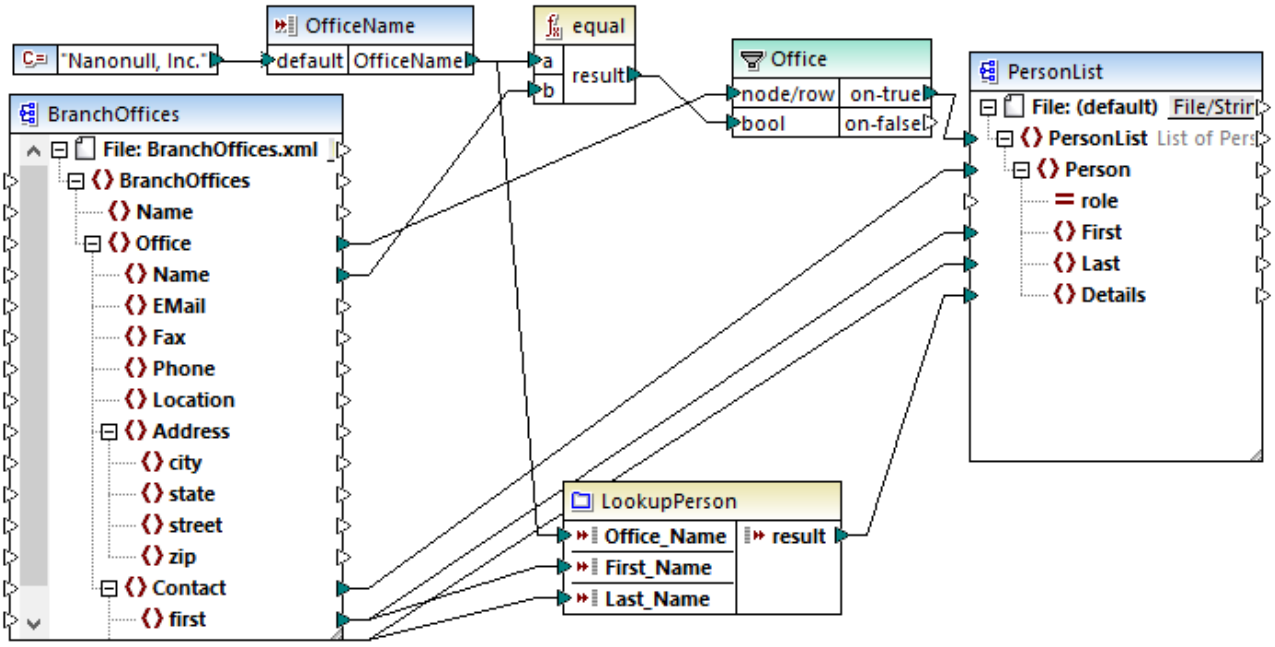
### Sortie

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>
```

## 6.4.4 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 :

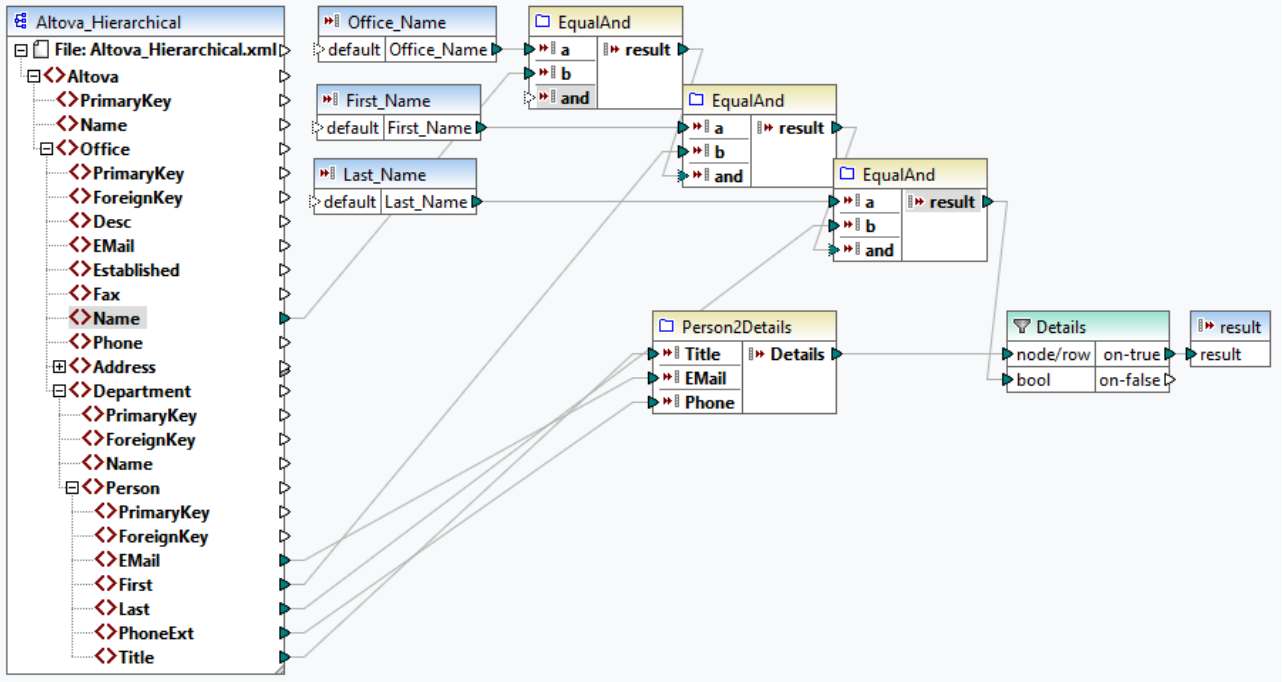
- Pour rechercher des données sur chaque employé (leur poste, adresse e-mail et titre) dans un fichier XML.
- Pour présenter ces données en tant que liste séparée par une virgule et mapper cette liste dans l'élément `Détails` de l'XML cible.
- Pour extraire l'information sur les employés uniquement d'un 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 `Détails` 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.

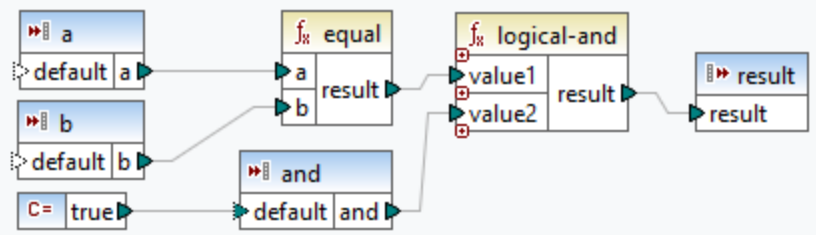



La FDU est définie comme suit :

- 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*

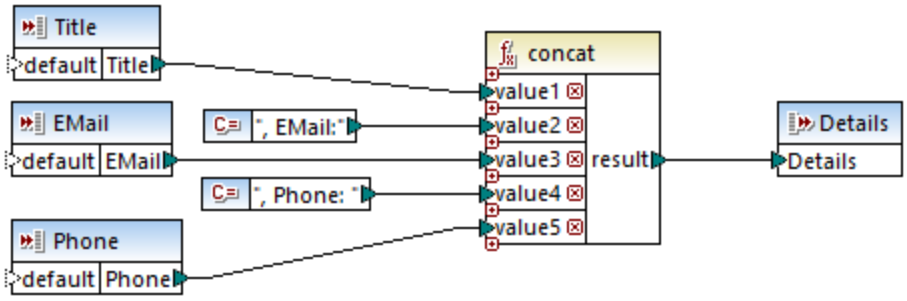
La fonction `EqualAnd` est une FDU séparée (*voir ci-dessous*) définie à l'intérieur de la FDU `LookupPerson`. Pour voir la structure interne de l'UDF `EqualAnd`, double-cliquez sur l'en-tête de la fonction.



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>
```

# 6.5 Fonctions personnalisées

Cette section explique comment importer les bibliothèques XSLT[508], XQuery[515], Java et .NET[520]. Cette section montre également comment référencer des bibliothèquesC#, C++ et Java manuellement[528].

## 6.5.1 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[26]. 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[22]. Voir aussi Gérer les Bibliothèques de fonction[471].

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 70.

### 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 509
Exemple : Totaliser les valeurs de nœud 512
Implémentation de moteur XSLT 1.0 1365
Implémentation de moteur XSLT 2.0 1366

## 6.5.1.1 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**.

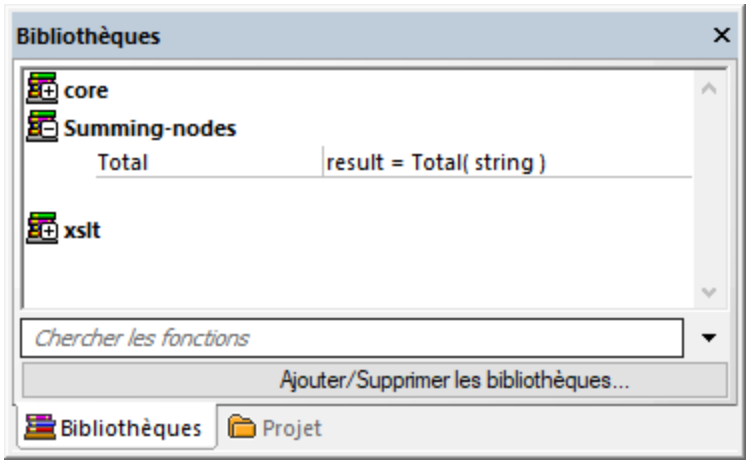
- Name-splitter.xslt. Ce fichier XSLT défini un modèle nommé appelé "**tokenize**" avec un seul paramètre "string". Le modèle fonctionne par le biais d'un string d'entrée et sépare les caractères en majuscule avec un espace pour chaque occurrence.
- Name-splitter.xml (le fichier d'instance XML de source 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[26]. La fenêtre **Gérer les Bibliothèques** s'ouvre (*voir la capture d'écran ci-dessous*).

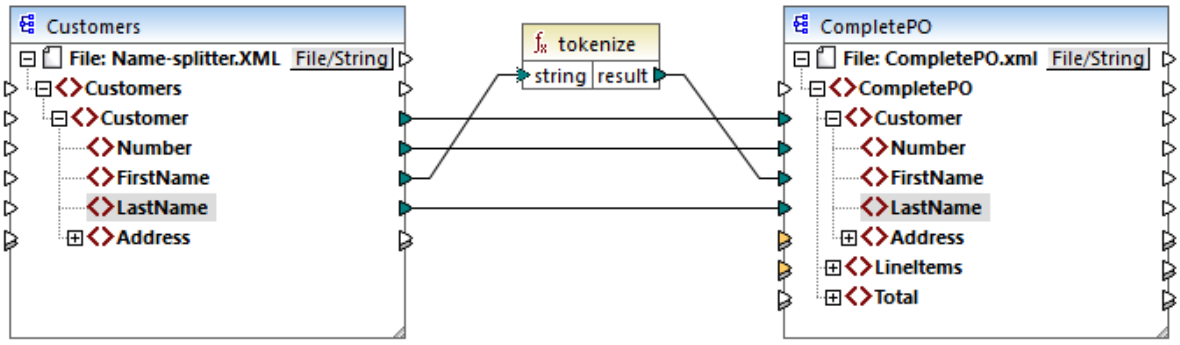

2. Pour importer des fonctions en tant que bibliothèque *locale* (uniquement dans le cadre du document actuel), cliquez sur **Ajouter** sous le nom actuel du mappage. Pour importer les fonctions en tant que bibliothèque *globale* (au niveau du programme), cliquez sur **Ajouter** à côté des **Importations 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.

## 6.5.1.2 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** :

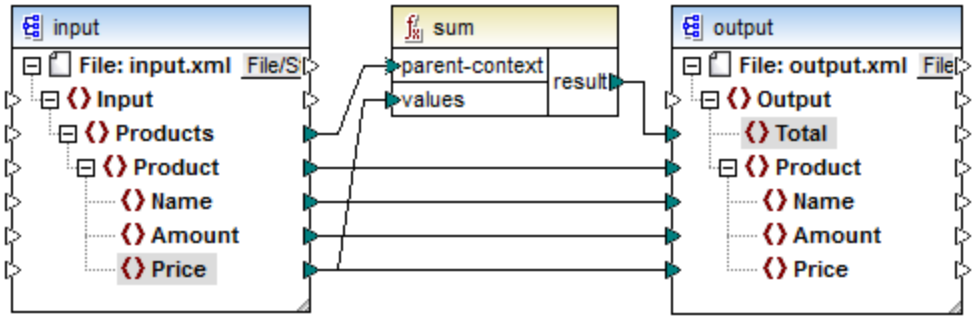
- **Summing-nodes.mfd** — le fichier de mappage
- **input.xml** — le fichier XML de source
- **input.xsd** — le schéma XML de source
- **output.xsd** — le schéma XML de cible
- **Summing-nodes.xslt** — Une feuille de style XSLT personnalisée contenant un modèle nommé pour additionner les nœuds individuels.

Il existe deux moyens différents d'atteindre l'objectif du mappage :

- En utilisant la fonction **sum** 556. 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 549.

### 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>
```

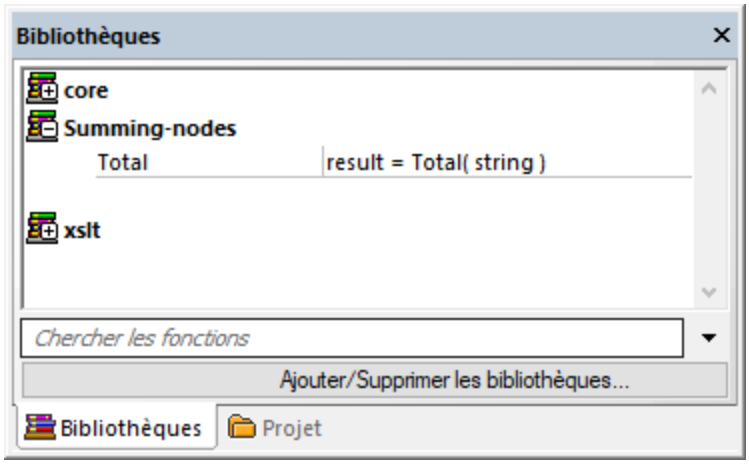
**Note :** Pour additionner les nœuds dans XSLT 2.0, modifier la déclaration de feuille de style en `version="2.0"`.

Avant d'importer la feuille de style XSLT dans MapForce, choisir XSLT 1.0 en tant que langage de transformation[22]. Vous êtes maintenant prêt à importer la fonction personnalisée, comme suit :

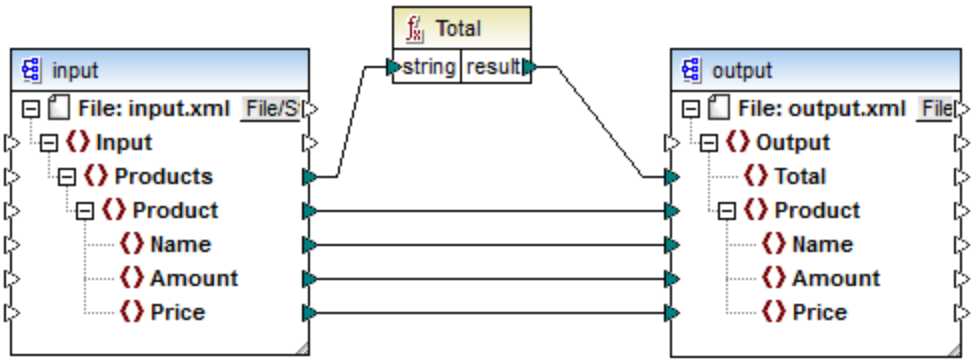
1. Cliquez sur la touche **Ajouter/Supprimer des bibliothèques** en bas de la fenêtre Bibliothèques[26]. La fenêtre **Gérer les Bibliothèques** s'ouvre (*voir la capture d'écran ci-dessous*).


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>
```

## 6.5.2 Importer une fonction XQuery 1.0 personnalisée

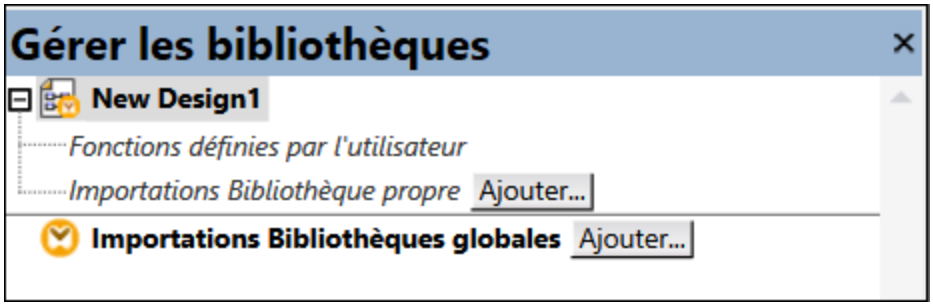
Lorsque XQuery est sélectionné en tant que langage de transformation de mappage, MapForce affiche les bibliothèques de fonctions intégrées disponibles pour XQuery dans la fenêtre de Bibliothèque. Si nécessaire, vous pouvez élargir la liste avec des fonctions Xquery personnalisées, en important des modules de bibliothèque 1.0 XQuery personnalisées dans MapForce.

Pour être éligible à l'importation dans MapForce, un fichier XQuery doit satisfaire aux exigences suivantes :

- Cela doit être un module de bibliothèque valide conformément à la spécification XQuery. En d'autres termes, il doit commencer avec une déclaration de module comme `module namespace <prefix>="<namespace name"`
- Toutes les fonctions déclarées dans le module de bibliothèque importé doivent retourner des types de données atomiques (par exemple, `xs:string`, `xs:boolean`, `xs:integer`, etc). Les paramètres de fonction doivent aussi avoir des types atomiques.

### Pour importer un module de bibliothèque XQuery :

1. Cliquez sur la touche **Ajouter/Supprimer des bibliothèques** en bas de la fenêtre Bibliothèques 26. 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. Recherchez le fichier de bibliothèque .xq ou .xquery et cliquez sur **Ouvrir**.

Les modules de bibliothèques importées apparaissent dans la fenêtre Bibliothèques, et vous pourrez glisser des fonctions spécifiques dans la zone de mappage et les utiliser comme tout autre composant de fonction MapForce, voir aussi Ajouter une fonction au mappage 468.

Si vous ne voyez pas le module de bibliothèque XQuery importé, veuillez vous assurer que XQuery est sélectionné en tant que langage de transformation 22.

### Voir aussi :

Mise en place du moteur XQuery 1368

## 6.5.2.1 Example: Import Custom XQuery Function

Cet exemple vous montre comment importer un module de bibliothèque XQuery de démonstration dans MapForce et appeler ses fonctions depuis un mappage. Le module de démonstration contenu dans cet exemple consiste en une seule fonction qui calcule des taxes sur les montants décimaux à 20% du montant. Dans un scénario de production, un module XQuery peut contenir plusieurs fonctions.

> Toutes les fonctions déclarées dans le module XQuery doivent retourner des types atomiques et leurs paramètres doivent aussi être de types de données atomiques. Sinon, le module n'est pas éligible pour une importation dans MapForce.

Vous trouverez le fichier de module XQuery de démonstration dans le chemin suivant relatif à votre dossier personnel "Documents" sur l'ordinateur dans lequel MapForce est installé : **<Documents>\Altova\MapForce2024\MapForceExamples\Tutorial\module.xq**.

```
xquery version "1.0";

module namespace demo="http://www.altova.com/mapforce/demo";

declare function demo:calculatetax($val as xs:decimal) as xs:decimal {
   $val*0.2
};
```

*module.xq*

Une fois avoir importé le fichier de module XQuery dans MapForce, vous pourrez appeler la fonction **demo:calculatetax** depuis un mappage. Veuillez noter que le calcul du montant des impôts à l'aide d'une fonction XQuery n'est destinée qu'à des fins de démonstration, vous pouvez obtenir le même résultat en utilisant les fonctions built-in de MapForce.

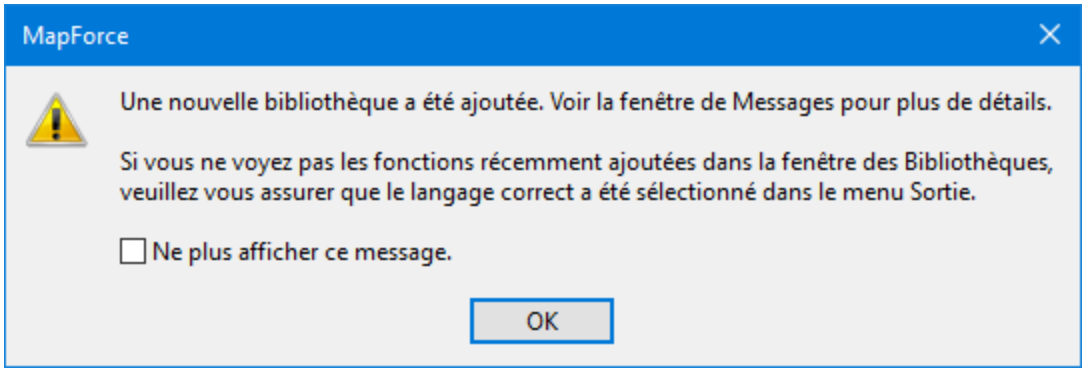
Un mappage de démonstration qui appelle la fonction **demo:calculatetax** ci-dessus est disponible sous le chemin suivant :
**<Documents>\Altova\MapForce2024\MapForceExamples\Tutorial\CalculateTax_XQuery.mfd**. Lorsque vous ouvrez tout d'abord ce mappage, MapForce affiche un avertissement qu'il contient un ou plusieurs composants qui ne sont pas disponibles dans XQuery. Cet avertissement est normal et se produit parce que le mappage référence une fonction provenant d'un module de bibliothèque XQuery personnalisé qui n'a pas encore été importé. Afin de supprimer l'avertissement et exécuter le mappage, nous allons importer le module XQuery manquant dans MapForce comme indiqué ci-dessous.

### Pour importer le module XQuery dans MapForce :

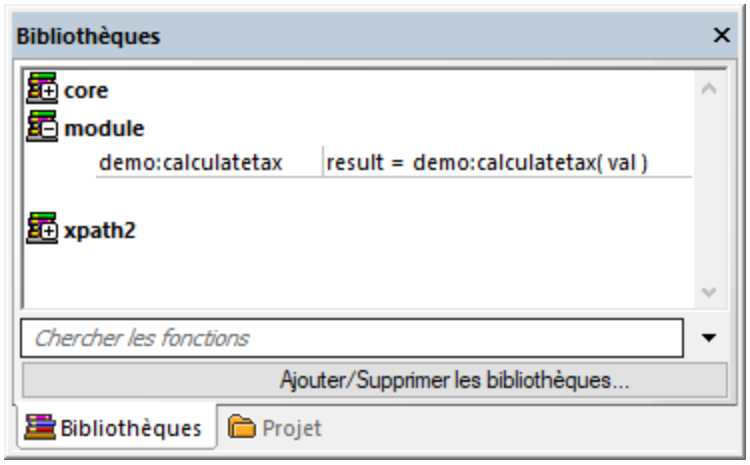
1. Cliquez sur la touche **Ajouter/Supprimer des bibliothèques** en bas de la fenêtre Bibliothèques[26]. La fenêtre **Gérer les Bibliothèques** s'ouvre (*voir la capture d'écran ci-dessous*).


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.

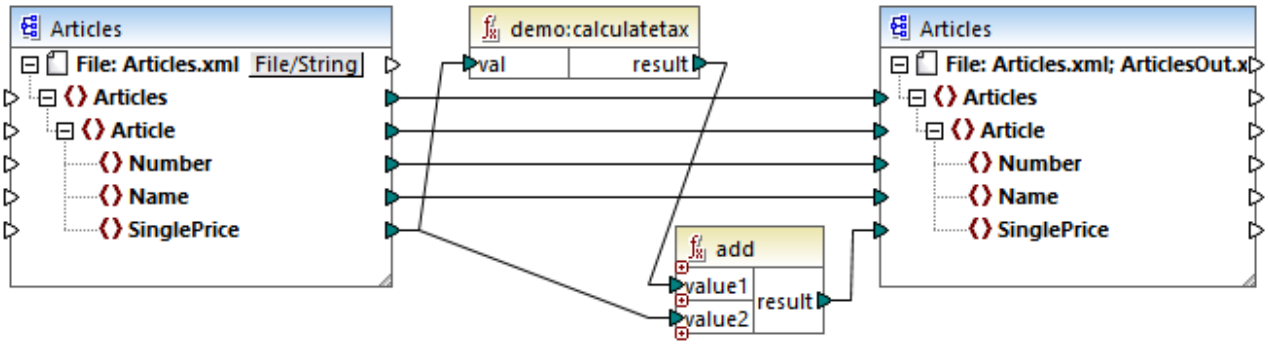
3. Rechercher **<Documents>\Altova\MapForce2024\MapForceExamples\Tutorial\module.xq** et cliquer sur **Ouvrir.** Un message apparaît vous informant qu'une nouvelle bibliothèque a été ajoutée.




La bibliothèque importée et la fonction **demo:calculatetax** sont maintenant visibles dans la fenêtre Bibliothèques.




De plus, le mappage peut maintenant être validé et exécuté sans avertissements. La fonction **demo:calculatetax** illustrée dans l'image ci-dessous provient du module XQuery importé et elle peut être ajoutée au mappage comme toute autre fonction intégrée, voir Ajouter une fonction dans le mappage 468.




*CalculateTax_XQuery.mfd*

## Explication du mappage

Le mappage **CalculateTax_XQuery.mfd** illustré ci-dessus prend en tant qu'entrée un fichier XML qui stocke des articles. Chaque article possède un seul prix, exprimé dans une valeur décimale, par exemple :

```
<Articles>
   <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*

Le mappage produit un fichier XML qui suit les mêmes règles que le fichier XML source. Ainsi, les composants de source et de cible ont tous les deux la même structure dans le mappage. Comme le suggère les connexions de mappage, presque tous les éléments sont mappés de manière claire de la cible à la source ; par exemple, pour chaque **Article** dans la source, il y aura un **Article** dans la cible. Les valeurs de tous les items sont copiés verbatim depuis le XML de source, sauf en ce qui concerne **SinglePrice**. La valeur de **SinglePrice** est calculée à l'aide des deux fonctions :

- La fonction XQuery **demo:calculatetax** calcule le montant de l'impôt en prenant le **SinglePrice** original en tant qu'entrée.
- La fonction intégrée MapForce **add** ajoute le montant de l'impôt au montant **SinglePrice** original et retourne le montant final.

Chose importante, le type de données de l'item **SinglePrice** est `xs:decimal;` ce qui correspond au type de paramètre d'entrée et au type de retour de la fonction XQuery.

La sortie produite par le mappage lorsque vous cliquez sur l'onglet **Sortie** est illustrée ci-dessous. Veuillez noter l'augmentation de 20% appliquée à chaque prix comparé au XML source.

```
<?xml version="1.0" encoding="UTF-8"?>
<Articles xmlns:xsi="http://www.w3.org/2001/XMLSchema-
    <Article>
        <Number>1</Number>
        <Name>T-Shirt</Name>
        <SinglePrice>30</SinglePrice>
    </Article>
    <Article>
        <Number>2</Number>
        <Name>Socks</Name>
        <SinglePrice>2.76</SinglePrice>
    </Article>
    <Article>
        <Number>3</Number>
        <Name>Pants</Name>
        <SinglePrice>40.8</SinglePrice>
    </Article>
    <Article>
        <Number>4</Number>
        <Name>Jacket</Name>
        <SinglePrice>69</SinglePrice>
    </Article>
</Articles>
```

## 6.5.3 Importer des bibliothèques Java et .NET personnalisées

Cette section explique comment importer les fichiers de classe Java compilés et les assemblies .NET DLL (y compris les assemblies .NET 4.0) dans MapForce. Si les bibliothèques importées contiennent des fonctions qui utilisent des types de données de base en tant que paramètres et retournent des types simples, ces fonctions apparaissent dans la fenêtre **Bibliothèques**, et peuvent être utilisées dans des mappages comme dans tout autre fonction disponible dans MapForce. La sortie de mappage des fonctions importées Java et .NET peuvent être consultées dans le volet **Sortie** et les fonctions sont disponibles dans le code généré. Pour en savoir plus sur comment importer les bibliothèques personnalisées, voir les exemples fournis dans Importer Classe Java personnalisée 524 et Importer un .NET DLL Assembly personnalisé 526.

**Important :**

- Afin d'importer des fonctions Java ou .NET, vous avez besoin de classes Java compilées (`.class`) ou des fichiers .NET.dll assembly. L'importation de fichiers `.jar` files ou `.dll` qui ne sont pas un .NET assembly n'est pas pris en charge.

- Les fichiers .NET assembly sont pris en charge lorsque le langage de mappage est défini sur C#. Les .NET assemblies peuvent être écris dans des langages .NET différents de C# (par exemple, C++.NET ou VB.NET), s'ils utilisent uniquement des types de données de base depuis le System Assembly en tant que paramètres et types de retour. Pour les détails, voir Prise en charge de la fonction .NET 522.

- Si vous voulez utiliser des fonctions .NET personnalisées dans l'aperçu de sortie built-in (dans le volet **Sortie**), ces fonctions doivent être compilées pour .NET Framework 4.x ou .NET Standard 2.0.

- Des fichiers de classes Java compilées (`.class`) sont prises en charge lorsque le langage de mappage est défini sur Java. Java Runtime Environment 7 ou plus récent doit être installé sur votre ordinateur. Seuls des types et membres spécifiques sont pris en charge (voir Prise en charge de la fonction Java 521).

- Vous ne pouvez pas définir le langage de mappage sur C++ si le mappage utilise `.class` ou des assemblies .NET DLL importés.

- Vous ne pouvez pas définir le langage de mappage sur XSLT si le mappage utilise des `.class` Java importées ou des assemblies .NET DLL importés (une fonction XSLT personnalisée qui agit en tant qu'un adaptateur devra être rédigée).

- L'importation de fonctions provenant de DLL C++ natifs est limitée et nécessite une approche particulière. Pour plus d'informations, voir Référencer les bibliothèques Java, C# et C++ manuellement 528.

- Toutes les fonctions appelées depuis un mappage MapForce doivent retourner la même valeur à chaque fois que la fonction est appelée avec les même paramètres d'entrée. L'ordre exact et le nombre de fois qu'une fonction est appelée par MapForce ne sont pas définis.

- Dans le cas de Java, les fichiers de classe importés et leurs packages n'ont pas besoin d'être ajoutés à la variable `CLASSPATH` puisque le moteur d'exécution Built-In et le code Java généré ajouteront automatiquement des packages importés respectivement au classpath du moteur Java ou à Ant. Néanmoins, toute dépendance des fichiers de classe importés et des packages ne seront pas gérés automatiquement. C'est pourquoi, si des fichiers de classe Java importés ou des packages dépendent d'autres fichiers de classe, assurez-vous d'ajouter les répertoires de parent de tous les packages dépendants de la variable d'environnement `CLASSPATH`.

## Prise en charge de la fonction Java

Les classes de niveau supérieur, les classes membre statiques et les classes de membre non-statiques sont prises en charge :

- `new <classname>(<arg1>, <arg2>, ...)`
- `<object>.new <member-class>(<arg1>, <arg2>, ...)`

Les fonctions membre et les fonctions statiques sont prises en charge :

- `<function>(<arg1>, <arg2>, ...)`
- `<object>.<method>(<arg1>, ...)`

Les connexions prises en charge entre Schéma XML et les types Java :

| Type Schéma | Type Java |
|---|---|
| `xs:string` | `String` |
| `xs:byte` | `byte` |
| `xs:short` | `short` |
| `xs:int` | `int` |

| Type Schéma | Type Java |
|---|---|
| xs:long | long |
| xs:boolean | booléenne |
| xs:float | float |
| xs:double | double |
| xs:decimal | java.math.BigDecimal |
| xs:integer | java.math.BigInteger |

Des connexions dans les deux sens sont possibles. D'autres types Java (y compris les types array) ne sont pas pris en charge. Les méthodes pour utiliser de tels paramètres ou des valeurs de retour seront ignorées. Les types d'objet sont pris en charge en appelant leur constructeur, ou en tant que valeur de retour d'une méthode. Ils peuvent être mappés à d'autres méthodes Java. Il n'est pas possible de manipuler l'objet en utilisant MapForce.

## Prise en charge de la fonction .NET

Les classes de niveau supérieur et les classes membre sont prises en charge :

- `new <classname>(<arg1>, <arg2>, ...)`

Les fonctions membre et les fonctions statiques sont prises en charge :

- `<function>(<arg1>, <arg2>, ...)`
- `<object>.<method>(<arg1>, ...)`

Les connexions prises en charge entre Schéma XML et les types .NET/C# :

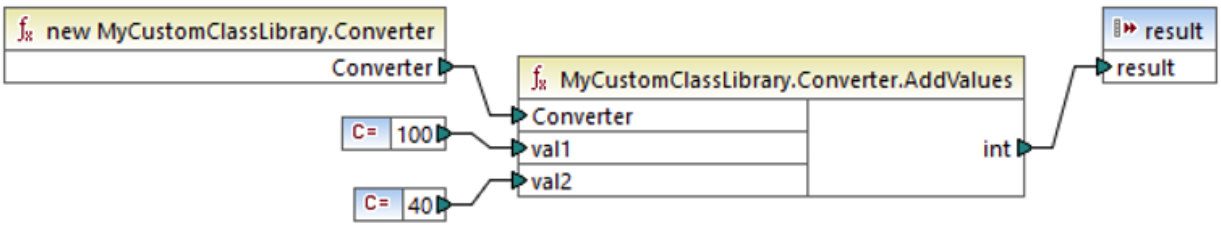
| Type Schéma | Type .NET | Type C# |
|---|---|---|
| xs:string | System.String | string |
| xs:byte | System.SByte | sbyte |
| xs:short | System.Int16 | short |
| xs:int | System.Int32 | int |
| xs:long | System.Int64 | long |
| xs:unsignedByte | System.Byte | byte |
| xs:unsignedShort | System.UInt16 | ushort |
| xs:unsignedInt | System.UInt32 | uint |
| xs:unsignedLong | System.UInt64 | ulong |
| xs:boolean | System.Boolean | bool |

| Type Schéma | Type .NET | Type C# |
|---|---|---|
| xs:float | System.Single | float |
| xs:double | System.Double | double |
| xs:decimal | System.Decimal | decimal |

Des connexions dans les deux sens sont possibles. Les autres types .NET/C# (y compris les types array) ne sont pas pris en charge. Les méthodes pour utiliser de tels paramètres ou des valeurs de retour seront ignorées. Les types d'objet sont pris en charge en appelant leur constructeur, ou en tant que valeur de retour d'une méthode. Ils peuvent être mappés à d'autres méthodes .NET. Il n'est pas possible de manipuler l'objet en utilisant MapForce.

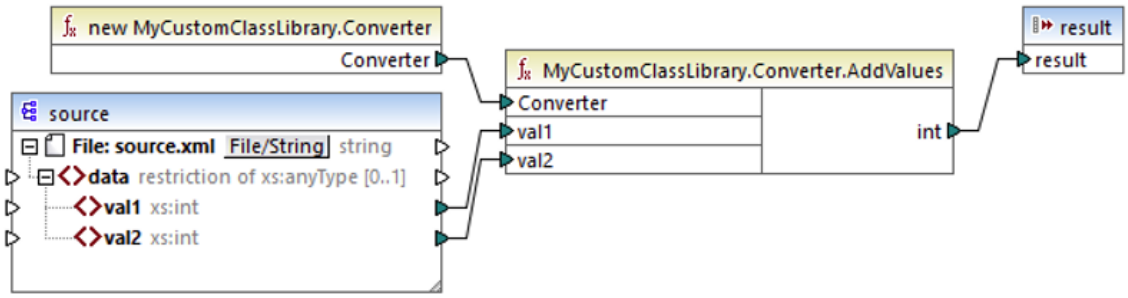
## Problèmes et contournements de types de données

Lorsqu'une fonction dans votre bibliothèque personnalisée attend des types d'entier, connectant des constantes de type `Number` aux arguments de la fonction, ceci peut causer une erreur d'incompatibilité semblable à celle-ci : `No match for MyCustomClassLibrary, Version=1.0.0.0, Culture=neutral, PublicKeyToken=null.MyCustomClassLibrary.Converter.AddValues(MyCustomClassLibrary.Converter , xs:decimal, xs:decimal). Vérifier types d'argument.` Ce problème est spécifique aux constantes de type `Number` uniquement. Un exemple de mappage qui pourrait générer cette erreur est affiché ci-dessous. Dans ce mappage, deux constantes de type `Number` sont connectées aux arguments de type de la fonction `Integer`.




Les contournements possibles sont décrits ci-dessous :

1. Changer le type de constante de **Number** à **All other**. Pour ce faire, double-cliquez sur la barre de titre du composant de la constante.
2. Au lieu d'une constante, utilisez un composant de source (par exemple, un fichier XML) qui fournit des valeurs du type de données attendues par la fonction.




3. Dans votre code externe, créez une fonction wrapper qui accepte une valeur décimale et retourne une valeur d'entier. La solution wrapper peut être importée en tant que bibliothèque séparée. C'est pourquoi vous ne devez pas changer le code source d'origine de la fonction cible pour utiliser cette approche.

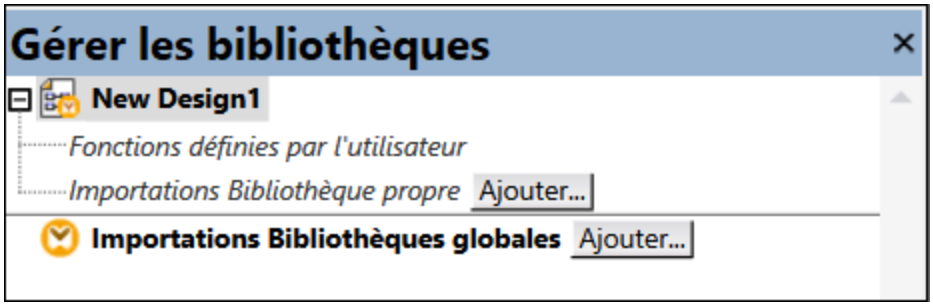
### 6.5.3.1 Exemple : Importer une classe Java personnalisée

Cet exemple illustre comment importer un fichier Java `.class` personnalisé dans MapForce.

**Note :** Java SE 8 Runtime Environment ou une version plus élevée est nécessaire pour compléter cet exemple.

*Importation de Java .class*
Pour ajouter un fichier Java `.class` en tant que bibliothèque MapForce, suivez les étapes suivantes :

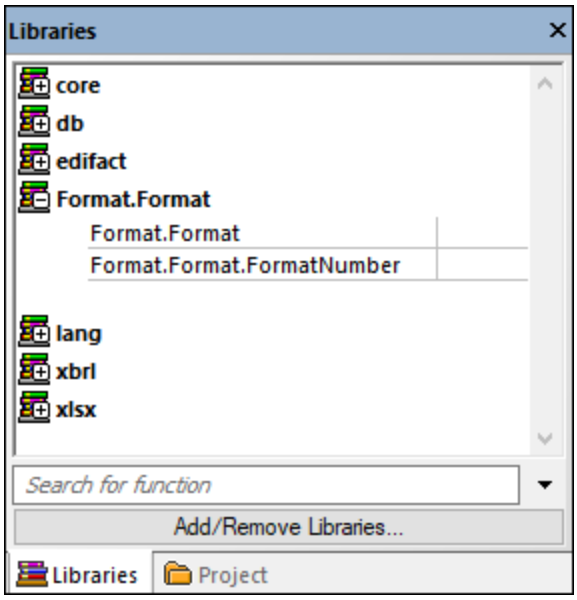
1. Cliquez sur la touche **Ajouter/Supprimer des bibliothèques** en bas de la fenêtre Bibliothèques[26]. La fenêtre **Gérer les Bibliothèques** s'ouvre (*voir la capture d'écran ci-dessous*).




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. Recherchez le fichier suivant : **`<Documents>\Altova\MapForce2024\MapForceExamples\Java\Format\Format.class`**. Un message apparaît vous informant qu'une nouvelle bibliothèque a été ajoutée. La bibliothèque importée est maintenant visible dans la fenêtre **Bibliothèques** (*voir la capture d'écran ci-dessous*).




Si vous ne voyez pas la nouvelle bibliothèque importée dans la fenêtre **Bibliothèques**, veuillez vous assurer que le langage de transformation[22] est configuré sur Java. Pour ajouter la fonction au-dessus du mappage, glissez-la depuis la fenêtre **Bibliothèques** vers la zone de mappage. Pour plus d'informations, voir Ajouter une fonction au mappage[468].

*Sortie de mappage*
Pour réaliser cet sortie de mappage dans MapForce, suivons les étapes suivantes :

1. Ouvrir le mappage suivant : **<Documents>\Altova\MapForce2024\MapForceExamples\Java\FormatNumber.mfd**. Il s'agit d'un mappage complet qui importe d'ores et déjà la bibliothèque `.class` Java mentionnée plus haut.
2. Cliquez sur la touche **Sortie** pour voir le résultat du mappage (*voir la capture d'écran ci-dessous*).

```
 1   Start date,End date,Region,Amount
 2   2008-01-01,2008-01-31,CA,"110.400,00"
 3   2008-01-01,2008-01-31,MA,"75.300,00"
 4   2008-02-01,2008-02-29,CA,"114.300,00"
 5   2008-02-01,2008-02-29,MA,"65.200,00"
 6   2008-03-01,2008-03-31,CA,"134.200,00"
 7   2008-03-01,2008-03-31,MA,"86.100,00"
 8   2008-04-01,2008-04-30,CA,"107.300,00"
 9   2008-04-01,2008-04-30,MA,"112.100,00"
10   2008-05-01,2008-05-31,CA,"114.400,00"
11   2008-05-01,2008-05-31,MA,"93.800,00"
```

*Mappage dans Java*
Pour exécuter le mappage dans Java, suivez les instructions ci-dessous :

1. Cliquez sur **Générer code dans | Java** dans le menu **fichier**.
2. Choisissez un répertoire cible dans lequel le code doit être généré et cliquez sur **OK**.
3. Importer les bibliothèques générées dans votre projet Java et construisez l'application Java. Pour plus d'informations, voir Exemple : Générer et exécuter du Code Java.

## 6.5.3.2 Exemple : Importer un .NET DLL Assembly personnalisé

Cet exemple illustre comme importer dans MapForce un DLL .NET personnalisé créé dans C#. Le code source de cet échantillon est disponible dans le chemin suivant : **<Documents>\Altova\MapForce2024\MapForceExamples\C#\Format**. Le fichier assembly `.dll` qui sera importé dans MapForce se trouve dans le répertoire **..\bin\Debug**. Vous pouvez aussi ouvrir le fichier de solution `.sln` dans Visual Studio et compiler un nouveau fichier `.dll`.

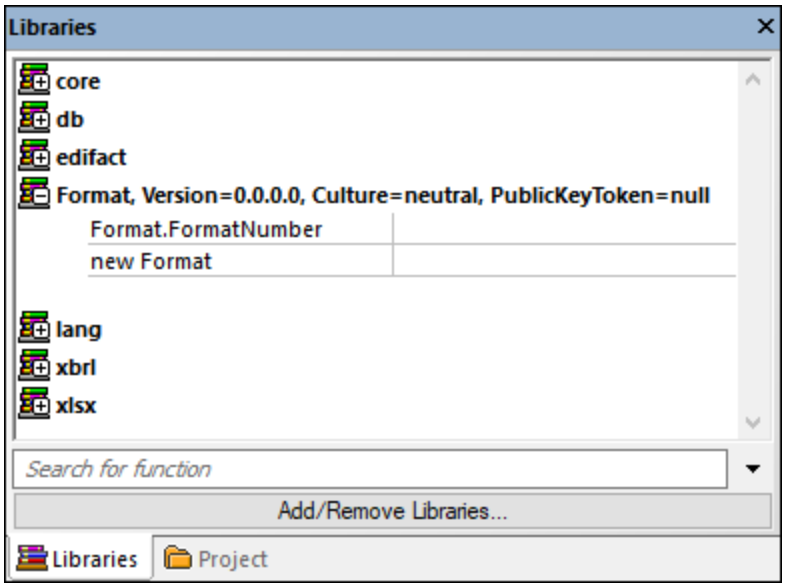
**Note :** si vous voulez utiliser des fonctions .NET personnalisées dans l'aperçu de sortie built-in (dans le volet **Sortie**), ces fonctions doivent être compilées pour .NET Framework 4.x ou .NET Standard 2.0.

*importation de .NET assembly*
Pour importer un fichier .NET assembly, suivez les étapes suivantes :

1. Cliquez sur la touche **Ajouter/Supprimer des bibliothèques** en bas de la fenêtre Bibliothèques 26. 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. Recherchez **Format.dll** dans le répertoire suivant : **...\MapForceExamples\C#\Format\bin\Debug**. Un message apparaît vous informant qu'une nouvelle bibliothèque a été ajoutée. La bibliothèque importée est maintenant visible dans la fenêtre **Bibliothèques**.




Si vous ne voyez pas la nouvelle bibliothèque importée dans la fenêtre **Bibliothèques**, veuillez vous assurer que le langage de transformation[22] est configuré sur C#. Pour ajouter la fonction au-dessus du mappage, glissez-la depuis la fenêtre **Bibliothèques** vers la zone de mappage. Pour plus d'information, voir Ajouter une fonction au mappage[468].

*Sortie de mappage*

Pour réaliser cet sortie de mappage, suivons les étapes suivantes :

1. Ouvrir le fichier **`FormatNumber.mfd`** disponible dans le dossier suivant : **`...\MapForceExamples\C#`**. Il s'agit d'un mappage échantillon qui a une bibliothèque `.dll` importée mentionnée ci-dessus.
2. Cliquez sur la touche **Sortie** pour voir le résultat du mappage (*voir la capture d'écran ci-dessous*).

```
 1  Start date,End date,Region,Amount
 2  2008-01-01,2008-01-31,CA,"110.400,00"
 3  2008-01-01,2008-01-31,MA,"75.300,00"
 4  2008-02-01,2008-02-29,CA,"114.300,00"
 5  2008-02-01,2008-02-29,MA,"65.200,00"
 6  2008-03-01,2008-03-31,CA,"134.200,00"
 7  2008-03-01,2008-03-31,MA,"86.100,00"
 8  2008-04-01,2008-04-30,CA,"107.300,00"
 9  2008-04-01,2008-04-30,MA,"112.100,00"
10  2008-05-01,2008-05-31,CA,"114.400,00"
11  2008-05-01,2008-05-31,MA,"93.800,00"
```

*Mappage dans C#*

Pour exécuter le mappage depuis une application C# personnalisée, suivez les instructions ci-dessous :

1. Cliquez sur **Générer code dans | C#** dans le menu **fichier**.
2. Choisissez un répertoire cible dans lequel le code doit être généré et cliquez sur **OK**.
3. Créez l'application avec Visual Studio, et exécutez l'application de console générée. Pour plus de détails, voir Générer le code C#.

### 6.5.4 Référencer les bibliothèques Java, C# et C++ manuellement

Cette section explique comment référencer des bibliothèques personnalisées dans un fichier `.mff` (fichier de fonction MapForce). Le fichier `.mff` contenant la référence peut ensuite être importé en tant que bibliothèque MapForce. Un fichier `.mff` est un fichier XML dans lequel vous définissez manuellement le lien entre les définitions de classe dans votre code personnalisé et MapForce. Une fois que vous avez créé le fichier `.mff` personnalisé, vous pouvez l'importer dans MapForce, de la même manière que vous importez un fichier .NET DLL ou un fichier de classe Java.

**Important :**

- Si vous voulez utiliser des fonctions .NET personnalisées dans l'aperçu de sortie built-in (dans le volet **Sortie**), ces fonctions doivent être compilées pour .NET Framework 4.x ou .NET Standard 2.0.

- Vous pouvez importer une fonction dans MapForce uniquement si votre type de retour et les paramètres sont de type simple. Pour en savoir plus sur la liste de types de données disponible dans chaque langage, voir Mappage de type de données 534.

- Lorsque vous importez des librairies de fonction depuis les fichiers `.mff` personnalisés, la consultation préalable du mappage directement dans MapForce (en cliquant sur la touche `Sortie`) est limitée. Pour des bibliothèques écrites dans C++, la consultation du mappage dans MapForce n'est pas prise en charge. Pour Java et C#, la consultation est disponible quand votre bibliothèque utilise des types de langage natif, mais n'est pas disponible si votre bibliothèque importe les classes générées d'Altova. Toutefois, vous pouvez générer du code dans le langage spécifique ciblé par votre bibliothèque. Les fonctions personnalisées seront disponibles dans le code généré, vous permettant d'exécuter le mappage depuis le code généré.

- L'ordre exact dans lequel des fonctions sont appelées par le code de mappage généré n'est pas défini. MapForce peut nécessiter de mettre en cache des résultats calculés à réutiliser ou évaluer des expressions dans n'importe quel ordre. Nous vous recommandons donc fortement de n'utiliser que les fonctions personnalisées qui ne présentent pas d'effets secondaires.

- Il est important de distinguer entre des fonctions définies par l'utilisateur et des bibliothèques de fonction personnalisée. Les fonctions définies par l'utilisateur sont créées graphiquement dans un mappage, et elles ne peuvent pas et de doivent pas être enregistrées dans un fichier `.mff`, parce qu'elles sont enregistrées ensemble avec le fichier de mappage `.mfd` dans lequel elles ont été créées. Pour plus d'informations, voir appeler et importer des FDU 492.

- Si vous êtes en train de mettre à jour depuis une version antérieure à 2010, vous devrez mettre à jour les types de données utilisées dans vos fonctions personnalisées. Pour plus de détails, voir Mappage de type de données 534.

Pour plus d'instructions concernant la création et la configuration d'un fichier personnalisé `.mff`, voir Configurer le fichier .mff 529. Les exemples sont fournis dans les rubriques suivantes :

- Référencer C# dans .mff 537
- Référencer C++ dans .mff 538
- Référencer Java dans .mff 541

### 6.5.4.1 Configurer le fichier .mff

Cette rubrique fournit des instructions comment configurer un fichier MapForce Function File (`.mff`). Un fichier `.mff` est un fichier de configuration de format XML qui permet d'importer des fonctions depuis les bibliothèques Java, C# ou C++ personnalisées dans MapForce pour qu'elles apparaissent dans la fenêtre **Bibliothèques**. Un fichier `.mff` est un intermédiaire entre vos bibliothèques personnalisées et MapForce. Le fichier `.mff` doit être configuré pour spécifier i) les interfaces pour les fonctions personnalisées et ii) pour trouver la mise en œuvre dans le code généré.

**Important :**

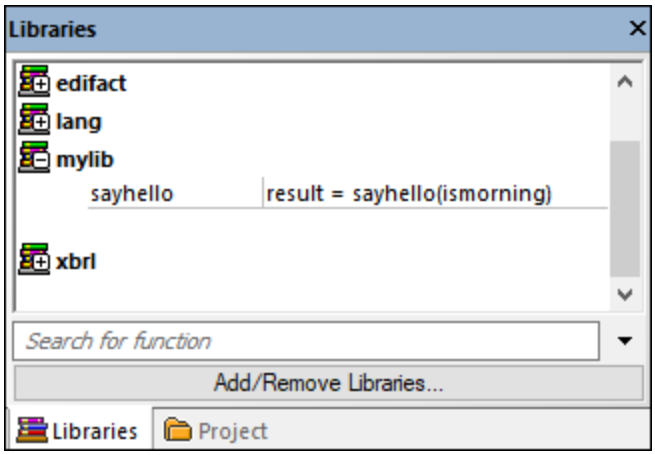
- Les fichiers `*.mff` doivent être valides par rapport au schéma suivant : **`C:\Program Files\MapForceLibraries\mff.xsd`**. Le schéma `mff.xsd` définit la configuration de bibliothèque personnalisée et est pour utilisation interne uniquement. Altova GmbH se réserve le droit de modifier ce format de fichier lors des nouvelles publications.
- Il est possible de ne définir qu'une seule classe C#, C++, ou Java par fichier `.mff`.

*Échantillon .mff pour C#*
L'extrait de code suivant illustre un fichier d'échantillon `.mff` pour C++ :

```
<?xml version="1.0" encoding="UTF-8"?>
<mapping version="9" library="mylib" xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="mff.xsd">
   <implementations>
      <implementation language="cpp">
         <setting name="namespace" value="mylib"/>
         <setting name="class" value="Greetings"/>
         <setting name="path" value="C:\Libraries\cpp"/>
         <setting name="include" value="Greetings.h"/>
         <setting name="source" value="Greetings.cpp"/>
      </implementation>
   </implementations>
   <group name="greetings">
      <component name="sayhello">
         <sources>
            <datapoint name="ismorning" type="xs:boolean"/>
         </sources>
         <targets>
            <datapoint name="result" type="xs:string"/>
         </targets>
         <implementations>
            <implementation language="cpp">
               <function name="SayHello"/>
            </implementation>
         </implementations>
         <description>
            <short>result = sayhello(ismorning)</short>
            <long>Returns "Good morning" or "Good day", depending on the input
parameter.</long>
         </description>
      </component>
```

```
   </group>
</mapping>
```

*Bibliothèque personnalisée importée*
L'image ci-dessous montre à quoi pourrait ressembler un fichier `.mff` personnalisé après l'importation dans MapForce. Veuillez noter que la bibliothèque personnalisée `mylib` apparaît en tant qu'entrée de bibliothèque (triée par ordre alphabétique), contenant la fonction string `sayhello`.




## Étapes de configuration

Pour configurer le fichier `.mff`, suivez les instructions ci-dessous.

*Étape 1. Configurer le nom de bibliothèque*
Vous trouverez le nom de bibliothèque du fichier `.mff` file (*voir ci-dessous*). Par convention, les noms de bibliothèque sont écrits en minuscule dans MapForce ; néanmoins, vous pouvez aussi utiliser des majuscules.

```
<mapping version="9" library="mylib" xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="mff.xsd">
```

Dans l'échantillon ci-dessus, l'entrée qui apparaîtra dans la fenêtre **Bibliothèques** sera appelée **mylib**.

*Étape 2. Configurer les mises en œuvre du langage*
L'élément `<implementations>` est un élément obligatoire qui spécifie quels langages votre bibliothèque doit prendre en charge et qui doit être ajoutée en tant qu'enfant de `<mapping>`, (*voir l'exemple ci-dessous*)

```
<!-- ... -->
<mapping version="9" library="mylib" xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="mff.xsd">
   <implementations>
      <implementation language="cpp">
         <setting name="namespace" value="mylib"/>
         <setting name="class" value="Greetings"/>
         <setting name="path" value="C:\Libraries\cpp"/>
         <setting name="include" value="Greetings.h"/>
         <setting name="source" value="Greetings.cpp"/>
```

```
      </implementation>
   </implementations>
<!-- ... -->
```

Les paramètres dans le cadre de chaque élément `<implementation>` permettent au code généré d'appeler les fonctions spécifiques définies dans Java, C++ ou C#. Un fichier `.mff` peut être rédigé de manière à ce qu'il cible plus d'un langage de programmation. Dans ce cas, chaque langage supplémentaire doit contenir un élément `<implementation>` supplémentaire. Les paramètres spécifiques pour chaque langage de programmation sont discutés ci-dessous.

*Référence de bibliothèque Java*

```
<!-- ... -->
<implementation language="java">
   <setting name="package" value="com.hello.functions"/>
   <setting name="class" value="Greetings"/>
</implementation>
<!-- ... -->
```

Il est important que le code généré puisse trouver votre fichier **`Greetings.class`**. C'est pourquoi vous devez vous assurer que votre classe a été ajoutée au chemin de classe Java.

*Référence de bibliothèque C#*

```
<!-- ... -->
      <implementation language="cs">
         <setting name="namespace" value="MyLibrary" />
         <setting name="class" value="Greetings" />
         <setting name="reference" value="C:
\Libraries\cs\MyLibrary\bin\debug\MyLibrary.dll" />
      </implementation>
<!-- ... -->
```

Pour C#, il est important que l'espace de noms dans le code corresponde à l'espace de noms défini dans le fichier `.mff` (dans le listing du code ci-dessus, l'espace de noms est **`MyLibrary`**). La même chose vaut pour le nom de classe (dans l'extrait de code ci-dessus, le nom de classe est **`Greetings`**). Le troisième paramètre, `reference`, fournit le chemin du `dll` qui doit être lié au code généré.

*Référence de bibliothèque C++*

```
<!-- ... -->
   <implementation language="cpp">
         <setting name="namespace" value="MyLibrary"/>
         <setting name="class" value="Greetings"/>
         <setting name="path" value="C:\Libraries\cpp"/>
         <setting name="include" value="Greetings.h"/>
         <setting name="source" value="Greetings.cpp"/>
   </implementation>
<!-- ... -->
```

En ce qui concerne C++, veuillez noter :

- `namespace` est l'espace de noms dans lequel votre classe `Greetings` sera défini. Il doit être égal à l'attribut `library` dans l'élément `mapping`.
- `path` est le chemin dans lequel se trouvent l'include et les fichiers de source.

- Lors de la génération de code pour un mappage, l'include et les fichiers de source seront copiés dans le répertoire **`targetdir/libraryname`**, qui est défini lors de la sélection de la commande **Fichier | Générer code dans | C++**, et inclus dans le fichier de projet.

Tous les fichiers include que vous fournissez seront inclus dans l'algorithme généré.

*Étape 3. Ajouter un composant*
Dans la fenêtre **Bibliothèques**, chaque fonction apparaît imbriquée sous un groupe de fonctions, par exemple des "string functions". Dans le fichier `.mff`, une fonction correspond à un élément `<component>`. En revanche chaque `<component>` doit être imbriqué sous un élément `<group>`, par exemple :

```
<!-- ... -->
<group name="string functions">
   <component name="sayhello">
   <!-- ... -->
   </component>
</group>
<!-- ... -->
```

Le code affiché ci-dessous définit une fonction d'échantillon (composant) appelée **`sayhello`**.

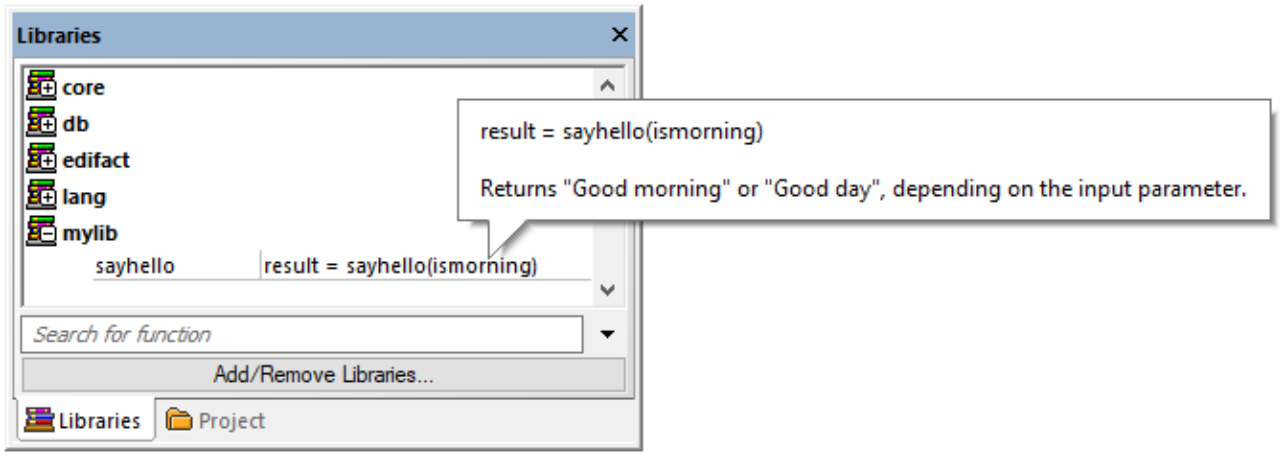
```
<!-- ... -->
<component name="sayhello">
   <sources>
      <datapoint name="ismorning" type="xs:boolean"/>
   </sources>
   <targets>
      <datapoint name="result" type="xs:string"/>
   </targets>
   <implementations>
   <!-- ... -->
   </implementations>
   <description>
     <short>result = sayhello(ismorning)</short>
     <long>Returns "Good morning" or "Good day", depending on the input
parameter.</long>
   </description>
</component>
<!-- ... -->
```

Voici à quoi ressemblerait le composant ci-dessus dans MapForce :


Dans le listing de code ci-dessus, l'élément `<datapoint>` peut être librement défini en tant que paramètre d'entrée ou de sortie d'une fonction (aussi nommé connecteur d'entrée ou de sortie). L'argument `type` de l'élément `<datapoint>` spécifie le type de données du paramètre ou le type de données de la valeur de retour. Un seul point de cible est autorisé pour chaque fonction. Le nombre de points de données source que vous pouvez définir n'est pas limité.

Le type de données de chaque point de données doit être un des types de Schéma XML (par exemple, `xs:string`, `xs:integer`, etc.). Ces types de données doivent correspondre aux types de données des paramètres de fonction que vous avez défini dans votre bibliothèque Java, C++ ou C#. Pour en savoir plus sur le mappage des types de données de schéma XML dans les types de langage, voir Mappage de type de données 534.

Les fonctions sont accompagnées par des descriptions brèves et longues dans la fenêtre **Bibliothèques**. La description brève est toujours affichée à droite du nom de fonction, alors que la description longue est affichée en tant qu'infobulle lorsque vous placez le curseur de la souris sur la description brève (*voir la capture d'écran ci-dessous*).




*Étape 4. Définir les mises en œuvre de langage*
Nous pouvons désormais connecter la fonction dans la fenêtre **Bibliothèques** avec la fonction dans les classes personnalisées Java, C# ou C++. Cela s'effectue par le biais de l'élément `<implementation>`. Une fonction peut avoir plusieurs éléments `<implementation>` - un pour chaque langage de programmation pris en charge. Une fonction peut être appelée `Hello` dans Java ou `SayHello` dans C++. C'est pourquoi vous devez spécifier un nom de fonction séparé pour chaque langage de programmation. Une fonction pour chacun des langages de programmation pourrait ressembler à l'exemple suivant :

```
<!-- ... -->
<component name="sayhello">
<!-- ... -->
   <implementations>
      <implementation language="cs">
         <function name="HelloFunction"/>
      </implementation>
      <implementation language="java">
         <function name="Hello"/>
      </implementation>
      <implementation language="cpp">
         <function name="SayHello"/>
      </implementation>
   </implementations>
<!-- ... -->
</component>
<!-- ... -->
```

La valeur que vous fournissez en tant que nom de fonction doit correspondre à la méthode dans la classe Java, C# ou C++.

### 6.5.4.2 Importer le fichier .mff dans MapForce

Une fois que vous avez créé[529] un fichier `.mff` personnalisé, vous pouvez l'importer dans MapForce comme suit :

1. Cliquez sur la touche **Ajouter/Supprimer des bibliothèques** en bas de la fenêtre Bibliothèques[26]. 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 `.mff` personnalisé et cliquez sur **Ouvrir.**

La bibliothèque importée devient visible dans la fenêtre **Bibliothèques** après avoir défini le langage de mappage sur un langage ciblé par la bibliothèque personnalisée.

Si vous enregistré le fichier `*.mff` dans **`...\Altova\MapForce2024\MapForceLibraries`**, qui est relatif aux **fichiers de programme** (ou au dossier **fichiers de programme (x86)**), la bibliothèque est automatiquement chargée dans la fenêtre **Bibliothèques** lorsque vous démarrez MapForce. Les bibliothèques et leurs fonctions peuvent être activées ou désactivées, en supprimant ou en ajoutant le fichier de bibliothèque respectif (`*.mff`).

### 6.5.4.3 Mappage de type de données

La table ci-dessous recense les types de données pris en charge en tant que types de retour de fonction et types de paramètres lorsque vous créez des fichiers `.mff` personnalisées qui référence des biblitohèques Java, C# et C++. La table contient les types de données natifs et non-natifs. Si vous nécessitez une prise en charge pour des types de données non-natifs comme les types de date, heure et durée Altova, vos bibliothèques Java et C# personnalisées doivent inclure une référence aux bibliothèques Altova. Dans le cas de C++, les bibliothèques Altova doivent toujours être importées. Pour plus d'informations concernant la génération des bibliothèques Altova, voir Générateur de code[935].

| Type de Schéma XML | Type Java | Type C# | Type C++ |
|---|---|---|---|
| anyAtomicType | String | string | string_type |
| anySimpleType | String | string | string_type |
| anyURI | String | string | string_type |
| base64Binary | byte[] | byte[] | altova::mapforce::blob |
| booléen | booléen | bool | bool |
| byte | int | int | int |
| date | com.altova.types.DateTime 1028 | Altova.Types.DateTime 1013 | altova::DateTime 998 |
| dateTime | com.altova.types.DateTime 1028 | Altova.Types.DateTime 1013 | altova::DateTime 998 |
| dayTimeDuration | com.altova.types.Duration 1032 | Altova.Types.Duration 1017 | altova::Duration 1001 |
| decimal | java.math.BigDecimal | decimal | double |
| double | double | double | double |
| duration | com.altova.types.Duration 1032 | Altova.Types.Duration 1017 | altova::Duration 1001 |
| ENTITIES | String | string | string_type |
| ENTITY | String | string | string_type |
| float | double | double | double |
| gDay | com.altova.types.DateTime 1028 | Altova.Types.DateTime 1013 | altova::DateTime 998 |
| gMonth | com.altova.types.DateTime 1028 | Altova.Types.DateTime 1013 | altova::DateTime 998 |
| gMonthDay | com.altova.types.DateTime 1028 | Altova.Types.DateTime 1013 | altova::DateTime 998 |
| gYear | com.altova.types.DateTime 1028 | Altova.Types.DateTime 1013 | altova::DateTime 998 |
| gYearMonth | com.altova.types.DateTime 1028 | Altova.Types.DateTime 1013 | altova::DateTime 998 |
| hexBinary | byte[] | byte[] | altova::mapforce::blob |
| ID | String | string | string_type |

| Type de Schéma XML | Type Java | Type C# | Type C++ |
|---|---|---|---|
| IDRef | String | string | string_type |
| IDREFS | String | string | string_type |
| int | int | int | int |
| integer | java.math.BigInteger | decimal | __int64 |
| language | String | string | string_type |
| long | long | long | __int64 |
| Nom | String | string | string_type |
| NCName | String | string | string_type |
| negativeInteger | java.math.BigInteger | decimal | __int64 |
| NMTOKEN | String | string | string_type |
| NMTOKENS | String | string | string_type |
| nonNegativeInteger | java.math.BigInteger | decimal | unsigned __int64 |
| nonPositiveInteger | java.math.BigInteger | decimal | __int64 |
| normalizedString | String | string | string_type |
| NOTATION | String | string | string_type |
| positiveInteger | java.math.BigInteger | decimal | unsigned __int64 |
| QName | javax.xml.namespace.QName | Altova.Types.QName | altova::QName |
| short | int | int | int |
| string | String | string | string_type |
| heure | com.altova.types.DateTime 1028 | Altova.Types.DateTime 1013 | altova::DateTime 998 |
| token | String | string | string_type |
| unsignedByte | long | ulong | unsigned __int64 |
| unsignedInt | long | ulong | unsigned __int64 |
| unsignedLong | java.math.BigInteger | ulong | unsigned __int64 |
| unsignedShort | long | ulong | unsigned __int64 |
| untypedAtomic | String | string | string_type |

| Type de Schéma XML | Type Java | Type C# | Type C++ |
|---|---|---|---|
| yearMonthDuration | com.altova.types.Duration 1032 | Altova.Types.Duration 1017 | altova::Duration 1001 |

## 6.5.4.4 Référencer la bibliothèque C# dans .mff

Cet exemple vous montre comment créer une bibliothèque C# d'échantillon et la référencer à un fichier MapForce Function File (.mff). Le fichier .mff peut ensuite être importé en tant que bibliothèque MapForce. Référencer une bibliothèque C# au format .mff ne constitue qu'un des moyens pour importer des bibliothèques C# dans MapForce. Une alternative plus simple est d'importer des .NET assemblies directement. Pour plus d'informations, voir Exemple : Importer un .NET DLL Assembly personnalisé 526.

### Étapes de configuration

Pour référencer une bibliothèque C# dans un fichier .mff, suivez les instructions ci-dessous.

**Note :** si vous voulez utiliser des fonctions .NET personnalisées dans l'aperçu de sortie built-in (dans le volet **Sortie**), ces fonctions doivent être compilées pour .NET Framework 4.x ou .NET Standard 2.0.

*Étape 1 : Créer une nouvelle bibliothèque de classe dans VS*

Créer un nouveau projet de bibliothèque de classe dans Visual Studio. Veuillez noter que la fonction a été définie en tant que public static.

```
namespace MyLibrary
{
    public class Greetings
    {
        public static string SayHello(bool isMorning)
        {
            if (isMorning)
                return "Good morning!";
            return "Good Day!";
        }
    }
}
```

*Étape 2 : Ajouter une référence à Altova.dll*

Si vous avez besoin de types de Schéma XML spéciaux (tels date et durée), ajoutez une référence depuis votre projet Visual Studio dans la bibliothèque Altova.dll. Pour obtenir cette bibliothèque, générez du code C# provenant d'un mappage sans fonctions personnalisées. Le fichier Altova.dll se trouvera dans le répertoire **..\Altova\bin\debug** relatif au répertoire dans lequel le code a été généré. Pour ajouter la référence à Altova.dll dans Visual Studio, cliquez sur **Ajouter référence** dans le menu du **Projet** et recherchez Altova.dll. Puis, ajoutez la ligne **utilisant Altova.Types;** à votre code. Pour toute information sur le mappage des types de données de schéma XML dans les types C#, voir Mappage de type de données 534.

*Étape 3 : Crééz votre projet VS*

Créer votre projet Visual Studio. Le fichier MyLibrary.dll est généré dans votre répertoire de sortie de projet.

*Étape 4.Créer un .mff et référencez votre bibliothèque C#*
En utilisant un éditeur XML, créer un nouveau fichier `.mff` et valider le par rapport au schéma suivant : **`C:\Program Files\MapForceLibraries\mff.xsd`**. Assurez-vous que toutes les références sous **`implementation language="cs"`** pointent vers les membres C# et les chemins corrects créés précédemment. La ligne **`function name="SayHello"`** doit référer au nom de la fonction exactement tel qu'il a été défini dans C#. Pour les détails, voir Configurer le fichier .mff 529.

```
<?xml version="1.0" encoding="UTF-8"?>
<mapping version="9" library="mylib" xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="mff.xsd">
   <implementations>
      <implementation language="cs">
         <setting name="namespace" value="MyLibrary" />
         <setting name="class" value="Greetings" />
         <setting name="reference" value="C:
\Libraries\cs\MyLibrary\bin\debug\MyLibrary.dll" />
      </implementation>
   </implementations>
   <group name="string functions">
      <component name="sayhello">
         <sources>
            <datapoint name="ismorning" type="xs:boolean"/>
         </sources>
         <targets>
            <datapoint name="result" type="xs:string"/>
         </targets>
         <implementations>
            <implementation language="cs">
               <function name="SayHello"/>
            </implementation>
         </implementations>
         <description>
            <short>result = sayhello(ismorning)</short>
            <long>Returns "Good morning" or "Good day", depending on the input
parameter.</long>
         </description>
      </component>
   </group>
</mapping>
```

*Étape 5 : Importer le fichier .mff en tant que bibliothèque*
Maintenant que notre bibliothèque personnalisée est référencée dans le fichier `.mff`, vous pouvez importer le fichier `.mff` dans MapForce en tant que bibliothèque. Pour plus d'informations, voir Importer les bibliothèques .mff 534.

## 6.5.4.5 Référencer C++ dans .mff

Cet exemple vous montre comment créer une bibliothèque C# d'échantillon et la référencer à un fichier MapForce Function File (`.mff`). Le fichier `.mff` peut ensuite être importé en tant que bibliothèque MapForce.

## Étapes de configuration

Pour référencer une bibliothèque C++ dans un fichier `.mff`, suivez les instructions ci-dessous.

*Étape 1 : Créer un fichier d'en-tête*

Créer un fichier d'en-tête (`.h`) pour votre bibliothèque de classe. La liste de code suivante illustre un fichier d'échantillon d'en-tête appelé **`Greetings.h`**.

```
#ifndef MYLIBRARY_GREETINGS_H_INCLUDED
#define MYLIBRARY_GREETINGS_H_INCLUDED

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000

using namespace altova;

namespace mylib {

class ALTOVA_DECLSPECIFIER Greetings
{
   public:
    static string_type SayHello(bool isMorning);
};

} // namespace mylib

#endif // MYLIBRARY_GREETINGS_H_INCLUDED
```

Veuillez noter que la fonction a été déclarée comme étant statique, et que l'espace de noms `altova` est importé. Ne pas oublier d'écrire `ALTOVA_DECLSPECIFIER` devant le nom de classe : cela garantit que vos classes compileront correctement, que vous utilisiez des liens dynamiques ou statiques dans le code généré.

*Étape 2 : Créer un fichier .cpp*

Créer un fichier `.cpp` portant le même nom que le fichier d'en-tête. Le fichier `.cpp` doit se trouver dans le même répertoire que le fichier `.h`. La liste de code suivante illustre un fichier d'échantillon `.cpp` appelé **`Greetings.cpp`** qui contient le fichier **`Greetings.h`** créé précédemment :

```
#include "StdAfx.h"
#include "../Altova/Altova.h"
#include "../Altova/AltovaException.h"
#include "../Altova/SchemaTypes.h"

#include "Greetings.h"

namespace mylib {

   string_type Greetings::SayHello(bool isMorning)
   {
      if( isMorning )
         return _T("Good morning!");
      return _T("Good day!");
   }
}
```

Veuillez noter les lignes qui importent le **StdAfx.h** et plusieurs bibliothèques Altova. Ces lignes ne doivent pas être modifiées. Si les chemins menant aux bibliothèques Altova sont correctes dans le code généré, ces chemins pointeront vers les fichiers respectifs. Contrairement à Java ou C#, vous ne devez pas compiler vos fichiers source C++. Ils seront copiés dans le code généré et seront compilés avec le reste du code de mappage généré.

*Étape 3 : Créer un .mff et référencer votre bibliothèque C++*
En utilisant un éditeur XML, créer un nouveau fichier `.mff` et valider le par rapport au schéma suivant : **C:\Program Files\MapForceLibraries\mff.xsd**. Veuillez vous assurer que les espaces de noms, noms de fonction ainsi que les types de données définis ici correspondent à ceux contenus dans le code C++, comme décrit dans Configurer le fichier .mff 529. Pour information sur la prise en charge des types de données, voir Mappage de type de données 534.

```
<?xml version="1.0" encoding="UTF-8"?>
<mapping version="9" library="mylib" xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="mff.xsd">
   <implementations>
      <implementation language="cpp">
         <setting name="namespace" value="mylib"/>
         <setting name="class" value="Greetings"/>
         <setting name="path" value="C:\Libraries\cpp"/>
         <setting name="include" value="Greetings.h"/>
         <setting name="source" value="Greetings.cpp"/>
      </implementation>
   </implementations>
   <group name="greetings">
      <component name="sayhello">
         <sources>
            <datapoint name="ismorning" type="xs:boolean"/>
         </sources>
         <targets>
            <datapoint name="result" type="xs:string"/>
         </targets>
         <implementations>
            <implementation language="cpp">
               <function name="SayHello"/>
            </implementation>
         </implementations>
         <description>
            <short>result = sayhello(ismorning)</short>
            <long>Returns "Good morning" or "Good day", depending on the input
parameter.</long>
         </description>
      </component>
   </group>
</mapping>
```

*Étape 4 : Importer le fichier .mff en tant que bibliothèque*
Maintenant que notre bibliothèque personnalisée est référencée dans le fichier `.mff`, vous pouvez importer le fichier `.mff` dans MapForce en tant que bibliothèque. Pour plus d'information, voir Importer le fichier .mff 534.

*Erreurs de compilation C++*
Afin de pouvoir exécuter des mappages qui utilisent des bibliothèques C++ natives, vous devrez générer du code C++ et exécuter le mappage depuis votre code C++ ou votre application, comme décrit dans Générer code C++. S vous recevez une erreur de compliation dans **`#import "msado15.dll" rename("EOF", "EndOfFile")`**, modifiez les propriétés de projet pour inclure une référence à **`msado15.dll`** dans **`C:\Program Files\Common Files\System\ADO`**.

## 6.5.4.6 Référencer Java dans .mff

Cet exemple vous montre comment créer une bibliothèque Java library d'échantillon et la référencer à un fichier MapForce Function (`.mff`). Le fichier `.mff` peut ensuite être importé en tant que bibliothèque MapForce. Référencer une bibliothèque Java dans un fichier `.mff` ne constitue qu'un des moyens pour importer des bibliothèques Java dans MapForce. Une alternative plus simple est d'importer des fichiers de `.class` Java directement. Pour plus d'informations, voir Exemple : Importer une classe Java personnalisée 524.

### Étapes de configuration

Pour référencer une bibliothèque C# dans un fichier `.mff`, suivez les instructions ci-dessous.

*Étape 1. Créer un nouveau projet*
Créer un nouveau projet Java dans votre environnement de développement de prédilection (par exemple, Eclipse).

*Étape 2. Ajouter le package com.mylib*
Ajouter au projet un nouveau package appelé **`mylib`** qui consiste en une classe appelée **`Greetings`**. Dans l'extrait de code ci-dessous, veuillez noter que la fonction `SayHello` a été définie en tant que `public static`.

```
package com.mylib;

public class Greetings {

   public static String SayHello ( boolean isMorning ) {
      if( isMorning )
         return "Good Morning!";
      return "Good Day!";
   }

}
```

*Étape 3. Importer les com.altova.types*
En option, si votre projet nécessite une prise en charge pour des types de schéma spéciaux comme, par exemple, la date, l'heure et la durée, importer le package `com.altova.types`. Pour obtenir ce package, générer du code Java depuis un mappage sans fonctions personnalisées : **`importer com.altova.types.*;`**.

*Étape 4. Compilez votre bibliothèque personnalisée*
Compilez votre bibliothèque personnalisée dans un fichier de classe et l'ajouter au classpath Java.

*Étape 5. Créez un .mff et référencez votre bibliothèque Java library*
En utilisant un éditeur XML, créer un nouveau fichier `.mff` et valider le par rapport au schéma suivant : **`C:\Program Files\MapForceLibraries\mff.xsd`**. Veuillez vous assurer que toutes les références sous

**`implementation language="java"`** pointent vers les membres Java corrects créés précédemment. Aussi, la ligne **`function name="SayHello"`** doit référer au nom de la fonction exactement tel qu'il a été défini dans Java. Pour les plus détails, voir Configurer le fichier .mff 529.

```
<?xml version="1.0" encoding="UTF-8"?>
<mapping version="9" library="custom" xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="mff.xsd">
   <implementations>
      <implementation language="java">
         <setting name="package" value="com.mylib"/>
         <setting name="class" value="Greetings"/>
      </implementation>
   </implementations>
   <group name="greetings">
      <component name="sayhello">
         <sources>
            <datapoint name="ismorning" type="xs:boolean"/>
         </sources>
         <targets>
            <datapoint name="result" type="xs:string"/>
         </targets>
         <implementations>
            <implementation language="java">
               <function name="SayHello"/>
            </implementation>
         </implementations>
         <description>
            <short>result = sayhello(ismorning)</short>
            <long>Returns "Good morning" or "Good day", depending on the input
parameter.</long>
         </description>
      </component>
   </group>
</mapping>
```

*Étape 6. Importer le fichier .mff en tant que bibliothèque*

Maintenant que notre bibliothèque personnalisée est référencée dans le fichier `.mff`, vous pouvez importer le fichier `.mff` dans MapForce en tant que bibliothèque. Pour plus d'information, voir Importer les Bibliothèques .mff 534.

# 6.6 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** des fonctions **match-pattern** 692 et **tokenize-regexp** 640
- Pour filtrer les nœuds dans lesquels une fonction de nœud doit s'appliquer. Pour plus d'informations, voir Appliquer des fonctions de nœud et les défauts par condition 481.

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".

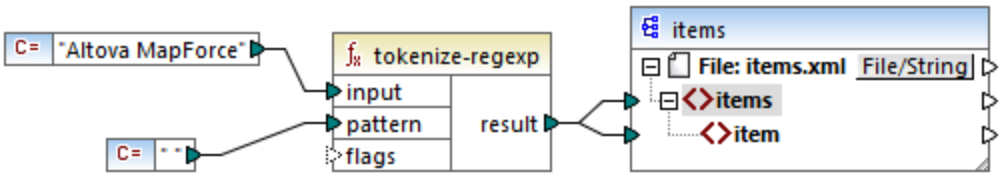
**Note :** Lors de la génération de code C++, C# ou Java, les fonctions avancées de la syntaxe d'expression régulière peuvent différer légèrement. Voir la documentation regex de chaque langage pour plus d'informations.

## 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[635]. 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).

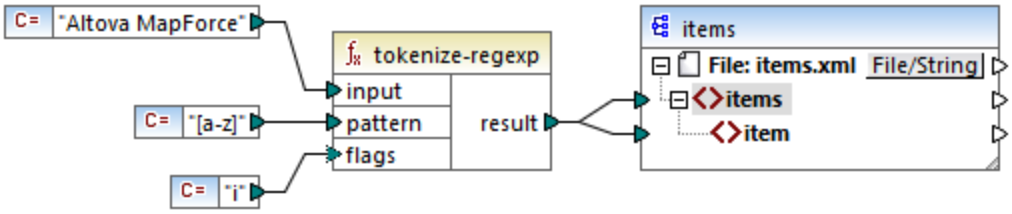
## Alternance

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".<br><br>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.<br><br>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.<br><br>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.<br>  <br> |
| **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**.<br><br>**Note:** Les caractères d'espace blanc se trouvant dans une classe de caractère ne sont pas supprimés, par exemple, `[#x20]`. |

# 6.7 Référence des bibliothèques de fonctions

Cette section de référence décrit les fonctions built-in de MapForce disponibles dans la fenêtre Bibliothèques[26]. 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[22].

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*

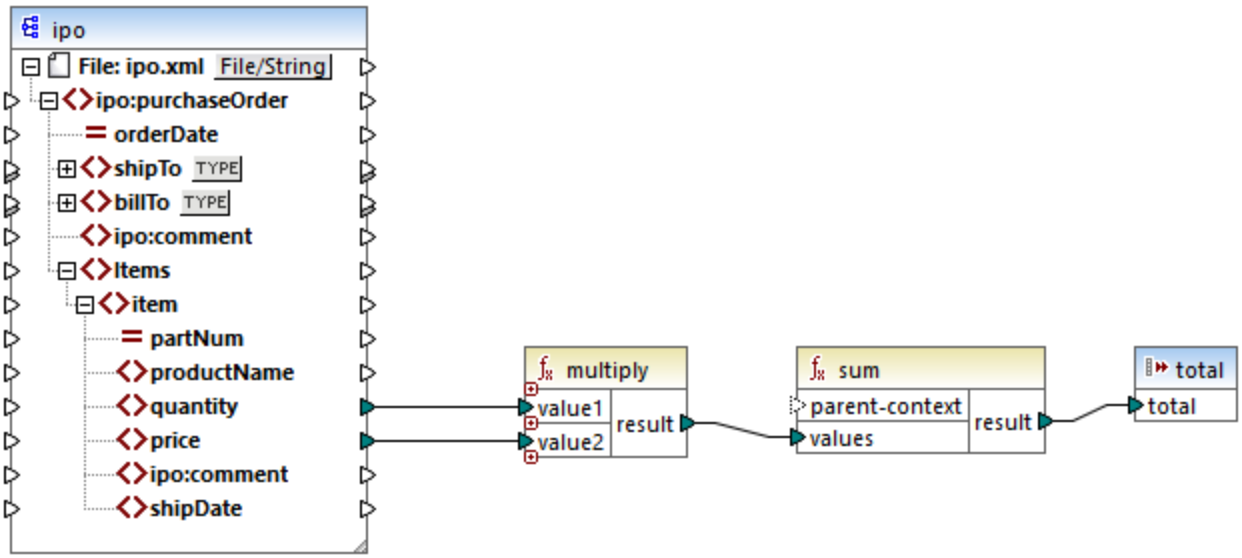
- **generate-id**, **system-property**: XSLT 1.0, XSLT 2.0, et XSLT 3.0.
- **current**, **document**, **element-available**, **function-available**, **unparsed-entity-uri**: XSLT 1.0.

## 6.7.1 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 810.
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**.

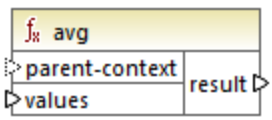



*SimpleTotal.mfd*

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.

## 6.7.1.1 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

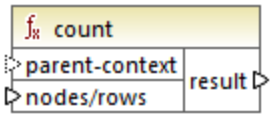
| Argument | Description |
|---|---|
| **parent-context** | Argument optionnel. Fournit le contexte parent. Voir aussi Exemple : Changer le contexte de Parent 810. |
| **valeurs** | Cet argument doit être connecté à un item de source qui fournit les données actuelles. Veuillez noter que la valeur d'argument fournie doit être numérique. |

### Exemple

Voir Exemple : Regrouper les enregistrements par clé 605.

## 6.7.1.2 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

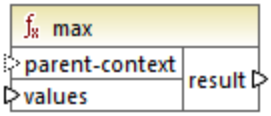
| Argument | Description |
|---|---|
| **parent-context** | Argument optionnel. Fournit le contexte parent. Voir aussi Exemple : Changer le contexte de Parent 810.<br><br>`parent-context` est un argument optionnel dans certaines fonctions d'agrégation core MapForce (comme dans **`min`**, **`max`**, **`avg`**, **`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 810, Exemple : Compter les lignes de table de base de données 421.

## 6.7.1.3 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

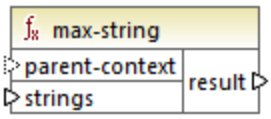
| Argument | Description |
|---|---|
| **parent-context** | Argument optionnel. Fournit le contexte parent. Voir aussi Exemple : Changer le contexte de Parent 810.<br><br>`parent-context` est un argument optionnel dans certaines fonctions d'agrégation core MapForce (comme dans **`min`**, **`max`**, **`avg`**, **`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 552. |

### Exemple

Voir Exemple: Regrouper les enregistrements par clé. 605

## 6.7.1.4 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

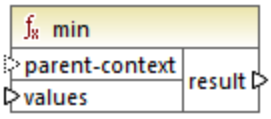
Built-in, C++, C#, Java, XQuery, XSLT 2.0, XSLT 3.0.

### Paramètres

| Argument | Description |
|---|---|
| **parent-context** | Argument optionnel. Fournit le contexte parent. Voir aussi Exemple : Changer le contexte de Parent 810.<br><br>`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`. |

## 6.7.1.5 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 810.<br><br>`parent-context` est un argument optionnel dans certaines fonctions d'agrégation core |

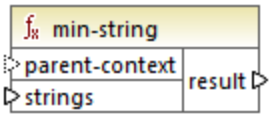
| Argument | Description |
|---|---|
| | MapForce (comme dans **min**, **max**, **avg**, **count**). Dans un composant de source qui possède plusieurs séquences hiérarchiques, le contexte parent détermine l'ensemble de nœuds dans lequel la fonction doit fonctionner. |
| **valeurs** | Cet argument doit 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 554. |

### Exemple

Voir Exemple : Regrouper les enregistrements par clé 605.

## 6.7.1.6 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

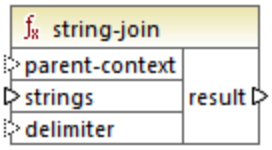
Built-in, C++, C#, Java, XQuery, XSLT 2.0, XSLT 3.0.

### Paramètres

| Argument | Description |
|---|---|
| **parent-context** | Argument optionnel. Fournit le contexte parent. Voir aussi Exemple : Changer le contexte de Parent 810.<br><br>`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`. |

## 6.7.1.7 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

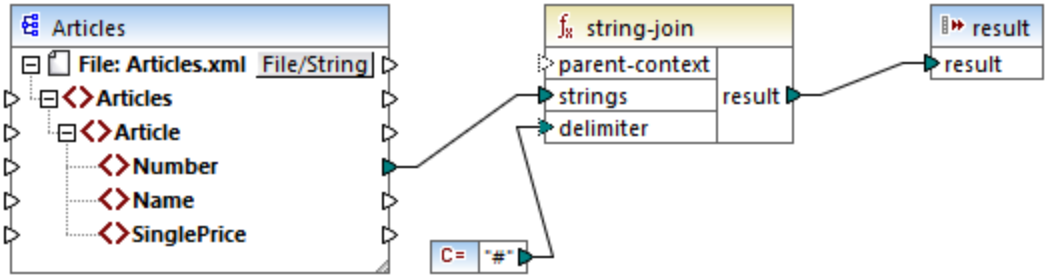
Built-in, C++, C#, Java, XQuery, XSLT 2.0, XSLT 3.0.

### Paramètres

| Argument | Description |
|---|---|
| **parent-context** | Argument optionnel. Fournit le contexte parent. Voir aussi Exemple : Changer le contexte de Parent 810.<br><br>`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

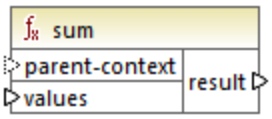
Dans l'exemple ci-dessous, le fichier XML de source contient quatre items **Article**, avec les nombres suivants : 1, 2, 3 et 4.



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**.

### 6.7.1.8 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 810.<br><br>`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

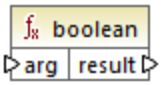
Voir Exemple : Additionner les valeurs de nœud 512.

## 6.7.2 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 565 pour forcer une comparaison numérique.

## 6.7.2.1 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[444]. 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

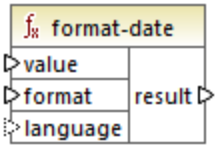
Built-in, C++, C#, Java, XQuery, XSLT 1.0, XSLT 2.0, XSLT 3.0.

### Paramètre

| Argument | Description |
|---|---|
| **arg** | Argument obligatoire. Fournit la valeur à convertir. |

## 6.7.2.2 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.

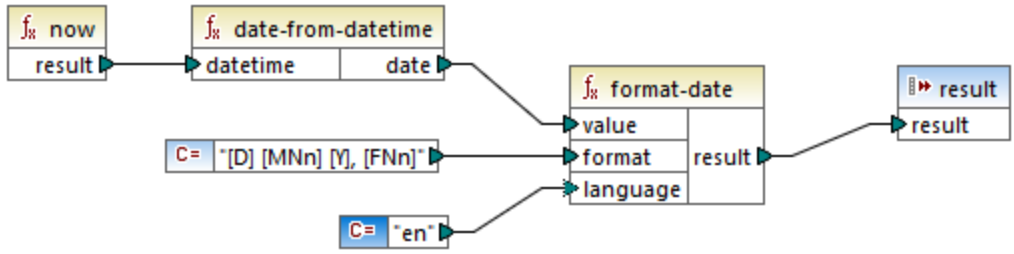
### Paramètres

| 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[558]. |

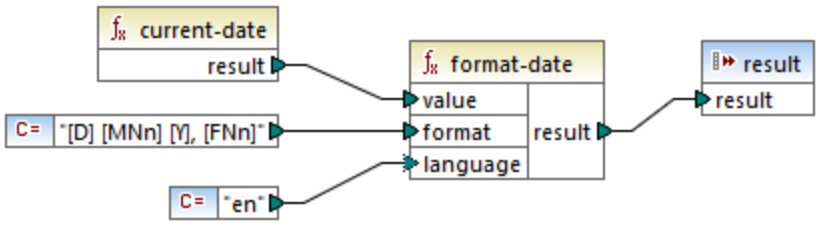
| Argument | Description |
|---|---|
| **language** | Argument optionnel. Lorsqu'il est fourni, le nom du mois est le jour de la semaine sont retournés dans un langage spécifique. Valeurs valides :<br><br>**de** Allemand<br>**en (défaut)** Anglais<br>**es** Espagnol<br>**fr** Français<br>**ja** Japonais |

### 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**.

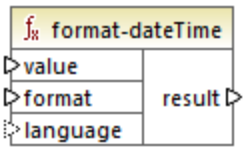



Veuillez noter que le mappage ci-dessus est conçu pour les langages de transformation Built-in, C++, C# ou Java. Dans XSLT 2.0, le même résultat peut être obtenu par le mappage suivant :




## 6.7.2.3 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 :<br>**de** Allemand<br>**en (défaut)** Anglais<br>**es** Espagnol<br>**fr** Français<br>**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 44 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.

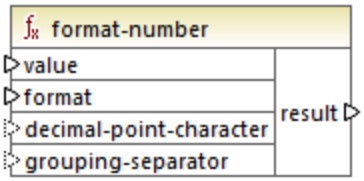
### Exemples

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 |

## 6.7.2.4 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[44] 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 |

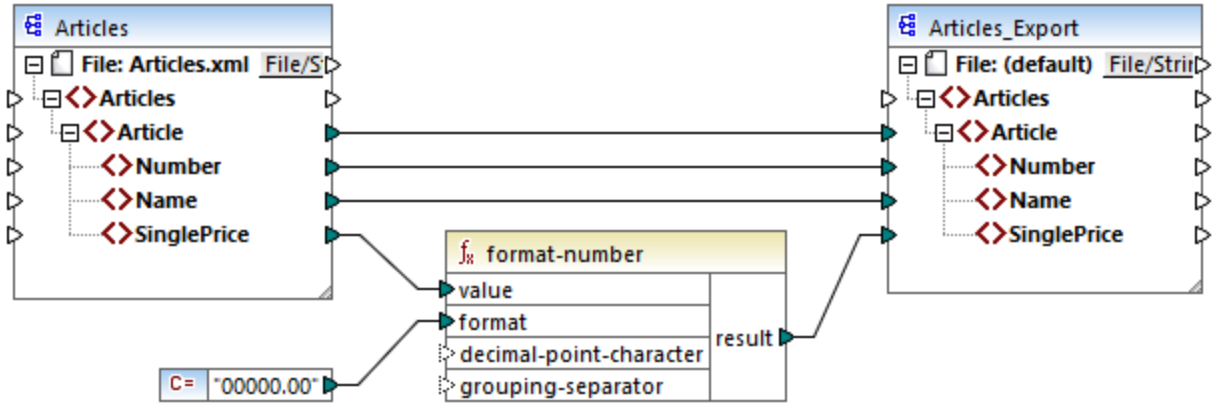
## Exemple

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*

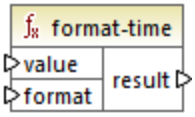
Par conséquent, les valeurs dans la cible sont devenues :

```
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**.

## 6.7.2.5 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

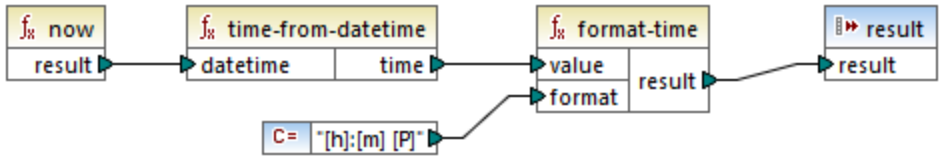
| Argument | Description |
|---|---|
| **value** | Argument obligatoire. Fournit la valeur `xs:time` à formater. |

| 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 558. |

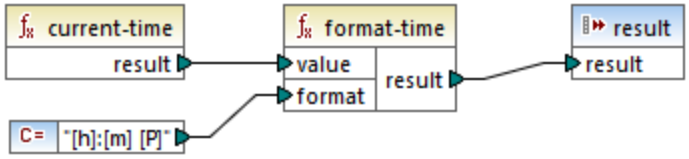
### Exemple

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."

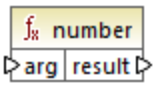



Veuillez noter que le mappage ci-dessus est conçu pour les langages de transformation Built-in, C++, C# ou Java. Dans XSLT 2.0, le même résultat peut être obtenu par le mappage suivant :




## 6.7.2.6 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

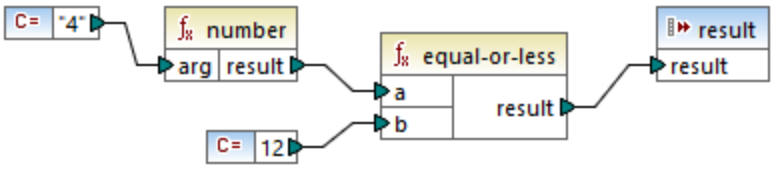
Built-in, C++, C#, Java, XQuery, XSLT 1.0, XSLT 2.0, XSLT 3.0.

### Paramètres

| Argument | Description |
|---|---|
| **arg** | Argument obligatoire. Fournit la valeur à convertir. |

### Exemple

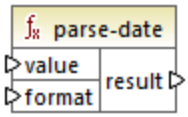
Dans l'exemple ci-dessous, la première constante est de type `string` et elle contient le string "4". La seconde constante contient la constante numérique 12. Pour que les deux valeurs puissent être comparées en tant que nombres, les types doivent s'accorder.




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".

## 6.7.2.7 parse-date

Convertit un string en une date. Cette fonction utilise la fonction parse-dateTime 567 en tant que base tout en ignorant le composant de temps. Le résultat est de type `xs:date`.




### Langages

Built-in, C++, C#, Java.

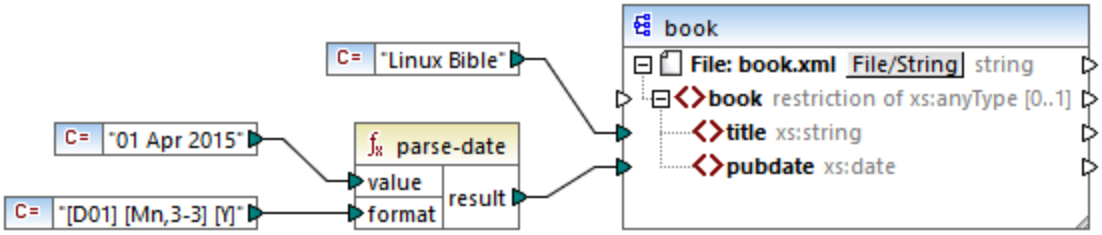
### Paramètres

| Argument | Description |
|---|---|
| **value** | Argument obligatoire. Fournit la valeur string à convertir. |
| **format** | Argument obligatoire. Fournit un string de format. Cet argument est utilisé de la même manière que l'argument **format** dans la fonction parse-dateTime 567. |

### Exemple

Le mappage ci-dessous parse le string **"01 Apr 2015"**, le convertit en une date et écrit le résultat en un item de cible (**pubdate**) de type `xs:date`. Cela a été obtenu en utilisant le format **[D01] [MNn,3-3] [Y]**, où :

- **[D01]** est la date du mois, exprimé en deux chiffres
- **[MNn,3-3]** est le nom du mois, avec une largeur minimum et maximum de 3 caractères
- **[Y]** est l'année

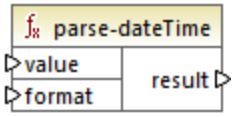



Le résultat est le suivant (excluant les déclaration XML et d'espace de noms) :

```
<book>
   <title>Linux Bible</title>
   <pubdate>2015-04-01</pubdate>
</book>
```

## 6.7.2.8 parse-dateTime

Convertit une valeur date/heure exprimée en tant que string dans une valeur de type `xs:dateTime`.




### Langages

Built-in, C++, C#, Java.

### Paramètres

| Argument | Description |
|---|---|
| **value** | La valeur string à convertir. |
| **format** | Spécifie le masque de format à appliquer sur la **value**. |

## Remarques

Un masque de format peut consister dans les composants suivants :

| Composant | echo Hello, World! | Présentation par défaut |
|---|---|---|
| **Y** | année (valeur absolue) | quatre chiffres (2010) |
| **M** | mois de l'année | 1-12 |
| **D** | jour du mois | 1-31 |
| **d** | jour de l'année | 1-366 |
| **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 |

Certains des composants ci-dessus prennent des modificateurs (par exemple, ils peuvent être utilisés pour interpréter une date soit en tant que chiffre unique, soit en tant que deux chiffres) :

| Modificateur | echo Hello, World! | Exemple |
|---|---|---|
| **1** | format numérique décimal sans zéros au début : 1, 2, 3, ... | 1, 2, 3 |
| **01** | format numérique, deux chiffres : 01, 02, 03, ... | 01, 02, 03 |
| **N** | nom du composant, casse majuscule | FEBRUARY, MARCH |
| **n** | nom du composant, casse minuscule | february, march |
| **Nn** | nom du composant, casse de titre | February, March |

**Note : Les modificateurs** *N*, *n*, et*Nn* ne prennent en charge que le composant *M* (mois).

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 :

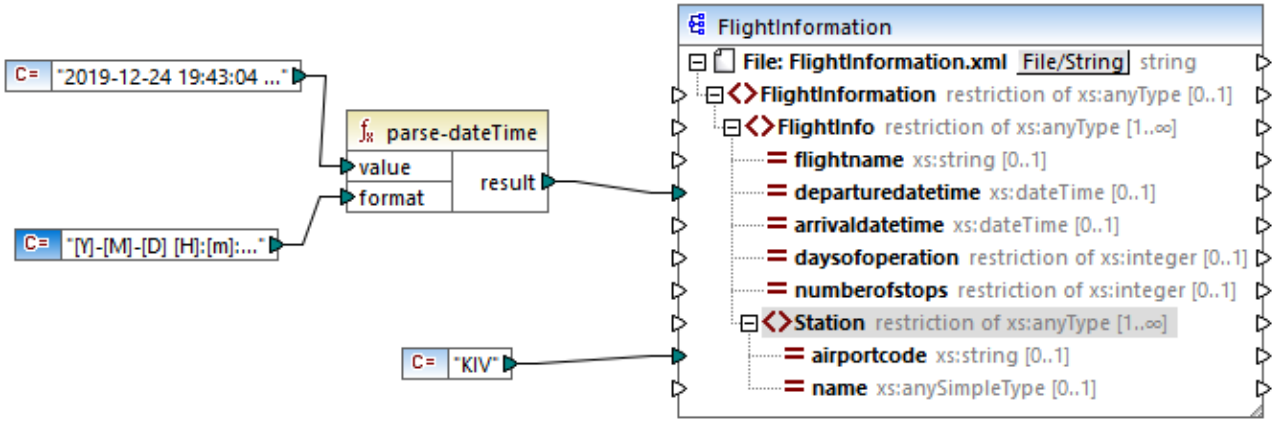
- **`[D,2]`** est le jour du mois, avec des zéros en début de ligne (deux chiffres).
- **`[MNn,3-3]`** est le nom du mois, écrit avec trois lettres, par ex. *Jan*, *Feb*, *Mar*, etc.

La table ci-dessous recense quelques exemples de format :

| Valeur | Format | Résultat |
|---|---|---|
| 21-03-2002 16:21:12.492 GMT+02:00 | `[D]-[M]-[Y] [H]:[m]:[s].[f] [z]` | 2002-03-21T16:21:12.492+02:00 |
| 315 2004 +01:00 | `[d] [Y] [Z]` | 2004-11-10T00:00:00+01:00 |
| 1.December.10 03:2:39 p.m. +01:00 | `[D].[MNn].[Y,2-2] [h]:[m]:[s] [P] [Z]` | 2010-12-01T15:02:39+01:00 |
| 20110620 | `[Y,4-4][M,2-2][D,2-2]` | 2011-06-20T00:00:00 |

## Exemple

Dans le mappage ci-dessous, la valeur string **`2019-12-24 19:43:04 +02:00`** est convertie dans son équivalent `dateTime` en appliquant le masque de format **`[Y]-[M]-[D] [H]:[m]:[s] [Z]`**.

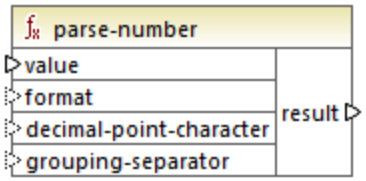



Le résultat est le suivant (excluant les déclaration XML et d'espace de noms) :

```
<FlightInformation>
   <FlightInfo departuredatetime="2019-12-24T19:43:04+02:00">
      <Station airportcode="KIV"/>
   </FlightInfo>
</FlightInformation>
```

## 6.7.2.9 parse-number

Convertit un string en un nombre décimal, conformément au format spécifié.




### Langages

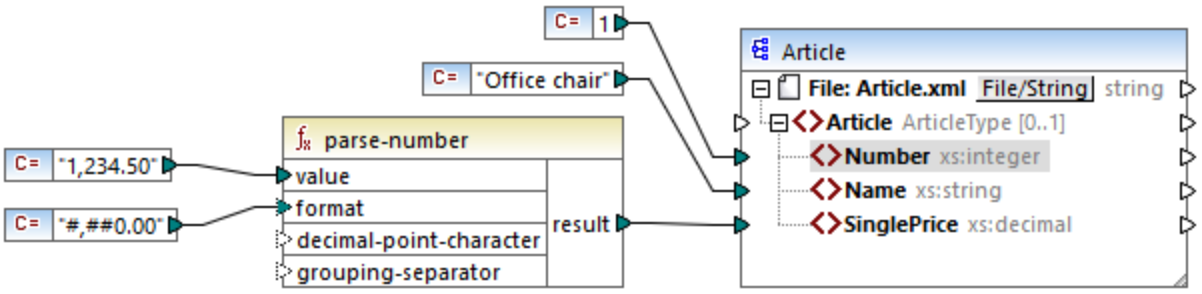
Built-in, C++, C#, Java.

### Paramètres

| Argument | Description |
|---|---|
| **value** | Le string à convertir en un nombre. |
| **format** | Argument optionnel. Un string format qui identifie la manière dont le nombre est formaté actuellement Le string format est le même que celui utilisé dans format-number[561].<br><br>La valeur par défaut est "#,##0.#" |
| **decimal-point-character** | Argument optionnel. Spécifie le caractère à utiliser en tant que le caractère de point décimal. La valeur par défaut est le caractère '.' |
| **grouping-separator** | Argument optionnel. Le séparateur/délimiteur est utilisé pour séparer les groupes de nombres. La valeur par défaut est le caractère "," (optionnel). |

### Exemple

Le mappage suivant parse la valeur de string **`"1,234.50"`** vers un équivalent décimal, en utilisant le masque de format **`#,##0.00`**. Dans ce mappage, il n'y a pas besoin de connecter les arguments **decimal-point-character** et **grouping-separator**, puisque leur valeur par défaut correspond au format du string d'entrée.

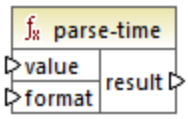



Le résultat de mappage est le suivant (en excluant les déclaration XML et d'espace de noms) :

```
<Article>
   <Number>1</Number>
   <Name>Office chair</Name>
   <SinglePrice>1234.5</SinglePrice>
</Article>
```

## 6.7.2.10 parse-time

Convertit un string en une valeur `xs:time`. Cette fonction utilise la fonction parse-dateTime[567] en tant que base tout en ignorant le composant de date.




### Langages

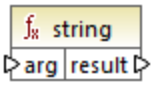
Built-in, C++, C#, Java.

### Paramètres

| Argument | Description |
|---|---|
| **value** | Argument obligatoire. Fournit la valeur string à convertir. |
| **format** | Argument obligatoire. Fournit un string format. Cet argument est utilisé de la même manière que l'argument **format** dans la fonction parse-dateTime[567]. |

### 6.7.2.11 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

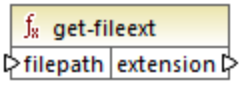
| Argument | Description |
|---|---|
| **arg** | Argument obligatoire. Fournit la valeur à convertir. |

## 6.7.3 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.

### 6.7.3.1 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`.

## 6.7.3.2 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\`.

## 6.7.3.3 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.

## 6.7.3.4 mfd-filepath

Si la fonction est appelée dans le mappage principal, elle retourne la même chose que la fonction main-mfd-filepath 573 , 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

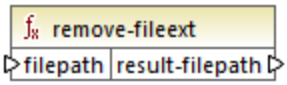
l'utilisateur qui est *importée* par un mfdfile, il retourne le chemin complet du fichier mfd *importé* qui contient la définition de la fonction définie par l'utilisateur.


### Langages

Built-in, C++, C#, Java, XQuery, XSLT 1.0, XSLT 2.0, XSLT 3.0.

## 6.7.3.5 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

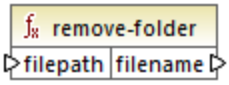
| Argument | Description |
|---|---|
| **filepath** | Argument obligatoire Fournit le chemin de fichier à traiter. |

### Exemple

Si vous fournissez "c:\data\Sample.mfd" en tant qu'argument, le résultat est **`c:\data\Sample`**.

## 6.7.3.6 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

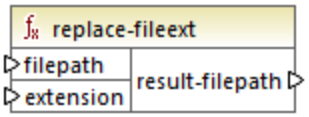
| Argument | Description |
|---|---|
| **filepath** | Argument obligatoire Fournit le chemin de fichier à traiter. |

### Exemple

Si vous fournissez "c:\data\Sample.mfd" en tant qu'argument, le résultat est `Sample.mfd`.

## 6.7.3.7 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

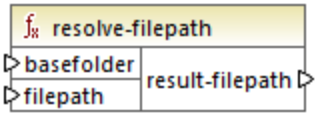
| Argument | Description |
|---|---|
| **filepath** | Argument obligatoire Fournit le chemin de fichier à traiter. |
| **extension** | Argument obligatoire Fournit la nouvelle extension à utiliser. |

### 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`.

## 6.7.3.8 resolve-filepath

Résoud un chemin de fichier relatif contre un dossier de base. La fonction prend en charge '.' (répertoire actuel) et '..' (répertoire parent).




### Langages

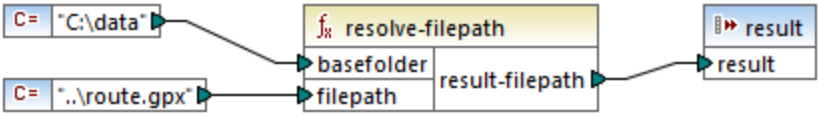
Built-in, C++, C#, Java, XQuery, XSLT 1.0, XSLT 2.0, XSLT 3.0.

### 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**.



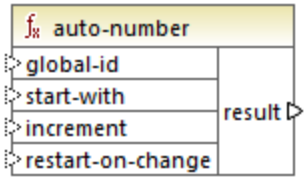

Le résultat de mappage est **C:\route.gpx**.

## 6.7.4 core | generator functions

La bibliothèque de fonctions **core / generator** contient des fonctions qui génèrent des valeurs.

### 6.7.4.1 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 618 à la place.

## Langages

Built-in, C++, C#, Java, XQuery, XSLT 1.0, XSLT 2.0, XSLT 3.0.

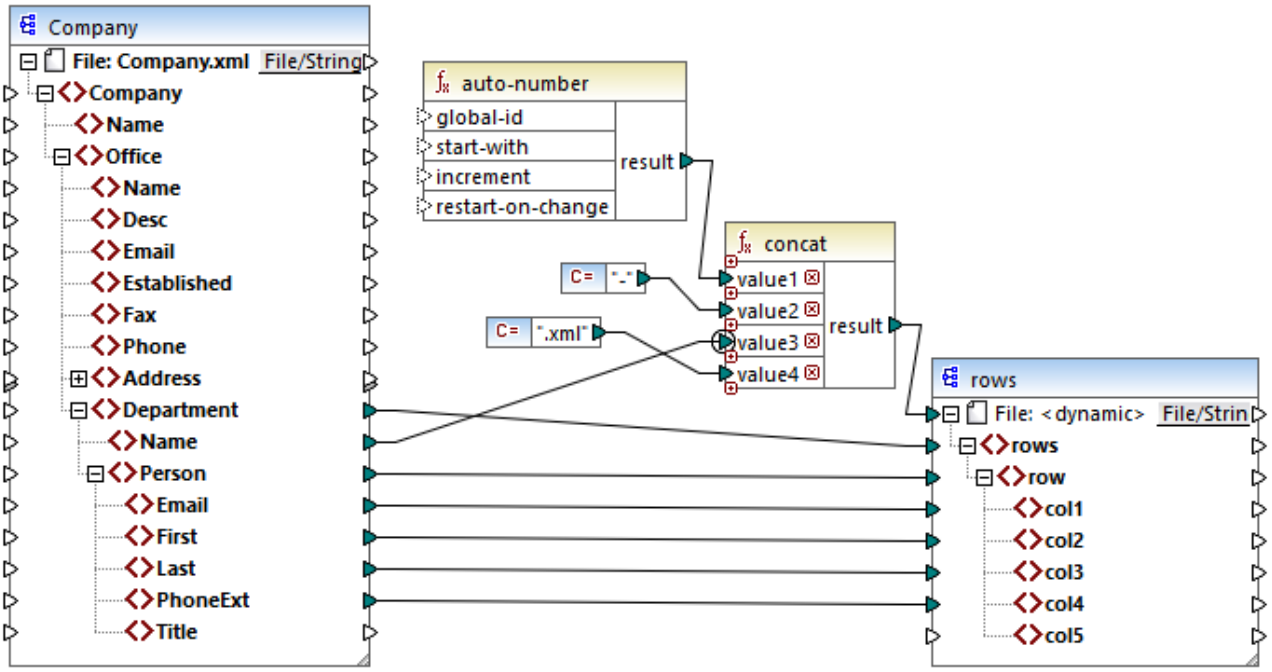
## Paramètres

| 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**. |
| **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 810.

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[814] 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.

## 6.7.5 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[444]. 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**).

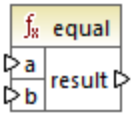
Si toutes les valeurs d'entrée sont de même type de données, alors la comparaison est effectuée pour le type commun. Si des valeurs d'entrée sont de types différents (par exemple, `integer` et `string` ou `string` et `date`), alors le type de données utilisé pour la comparaison est le plus général (le 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 131.

## 6.7.5.1 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

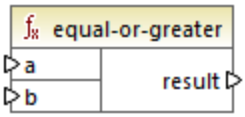
Built-in, C++, C#, Java, XQuery, XSLT 1.0, XSLT 2.0, XSLT 3.0.

### Paramètres

| Argument | Description |
|---|---|
| a | Paramètre obligatoire. Fournit la première valeur à comparer. |
| b | Paramètre obligatoire. Fournit la seconde valeur à comparer. |

## 6.7.5.2 equal-or-greater

Retourne Booléenne **true** si *a* est égal ou supérieur à *b*; **false** sinon.




### Langages

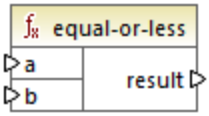
Built-in, C++, C#, Java, XQuery, XSLT 1.0, XSLT 2.0, XSLT 3.0.

### Paramètres

| Argument | Description |
|---|---|
| **a** | Paramètre obligatoire. Fournit la première valeur à comparer. |
| **b** | Paramètre obligatoire. Fournit la seconde valeur à comparer. |

## 6.7.5.3 equal-or-less

Retourne Booléenne **true** si *a* est égal ou inférieur à *b*; **false** sinon.




### Langages

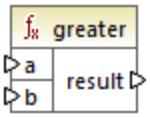
Built-in, C++, C#, Java, XQuery, XSLT 1.0, XSLT 2.0, XSLT 3.0.

### Paramètres

| Argument | Description |
|---|---|
| **a** | Paramètre obligatoire. Fournit la première valeur à comparer. |
| **b** | Paramètre obligatoire. Fournit la seconde valeur à comparer. |

## 6.7.5.4 greater

Retourne Booléenne **true** si *a* est supérieur à *b*; **false** sinon.




### Langages

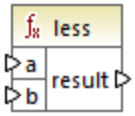
Built-in, C++, C#, Java, XQuery, XSLT 1.0, XSLT 2.0, XSLT 3.0.

### Paramètres

| Argument | Description |
|---|---|
| **a** | Paramètre obligatoire. Fournit la première valeur à comparer. |
| **b** | Paramètre obligatoire. Fournit la seconde valeur à comparer. |

## 6.7.5.5 less

Retourne Booléenne **true** si *a* est inférieur à *b*; **false** sinon.




### Langages

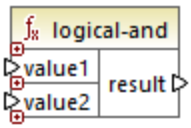
Built-in, C++, C#, Java, XQuery, XSLT 1.0, XSLT 2.0, XSLT 3.0.

### Paramètres

| Argument | Description |
|---|---|
| **a** | Paramètre obligatoire. Fournit la première valeur à comparer. |
| **b** | Paramètre obligatoire. Fournit la seconde valeur à comparer. |

## 6.7.5.6 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 469.




### Langages

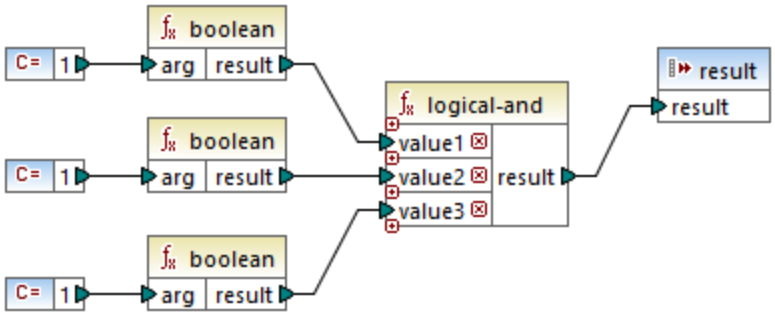
Built-in, C++, C#, Java, XQuery, XSLT 1.0, XSLT 2.0, XSLT 3.0.

### Paramètres

| Argument | Description |
|---|---|
| **value1** | Paramètre obligatoire. Fournit la première valeur à comparer. |
| **value2** | Paramètre obligatoire. Fournit la seconde valeur à comparer. |

### Exemple

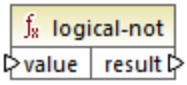
Le mappage illustré ci-dessous retourne **true** parce que toutes les valeurs d'entrée dans la fonction **`logical-and`** sont **true** également. Si une de ces valeurs d'entrée était **false**, le résultat de mappage serait **false** également.




Voir aussi Exemple : Consultation et Concaténation 504.

## 6.7.5.7 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.

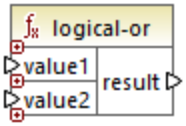
### Paramètres

| Argument | Description |
|---|---|
| **value** | Paramètre obligatoire. Fournit la valeur d'entrée. |

## 6.7.5.8 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 469.




### Langages

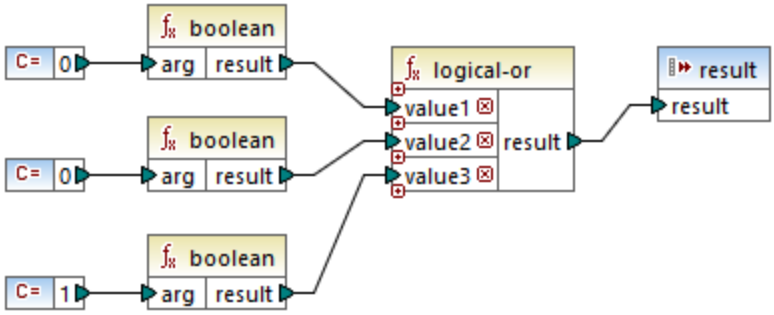
Built-in, C++, C#, Java, XQuery, XSLT 1.0, XSLT 2.0, XSLT 3.0.

### Paramètres

| Argument | Description |
|---|---|
| **value1** | Paramètre obligatoire. Fournit la première valeur à comparer. |
| **value2** | Paramètre obligatoire. Fournit la seconde valeur à comparer. |

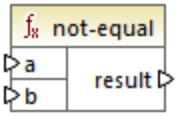
### Exemple

Le résultat du mappage ci-dessous est **true**, parcequ'au moins un des arguments de la fonction est **true**.




### 6.7.5.9 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.

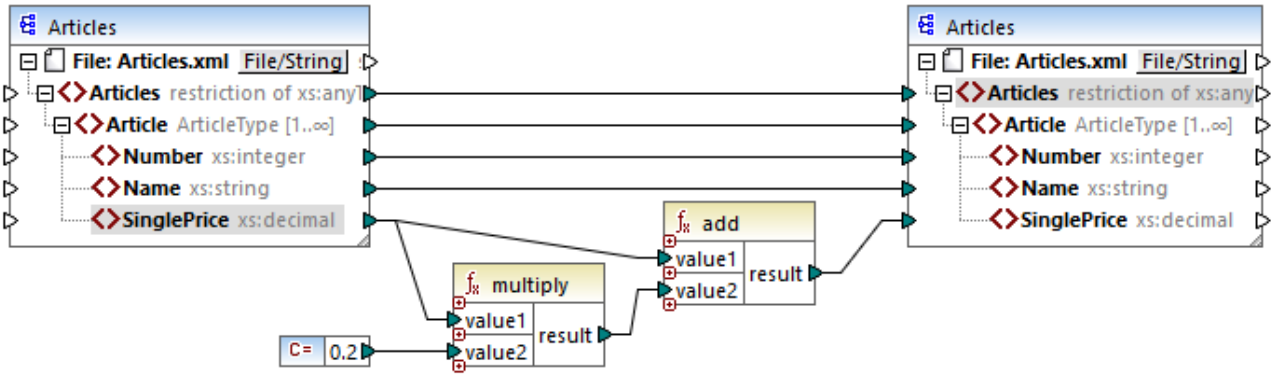
#### Paramètres

| Argument | Description |
|---|---|
| **a** | Paramètre obligatoire. Fournit la première valeur à comparer. |
| **b** | Paramètre obligatoire. Fournit la seconde valeur à comparer. |

## 6.7.6 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`.

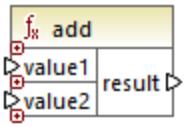
La plupart des fonctions math prennent deux paramètres d'entrée (**value1**, **value2**) qui sont des opérandes de l'opération mathématique. Les valeurs d'entrée sont converties automatiquement au type `decimal` pour un traitement ultérieur. Le résultat des fonctions math est aussi de type `decimal`.




L'exemple indiqué ci-dessus ajoute une taxe de vente de 20% pour chacun des articles mappés dans le composant de cible.

## 6.7.6.1 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 469.




### Langages

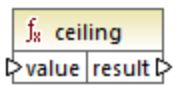
Built-in, C++, C#, Java, XQuery, XSLT 1.0, XSLT 2.0, XSLT 3.0.

### Paramètres

| Argument | Description |
|---|---|
| **value1** | Paramètre obligatoire. Fournit le premier opérande. |
| **value2** | Paramètre obligatoire. Fournit la seconde opérande. |

## 6.7.6.2 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

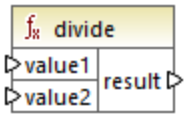
| Argument | Description |
|---|---|
| **value** | Paramètre obligatoire. Fournit la valeur d'entrée de la fonction. |

### 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**.

## 6.7.6.3 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 589 pour définir la précision du résultat.




### Langages

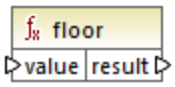
Built-in, C++, C#, Java, XQuery, XSLT 1.0, XSLT 2.0, XSLT 3.0.

### Paramètres

| Argument | Description |
|---|---|
| **value1** | Paramètre obligatoire. Fournit le premier opérande. |
| **value2** | Paramètre obligatoire. Fournit la seconde opérande. |

## 6.7.6.4 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

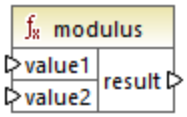
| Argument | Description |
|---|---|
| **value** | Paramètre obligatoire. Fournit la valeur d'entrée de la fonction. |

### 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**.

## 6.7.6.5 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. |

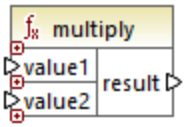
### Exemple

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.

## 6.7.6.6 multiply

Multiplie **value1** par **value2** et retourne le résultat en tant que valeur décimale.




### Langages

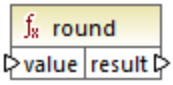
Built-in, C++, C#, Java, XQuery, XSLT 1.0, XSLT 2.0, XSLT 3.0.

### Paramètres

| Argument | Description |
|---|---|
| **value1** | Paramètre obligatoire. Fournit le premier opérande. |
| **value2** | Paramètre obligatoire. Fournit la seconde opérande. |

## 6.7.6.7 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

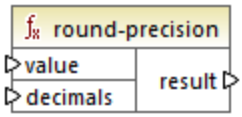
Built-in, C++, C#, Java, XQuery, XSLT 1.0, XSLT 2.0, XSLT 3.0.

### Paramètres

| Argument | Description |
|---|---|
| **value** | Paramètre obligatoire. Fournit la valeur d'entrée de la fonction. |

## 6.7.6.8 round-precision

Arrondit la valeur d'entrée à N *N* décimales, où *N* est l'argument **decimals**.




### Langages

Built-in, C++, C#, Java.

### Paramètres

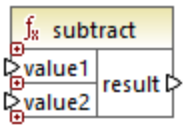
| Argument | Description |
|---|---|
| **value** | Paramètre obligatoire. Fournit la valeur d'entrée de la fonction. |
| **decimals** | Paramètre obligatoire. Spécifie le nombre des décimales pour arrondir. |

### Exemple

Arrondir la valeur **2.777777** à deux décimales **2.78**. Arrondir la valeur **0.1234** à 3 décimales **0.123**.

## 6.7.6.9 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. |

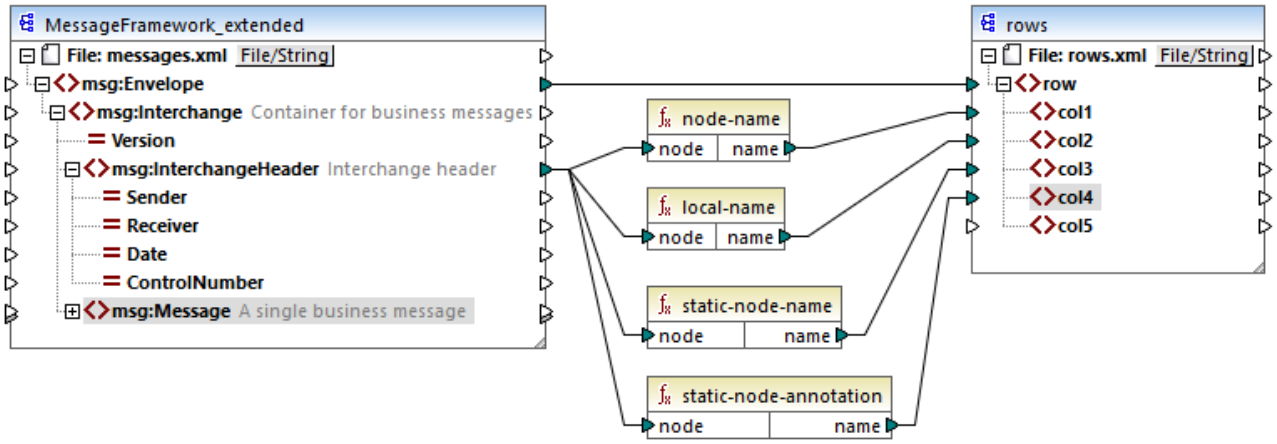
## 6.7.7 core | node functions

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 131.

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 766.

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.

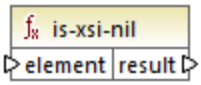



La sortie du mappage est le suivant (excluant les déclarations XML et d'espace de noms) :

```
<row>
   <col1>msg:InterchangeHeader</col1>
   <col2>InterchangeHeader</col2>
   <col3>msg:InterchangeHeader</col3>
   <col4>Interchange header</col4>
</row>
```

## 6.7.7.1 is-xsi-nil

Retourne **true** si le nœud **element** a l'attribut `xsi:nil` défini sur **true**.




### Langages

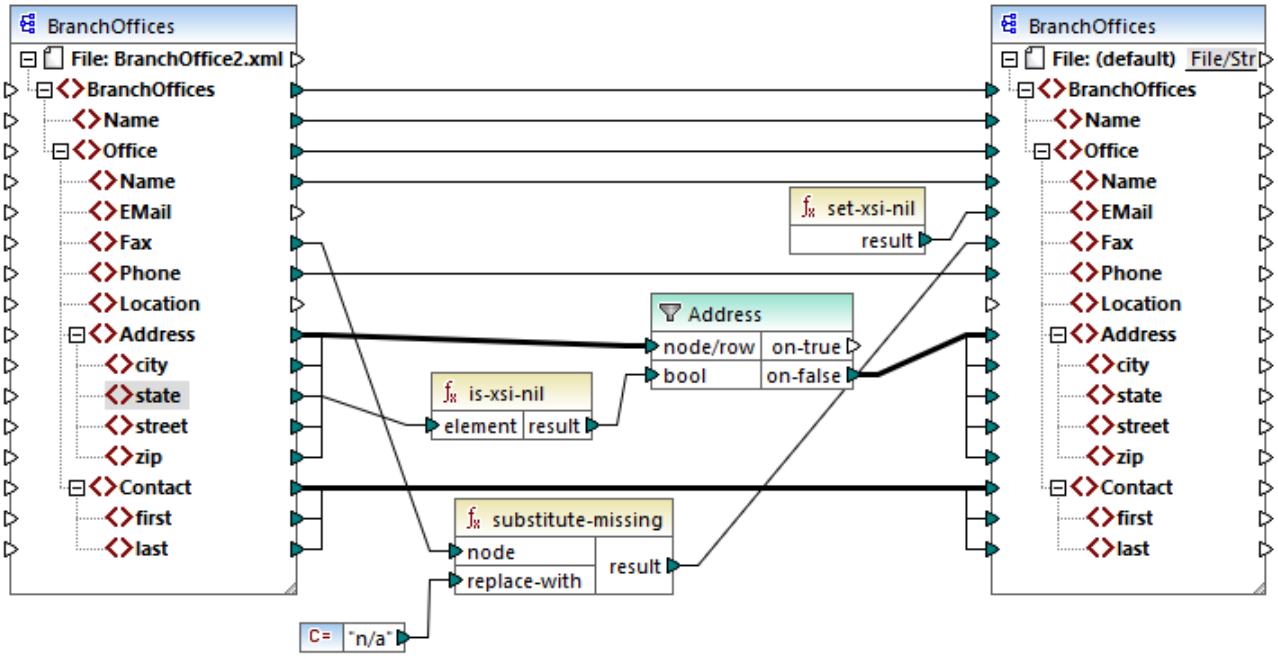
Built-in, C++, C#, Java, XQuery, XSLT 1.0, XSLT 2.0, XSLT 3.0.

### 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>
```

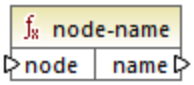
Le résultat du mappage est qu'aucun **Address** n'est copié dans la cible, parce qu'il n'y a qu'un seul **Address** dans la source, et l'attribut `xsi:nil` est défini sur "true" pour l'élément **state**. Par conséquent, la sortie de mappage est le suivant :

```
<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>
```

## 6.7.7.2 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

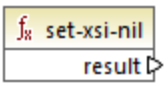
Built-in, C++, C#, Java, XQuery, XSLT 2.0, XSLT 3.0.

### Paramètres

| Argument | Description |
|---|---|
| **node** | Paramètre obligatoire. Connecter cette entrée dans le nœud dont vous souhaitez obtenir le nom. |

## 6.7.7.3 set-xsi-nil

Défini le nœud de cible sur xsi:nil.



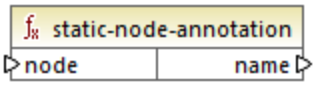

### Langages

Built-in, C++, C#, Java, XQuery, XSLT 2.0, XSLT 3.0.

## 6.7.7.4 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[492]" qui est directement connectée à un paramètre[497] 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.

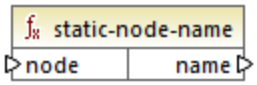
### Paramètres

| Argument | Description |
|---|---|
| **node** | Paramètre obligatoire. Connecter cette entrée vers le nœud dont vous souhaitez obtenir l'annotation. |

## 6.7.7.5 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[492]" qui est directement connectée à un paramètre[497] 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

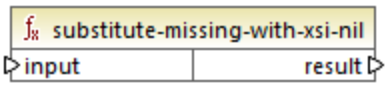
Built-in, C++, C#, Java, XQuery, XSLT 1.0, XSLT 2.0, XSLT 3.0.

### Paramètres

| Argument | Description |
|---|---|
| **node** | Paramètre obligatoire. Connecter cette entrée dans le nœud dont vous souhaitez obtenir le nom. |

## 6.7.7.6 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

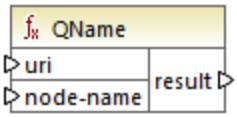
| Argument | Description |
|---|---|
| **input** | Paramètre obligatoire. Connecter cette entrée dans le nœud dont vous souhaitez obtenir le nom. |

## 6.7.8 core | QName functions

Les fonctions QName permettent de manipuler les Qualified Names (QName) dans les documents XML.

### 6.7.8.1 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

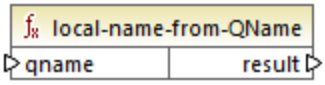
Built-in, C++, C#, Java, XQuery, XSLT 2.0, XSLT 3.0.

### Paramètres

| Nom | Description |
|---|---|
| **uri** | Obligatoire. Fournit l'URI. |
| **node-name** | Obligatoire. Fournit le nom du nœud. |

### 6.7.8.2 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`. |

## 6.7.8.3 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

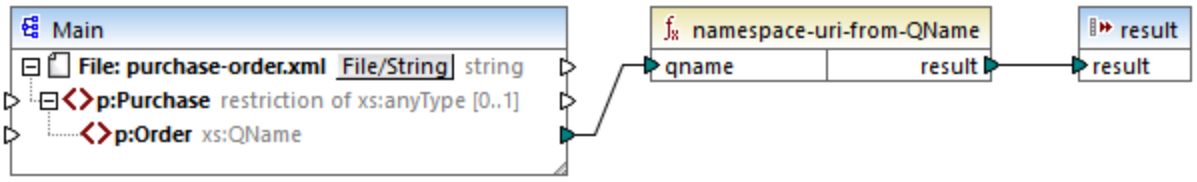
| Nom | Description |
|---|---|
| **qname** | Obligatoire. Fournit la valeur d'entrée de la fonction. |

### 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 :



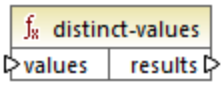

La sortie de ce mappage est **`http://NamespaceTest.com/Order`**.

## 6.7.9 core | sequence functions

Les fonctions de séquence permettent le traitement des sequences[803] d'entrée et du regroupement de leur contenu.

### 6.7.9.1 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[803] de zéros ou plus de valeurs. Par exemple, la connexion peut provenir d'un item XML de source , un champ CSV, un enregistrement de base de données, etc. |

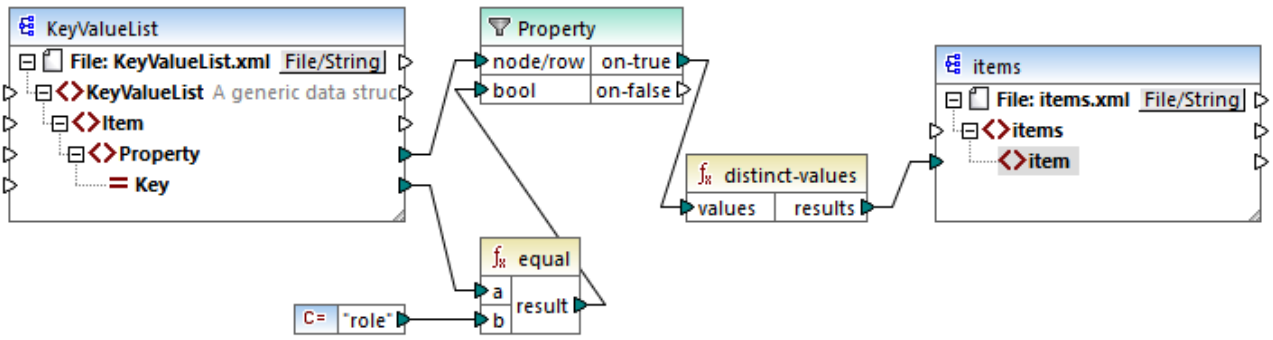
#### Exemple

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 :

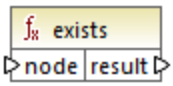
- Chaque élément **Property** provenant du fichier XML de source est traité par un filtre.
- La connexion à l'entrée **bool** du filtre assure que seuls des éléments **Property** où l'attribut **Key** est égale à "role" sont fournis dans le composant de la cible. Le string "role" est fourni par une constante. 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>
```

## 6.7.9.2 exists

Retourne **true** si le nœud connecté existe; **false** sinon. Puisqu'elle retourne une valeur booléenne, cette fonction est généralement utilisée avec filters 444, pour filtrer uniquement les enregistrements qui ont (ou peut-être qui n'ont pas) un élément ou un attribut enfant.




### Langages

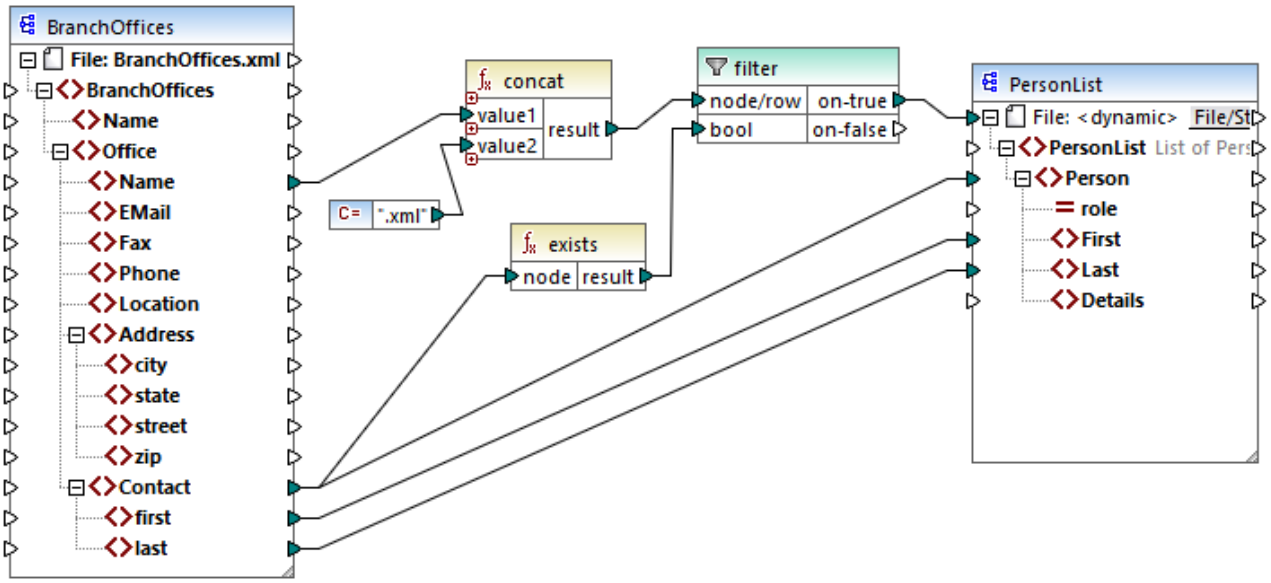
Built-in, C++, C#, Java, XQuery, XSLT 1.0, XSLT 2.0, XSLT 3.0.

### 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*

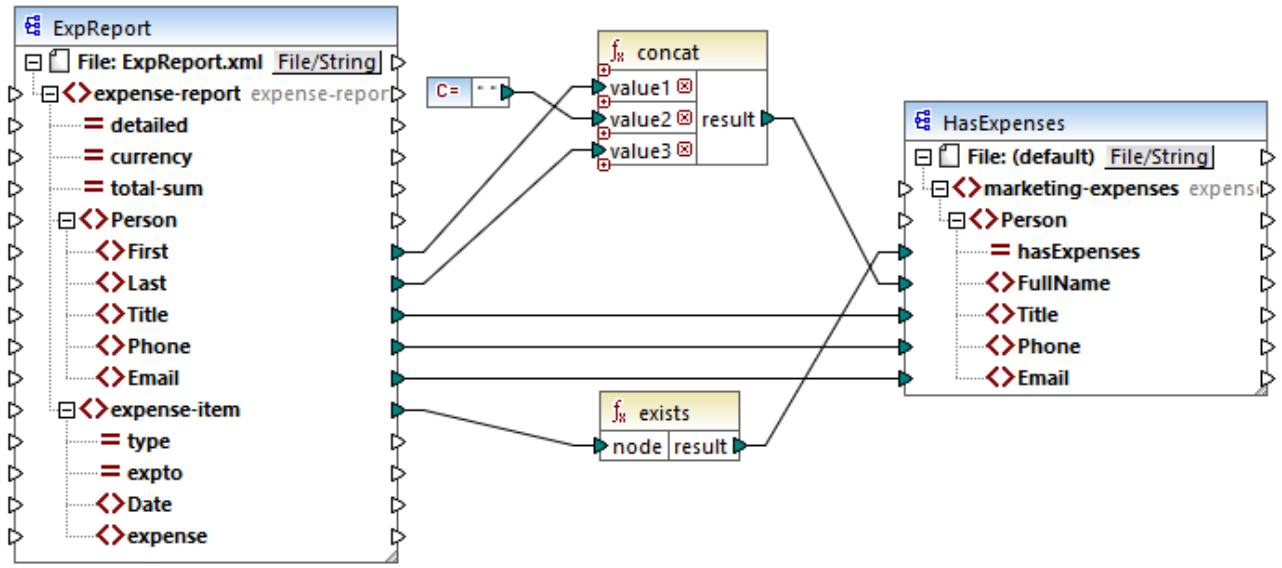
Dans le fichier source **BranchOffices.xml**, vous trouverez trois éléments **Office**. Notablement, un des trois bureaux n'a pas d'éléments enfant **Contact**. L'objectif de ce mappage est multiple :

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 787 .

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.

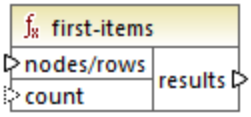



*HasMarketingExpenses.mfd*

Voir aussi Exemple : Exception lorsque le nœud n'existe pas 464.

## 6.7.9.3 first-items

Retourne les premiers items *N* de la séquence d'entrée, où *N* est fourni par le paramètre **count**.




### Langages

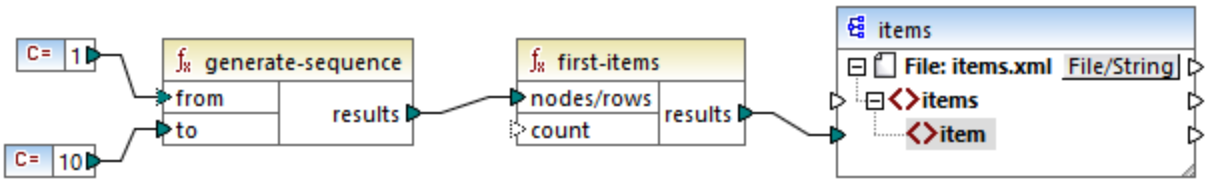
Built-in, C++, C#, Java, XQuery, XSLT 2.0, XSLT 3.0.

### Paramètres

| Nom | Description |
|---|---|
| **nodes/rows** | Cette entrée doit recevoir une connexion depuis un item de mappage qui fournit une séquence 803 de zéros ou plus de valeurs. Par exemple, la connexion peut provenir d'un item XML de source , un champ CSV, un enregistrement de base de données, etc. |
| **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.



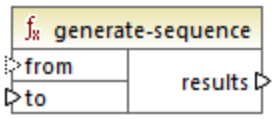

Étant donné que l'argument **count** n'a pas de valeur, la valeur par défaut de **1** s'applique. Par conséquence, seule la première valeur provenant de la séquence est générée dans la sortie du mappage :

```
<items>
   <item>1</item>
</items>
```

Pour un exemple plus réaliste, voir le mappage **FindHighestTemperatures.mfd** discuté dans Fournir les paramètres au mappage 398.

## 6.7.9.4 generate-sequence

Crée une séquence des entiers en utilisant les paramètres "from" et "to" en tant que les limites.




### Langages

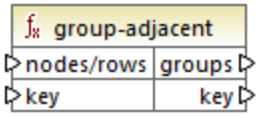
Built-in, C++, C#, Java, XQuery, XSLT 2.0, XSLT 3.0.

### 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). |

## 6.7.9.5 group-adjacent

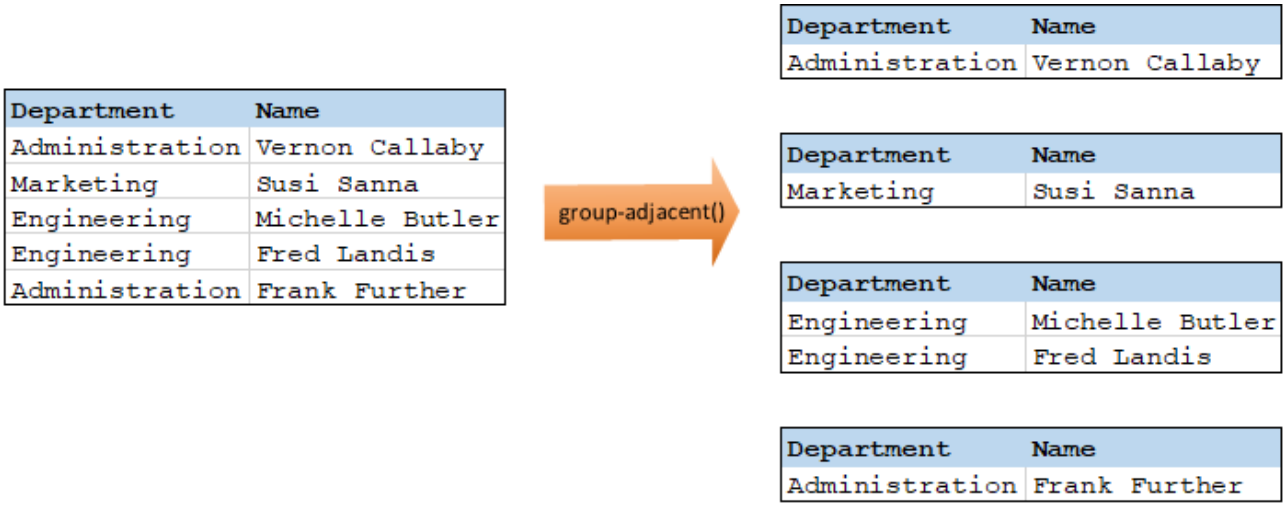
La fonction **group-adjacent** regroupe les items connectés aux entrées **nodes/rows** par la clé connectée à l'entrée **key**. Veuillez noter que cette fonction place les items qui partagent la même clé dans des groupes séparés s'ils ne sont pas adjacents. Si plusieurs items consécutifs (adjacents) 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

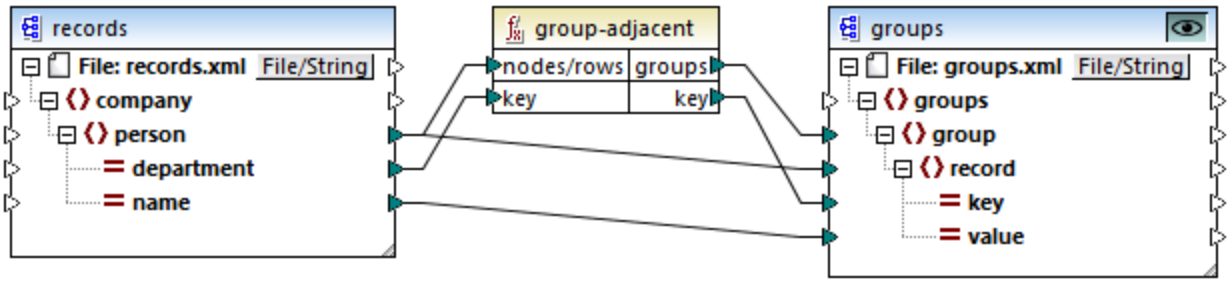
| Nom | Description |
|---|---|
| **nodes/rows** | Cette entrée doit recevoir une connexion depuis un item de mappage qui fournit une séquence[803] de zéros ou plus de valeurs. Par exemple, la connexion peut provenir d'un item XML de source , un champ CSV, un enregistrement de base de données, etc. |
| **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 :

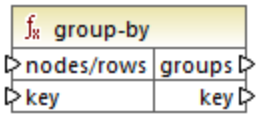
```
<groups>
   <group>
      <record key="Administration" value="Vernon Callaby"/>
   </group>
   <group>
      <record key="Marketing" value="Susi Sanna"/>
   </group>
   <group>
```

```
      <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**.

## 6.7.9.6 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

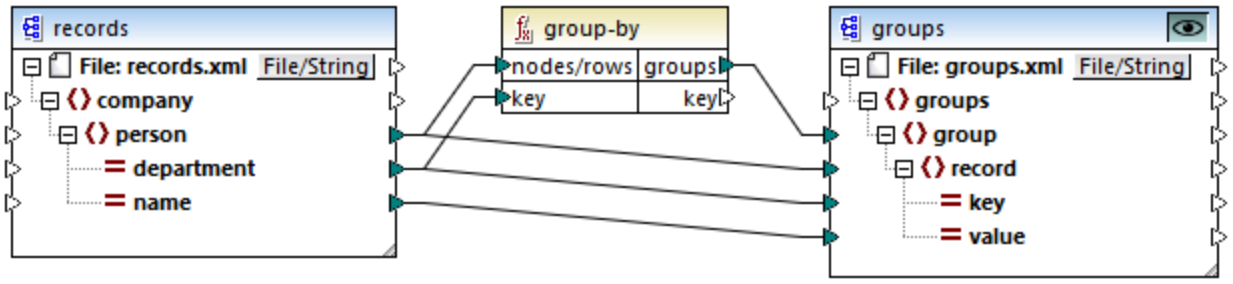
| Nom | Description |
|---|---|
| **nodes/rows** | Cette entrée doit recevoir une connexion depuis un item de mappage qui fournit une séquence[803] de zéros ou plus de valeurs. Par exemple, la connexion peut provenir d'un item XML de source , un champ CSV, un enregistrement de base de données, etc. |
| **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

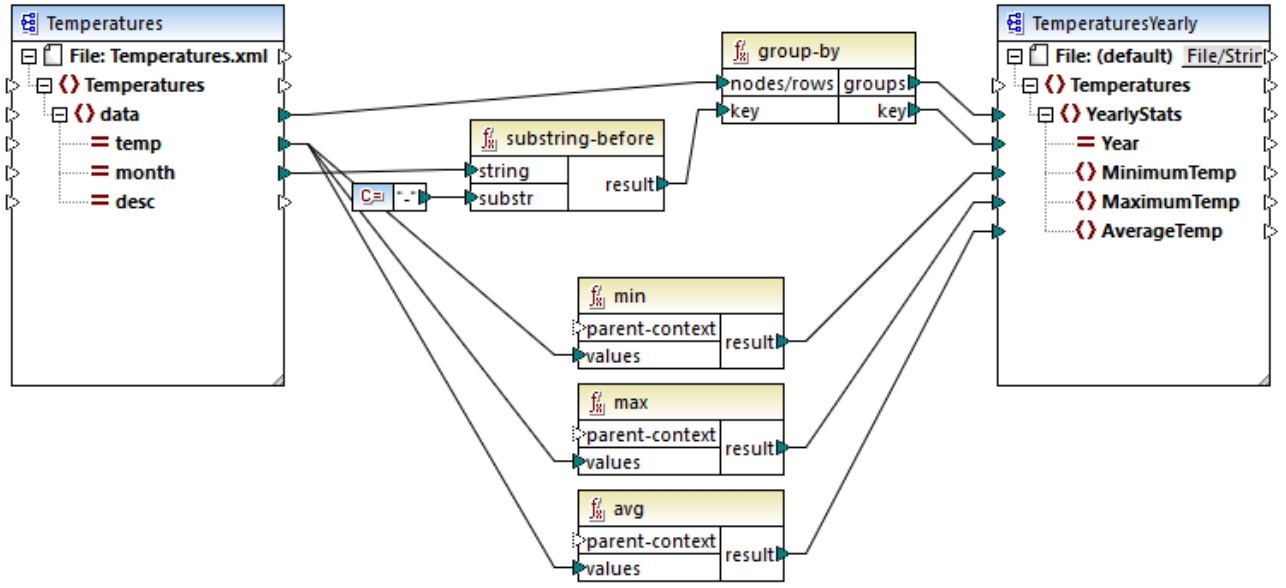
Cet exemple vous montre comment regrouper des enregistrements avec l'aide de la fonction **group-by**, de même, il illustre comment agréger des données. Cet exemple est accompagné par un mappage de démonstration disponible sous le chemin suivant : **<Documents>\Altova\MapForce2024\MapForceExamples\GroupTemperaturesByYear.mfd**. 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**[553], **max**[552], et **avg**[550].




*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**[634] 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 :

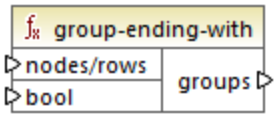
```
<Temperatures>
   <YearlyStats Year="2006">
      <MinimumTemp>-3.6</MinimumTemp>
      <MaximumTemp>23.2</MaximumTemp>
      <AverageTemp>11.375</AverageTemp>
   </YearlyStats>
   <YearlyStats Year="2007">
```

```
      <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.

## 6.7.9.7 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 803 de zéros ou plus de valeurs. Par exemple, la |

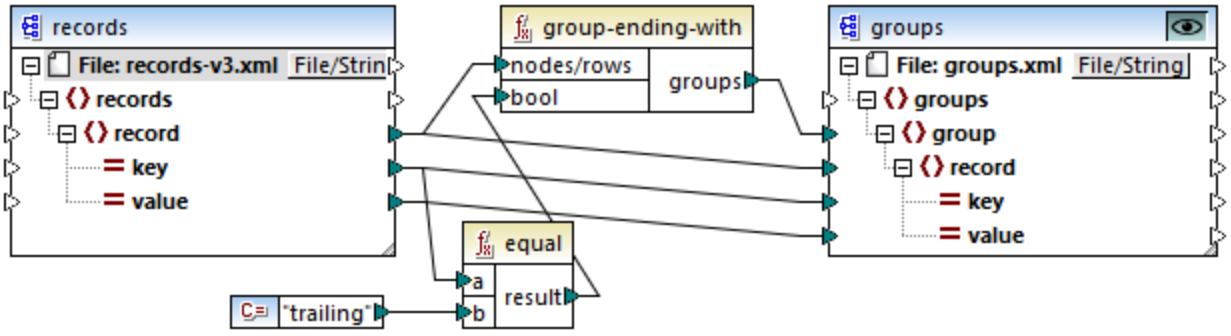
| Nom | Description |
|---|---|
| | connexion peut provenir d'un item XML de source , un champ CSV, un enregistrement de base de données, etc. |
| **bool** | Fournit la condition Booléenne qui lance un nouveau groupe si **true**. |

## 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 :

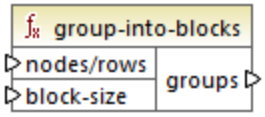
```
<groups>
   <group>
      <record key="line" value="A"/>
      <record key="line" value="B"/>
      <record key="trailing" value="Total 1"/>
   </group>
   <group>
      <record key="line" value="C"/>
      <record key="trailing" value="Total 2"/>
```

```
   </group>
</groups>
```

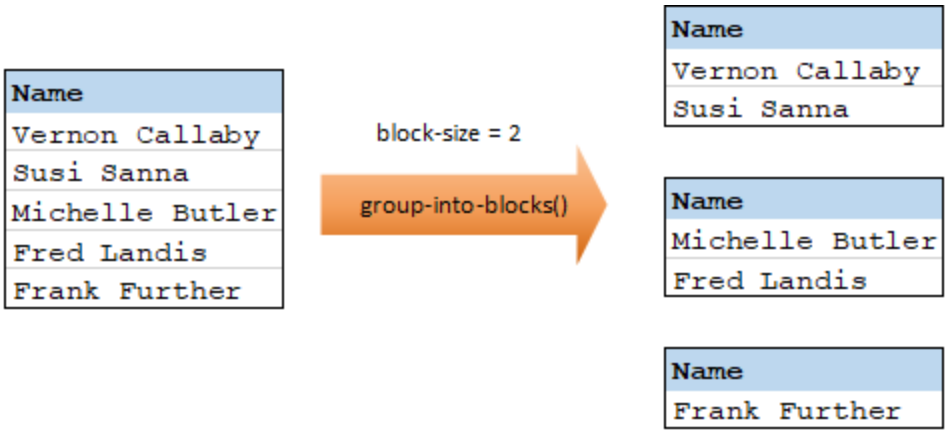
Cet exemple, avec d'autres exemples de regroupement, fait partie du fichier de mappage suivant : **<Documents>\Altova\MapForce2024\MapForceExamples\Tutorial\GroupingFunctions.mfd**. Ne pas oublier de cliquer sur la touche **Aperçu** applicable à la fonction que vous souhaitez consulter préalablement, avant de cliquer sur l'onglet **Sortie**.

## 6.7.9.8 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[803] de zéros ou plus de valeurs. Par exemple, la connexion peut provenir d'un item XML de source , un champ CSV, un enregistrement de base de données, etc. |

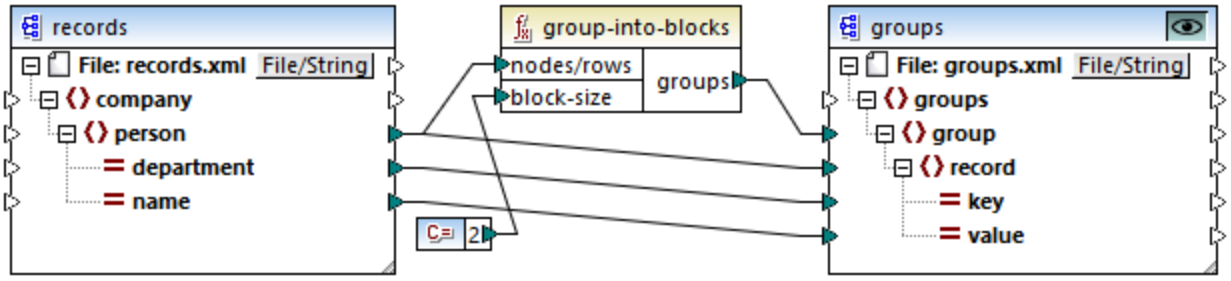
| Nom | Description |
|---|---|
| **block-size** | Spécifie la taille de chaque groupe |

## 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 :

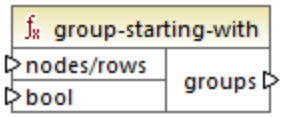
```
<groups>
   <group>
      <record key="Administration" value="Vernon Callaby"/>
      <record key="Marketing" value="Susi Sanna"/>
   </group>
   <group>
      <record key="Engineering" value="Michelle Butler"/>
      <record key="Engineering" value="Fred Landis"/>
   </group>
   <group>
      <record 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**.

## 6.7.9.9 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 :

| Key | Value |
|---|---|
| heading | Intro |
| line | A |
| line | B |
| heading | Body |
| line | C |

group-starting-with()

| Key | Value |
|---|---|
| heading | Intro |
| line | A |
| line | B |

| Key | Value |
|---|---|
| heading | Body |
| line | C |

**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

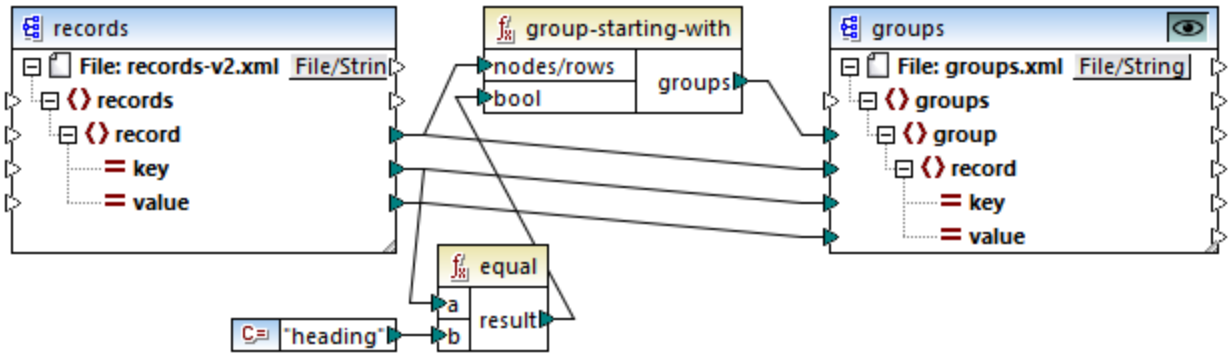
| Nom | Description |
|---|---|
| **nodes/rows** | Cette entrée doit recevoir une connexion depuis un item de mappage qui fournit une séquence[803] de zéros ou plus de valeurs. Par exemple, la connexion peut provenir d'un item XML de source , un champ CSV, un enregistrement de base de données, etc. |
| **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.



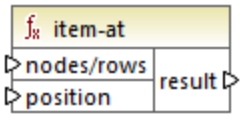

Le résultat de mappage est le suivant :

```
<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**.

## 6.7.9.10 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

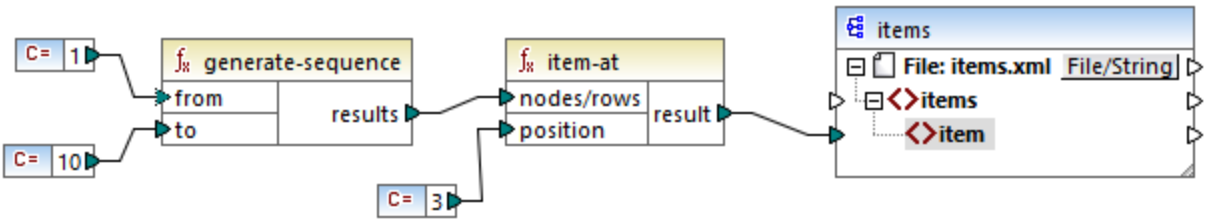
Built-in, C++, C#, Java, XQuery, XSLT 2.0, XSLT 3.0.

### Paramètres

| Nom | Description |
|---|---|
| **nodes/rows** | Cette entrée doit recevoir une connexion depuis un item de mappage qui fournit une séquence[803] de zéros ou plus de valeurs. Par exemple, la connexion peut provenir d'un item XML de source , un champ CSV, un enregistrement de base de données, etc. |
| **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.

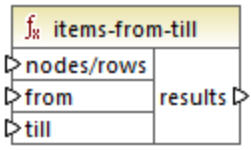



Étant donné que l'argument **position** est défini sur **3**, seule la troisième valeur provenant de la séquence est transmise vers la cible. Par conséquent, la sortie de mappage est la suivante (à l'exception des déclarations XML et de schéma) :

```
<items>
   <item>3</item>
</items>
```

## 6.7.9.11 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

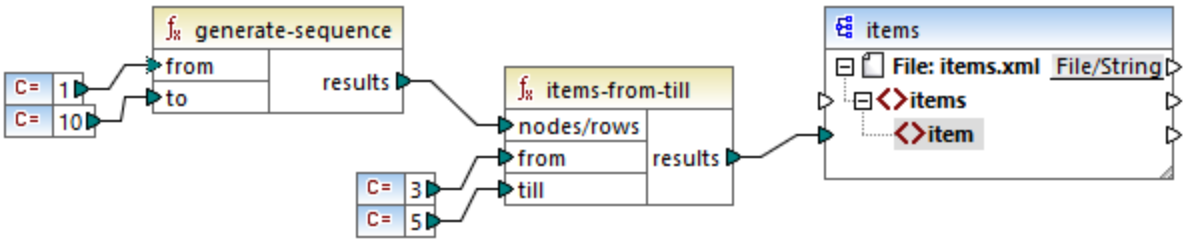
Built-in, C++, C#, Java, XQuery, XSLT 2.0, XSLT 3.0.

### Paramètres

| Nom | Description |
|---|---|
| **nodes/rows** | Cette entrée doit recevoir une connexion depuis un item de mappage qui fournit une séquence[803] de zéros ou plus de valeurs. Par exemple, la connexion peut provenir d'un item XML de source , un champ CSV, un enregistrement de base de données, etc. |
| **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.

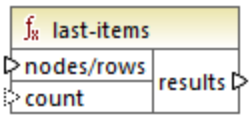



Étant donné que les arguments **from** et **till** sont définis sur **3** et **5**, respectivement, seul le sous-ensemble de valeurs provenant de **3** à **5** est transmis vers la cible. Par conséquent, la sortie de mappage est la suivante (à l'exception des déclarations XML et de schéma) :

```
<items>
   <item>3</item>
   <item>4</item>
   <item>5</item>
</items>
```

## 6.7.9.12 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

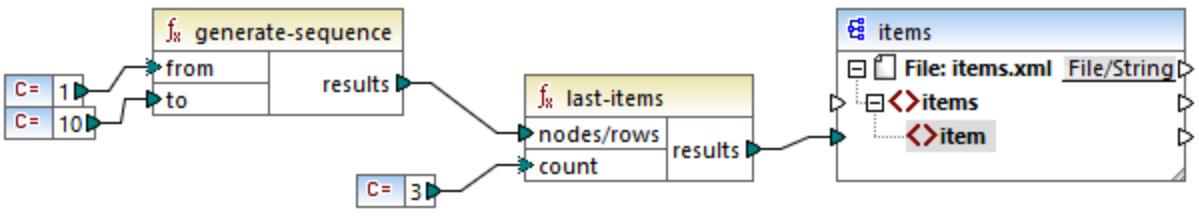
Built-in, C++, C#, Java, XQuery, XSLT 2.0, XSLT 3.0.

### Paramètres

| Nom | Description |
|---|---|
| **nodes/rows** | Cette entrée doit recevoir une connexion depuis un item de mappage qui fournit une séquence 803 de zéros ou plus de valeurs. Par exemple, la connexion peut provenir d'un item XML de source , un champ CSV, un enregistrement de base de données, etc. |
| **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.

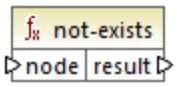



Étant donné que l'argument count est défini sur **3**, seule les trois dernières valeurs provenant de la séquence sont transmises vers la cible. Par conséquent, la sortie de mappage est la suivante (à l'exception des déclarations XML et de schéma) :

```
<items>
   <item>8</item>
   <item>9</item>
   <item>10</item>
</items>
```

## 6.7.9.13 not-exists

Retourne **false** si le nœud connecté existe; **true** sinon. Cette fonction est l'inverse de la fonction exists[599] mais à part cela elle a la même utilisation.




### Langages

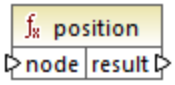
Built-in, C++, C#, Java, XQuery, XSLT 1.0, XSLT 2.0, XSLT 3.0.

### Paramètres

| Nom | Description |
|---|---|
| **node** | Le nœud à tester pour non-existence. |

## 6.7.9.14 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.

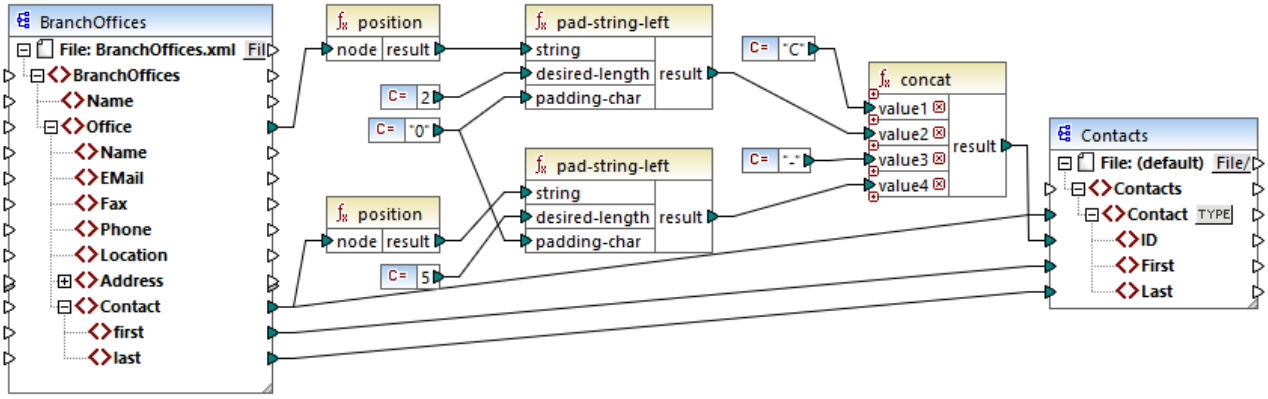
### Paramètres

| Nom | Description |
|---|---|
| **node** | Cette entrée doit recevoir une connexion depuis un item de mappage qui fournit une séquence[803] de zéros ou plus de valeurs. Par exemple, la connexion peut |

| Nom | Description |
|---|---|
| | provenir d'un item XML de source , un champ CSV, un enregistrement de base de données, etc. |

## 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.

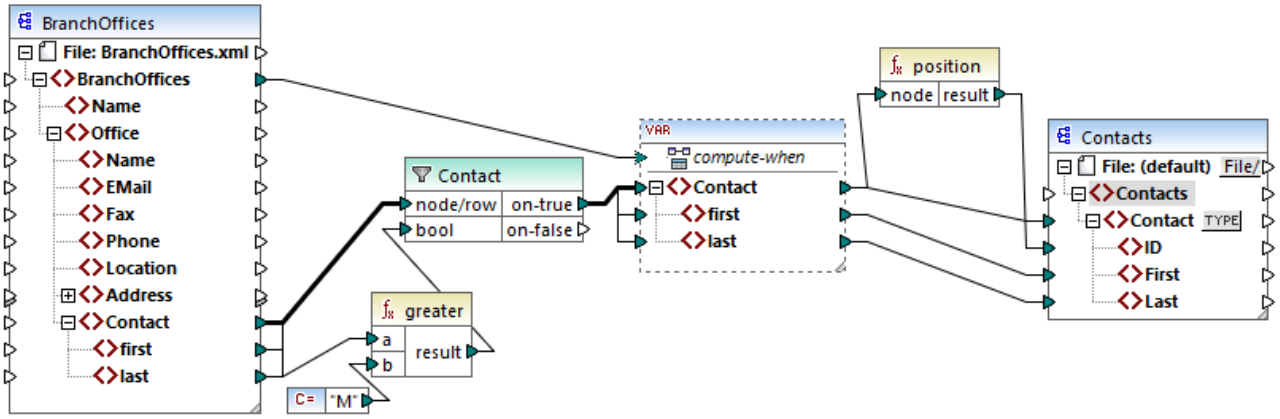
Lorsque vous utilisez la fonction **position**, il est important de considérer le contexte de mappage[804] actuel. Plus spécifiquement, lorsque le mappage est exécuté, le contexte de mappage initial est établi depuis l'item root du composant cible vers l'item de source qui y est connecté (même indirectement 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>
```

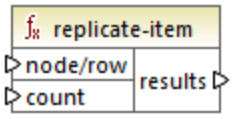
Il peut aussi exister des cas où vous souhaitez obtenir la position d'items résultant après l'application d'un filtre[444]. Veuillez noter que le composant de filtre n'est pas une fonction de séquence et qu'il ne peut pas être utilisé *directement* en conjonction avec la fonction de **`position`** pour trouver la position des items filtrés. Indirectement, cela est possible en ajoutant un composant de variable[413] 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 422.

## 6.7.9.15 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

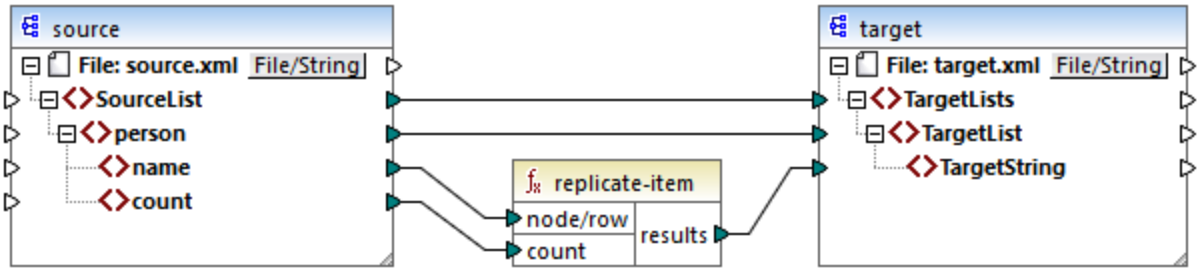
| Nom | Description |
|---|---|
| **node/row** | Cette entrée doit recevoir une connexion depuis un item de mappage qui fournit une séquence 803 de zéros ou plus de valeurs. Par exemple, la connexion peut provenir d'un item XML de source , un champ CSV, un enregistrement de base de données, etc. |
| **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** :



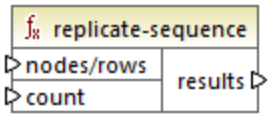

La sortie est la suivante :

```
<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>
```

## 6.7.9.16 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

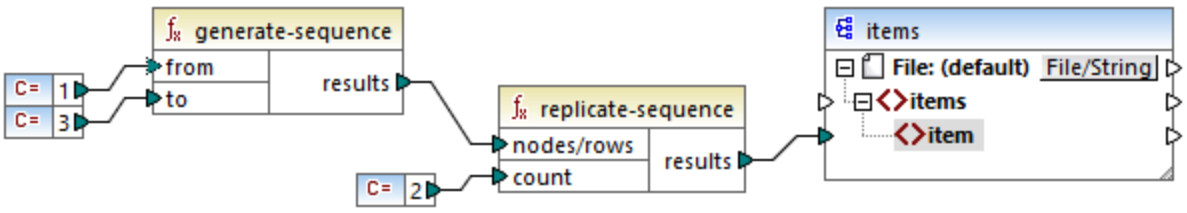
Built-in, C++, C#, Java, XQuery, XSLT 2.0, XSLT 3.0.

### Paramètres

| Nom | Description |
|---|---|
| **node/rows** | Cette entrée doit recevoir une connexion depuis un item de mappage qui fournit une séquence[803] de zéros ou plus de valeurs. Par exemple, la connexion peut provenir d'un item XML de source , un champ CSV, un enregistrement de base de données, etc. |
| **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.



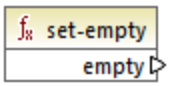

Puisque l'argument **count** est défini sur **2**, la séquence est répliquée deux fois puis est transmise sur la cible. Par conséquent, la sortie de mappage est la suivante (à l'exception des déclarations XML et de schéma) :

```
<items>
   <item>1</item>
```

```
   <item>2</item>
   <item>3</item>
   <item>1</item>
   <item>2</item>
   <item>3</item>
</items>
```

## 6.7.9.17 set-empty

Retourne une séquence vide.

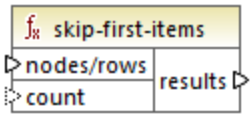



### Langages

Built-in, C++, C#, Java, XQuery, XSLT 2.0, XSLT 3.0.

## 6.7.9.18 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

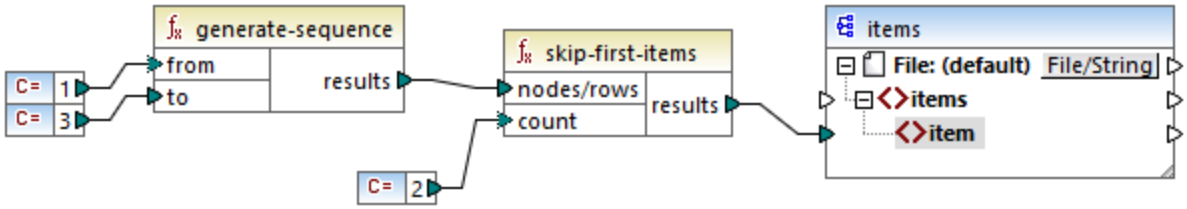
Built-in, C++, C#, Java, XQuery, XSLT 2.0, XSLT 3.0.

### Paramètres

| Nom | Description |
|---|---|
| **node/rows** | Cette entrée doit recevoir une connexion depuis un item de mappage qui fournit une séquence 803 de zéros ou plus de valeurs. Par exemple, la connexion peut provenir d'un item XML de source , un champ CSV, un enregistrement de base de données, etc. |
| **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.

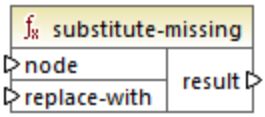



Puisque l'argument **count** est défini sur **2**, les premiers deux items sont sautés et les items restants sont transférés à la cible. Par conséquent, la sortie de mappage est la suivante (à l'exception des déclarations XML et de schéma) :

```
<items>
   <item>3</item>
</items>
```

## 6.7.9.19 substitute-missing

Cette fonction est une combinaison pratique de la fonction de exists[599] et if-else condition[448]. 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

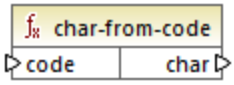
| Nom | Description |
|---|---|
| **node** | Cette entrée doit recevoir une connexion depuis un item de mappage qui fournit une séquence[803] de zéros ou plus de valeurs. Par exemple, la connexion peut provenir d'un item XML de source , un champ CSV, un enregistrement de base de données, etc. |
| **replace-with** | Cette entrée doit recevoir une connexion depuis un item de mappage qui fournit la valeur de remplacement |

## 6.7.10 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.

### 6.7.10.1 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`** 628.




#### Langages

Built-in, C++, C#, Java, XQuery, XSLT 2.0, XSLT 3.0.

#### Paramètres

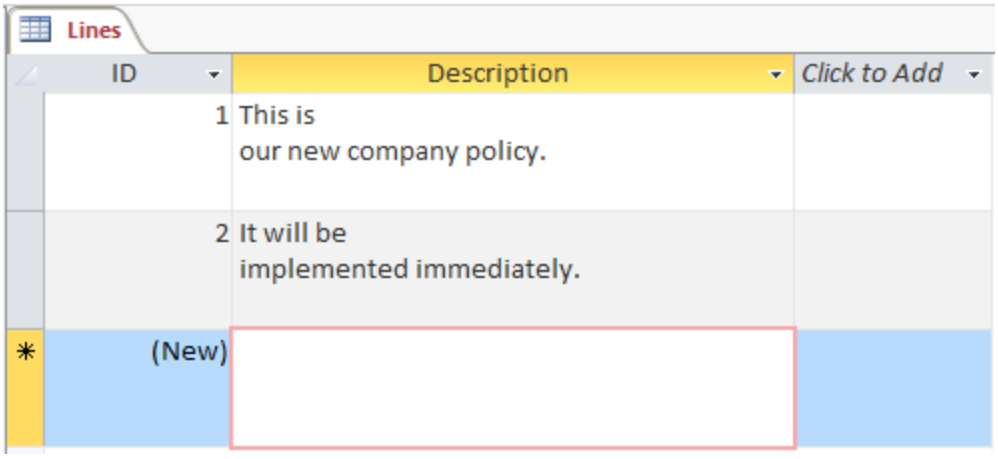
| Nom | Description |
|---|---|
| **code** | La valeur Unicode, en tant que nombre décimal. |

#### Exemple 1

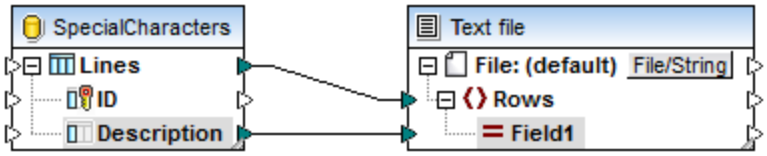
Conformément aux graphiques disponibles sur le site web Unicode (https://www.unicode.org/charts/), le point d'exclamation a la valeur hexadécimale de **`0021`**. La valeur correspondante en format décimal est **33**. C'est pourquoi, si vous fournissez **`33`** en tant qu'argument à la fonction **`char-from-code`**, 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 :

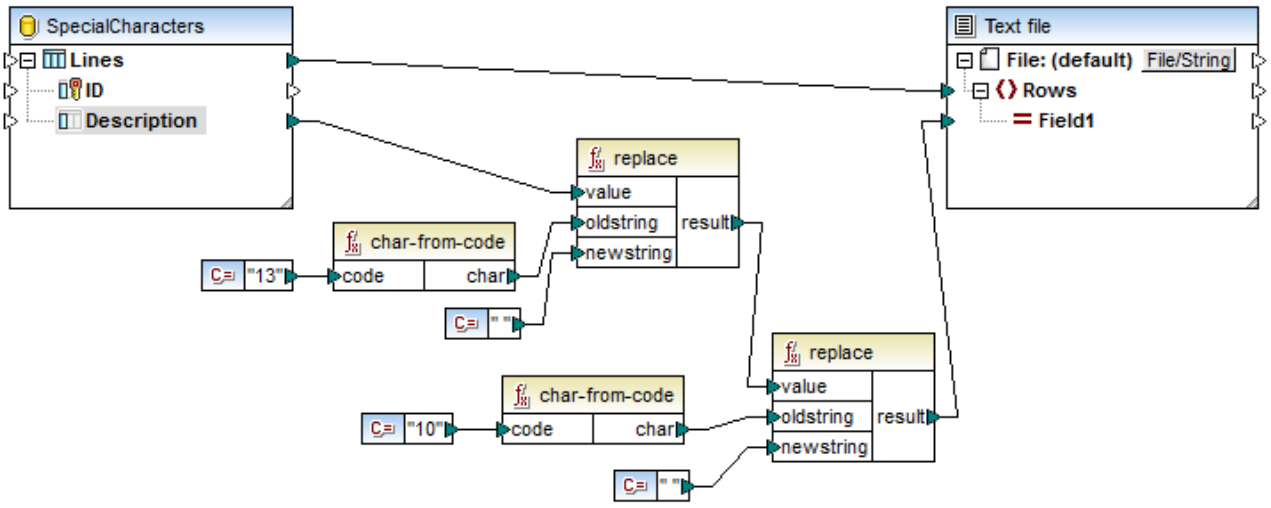



Néanmoins, étant donné que chaque ligne "Description" dans Access contient plusieurs lignes séparées par des caractères CR/LF, la sortie de mappage contient aussi des sauts de ligne, ce qui n'est pas le résultat intenté :

```
1  "This is
2  our new company policy."
3  "It will be
4  implemented immediately."
5
```

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.

```
1    This is  our new company policy.
2    It will be  implemented immediately.
3
```

## 6.7.10.2 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

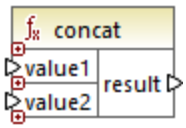
| Nom | Description |
|---|---|
| **char** | La valeur de string d'entrée. |

### Exemple

Si l'entrée**char** est le caractère **$** (signe de dollar), la fonction retourne **36** (qui est la valeur Unicode décimale pour ce caractère).

## 6.7.10.3 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 625 puis fournir le résultat en tant qu'entrée dans la fonction **`concat`**.

### Langages

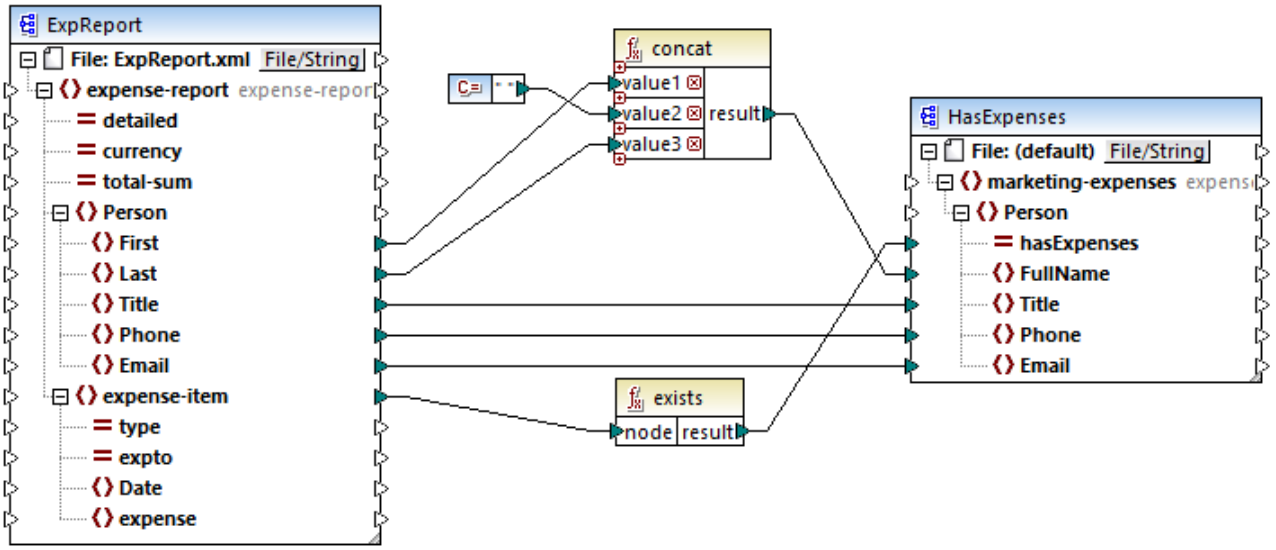
Built-in, C++, C#, Java, XQuery, XSLT 1.0, XSLT 2.0, XSLT 3.0.

### Paramètres

| Nom | Description |
|---|---|
| **value1** | La première valeur d'entrée. |
| **value2** | La deuxième valeur d'entrée. |
| **valueN** | La *N* valeur d'entrée. |

### Exemple

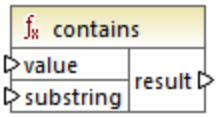
Dans le mappage illustré ci-dessous, la fonction **`concat`** réunit le prénom, la constante " " et le nom de famille. La valeur de retour est ensuite écrite dans l'item de cible **FullName**. Le mappage de cette fonction est disponible dans le chemin suivant :
**<Documents>\Altova\MapForce2024\MapForceExamples\HasMarketingExpenses.mfd**.




*HasMarketingExpenses.mfd*

## 6.7.10.4 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

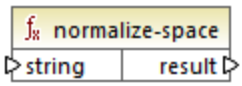
| Nom | Description |
|---|---|
| **value** | La valeur d'entrée (c'est à dire la "pile de foin"). |
| **substring** | Le sous-string à chercher (c'est à dire l'"aiguille"). |

### Exemple

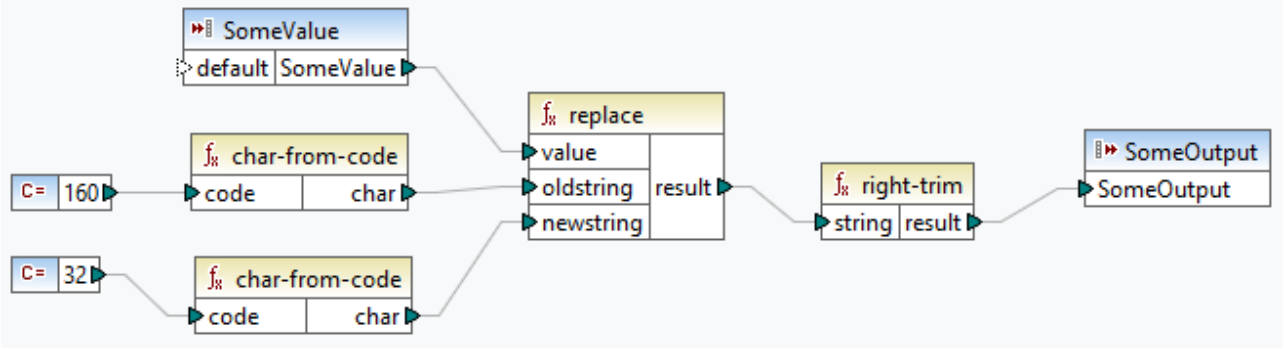
Si la **value** d'entrée est "category" et **substr** est "cat", la fonction retourne **true**.

## 6.7.10.5 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

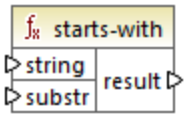
| Nom | Description |
|---|---|
| **string** | Le string d'entrée à normaliser. |

### Exemple

Si le string d'entrée est `The quick  brown fox`, la fonction retourne `The quick brown fox`.

## 6.7.10.6 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

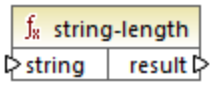
| Nom | Description |
|---|---|
| **string** | Type |
| **substr** | Le sous-string à vérifier. |

### Exemple

Si le **string** d'entrée est `category` et **substr** est `cat`, la fonction retourne **true**.

## 6.7.10.7 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

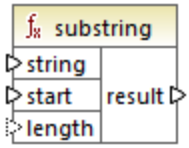
| Nom | Description |
|---|---|
| **string** | Type |

### Exemple

Si le string d'entrée est `car`, la fonction retourne **3**. Si le string d'entrée est string vide, la fonction retourne **0**.

## 6.7.10.8 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

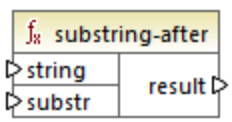
| Nom | Description |
|---|---|
| **string** | Type |
| **start** | Spécifie la position de démarrage (index) à partir de laquelle le sous-string.devrait être extrait. Le premier index est **1**. |
| **length** | Optionnel Spécifie le nombre de caractères à extraire. Si le paramètre **length** n'est 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` .

## 6.7.10.9 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

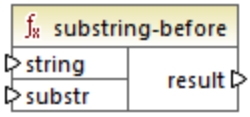
| Nom | Description |
|---|---|
| **string** | Le string d'entrée |
| **substr** | Le sous-string. Tout caractère après la première occurrence de **substr** est le résultat de la fonction. |

### 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`.

## 6.7.10.10 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

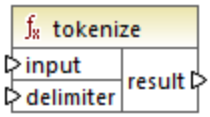
| Nom | Description |
|---|---|
| **string** | Type |
| **substr** | Le sous-string. Tout caractère avant la première occurrence de **substr** est le résultat de la fonction. |

### 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`.

## 6.7.10.11 tokenize

Partage le string d'entrée en une séquence de strings en utilisant le délimiteur fourni en tant qu'argument.




### Langages

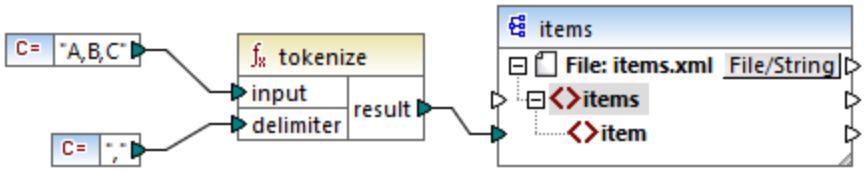
Built-in, C++, C#, Java, XQuery, XSLT 2.0, XSLT 3.0.

### Paramètres

| Nom | Description |
|---|---|
| **input** | Type |
| **delimiter** | Le délimiteur à utiliser. |

### Exemple

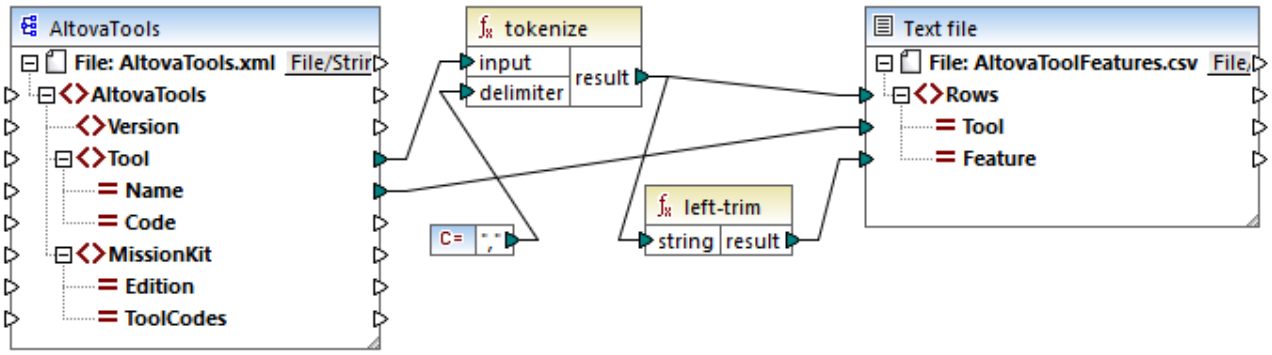
Si le string d'entrée est `A,B,C` et le délimiteur est `,` alors la fonction retourne une séquence de trois strings : `A`, `B` et `C`.




Dans le mappage fictif illustré ci-dessus, le résultat de la fonction est une séquence de strings. Conformément au mappage général rules 803, 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>
```

Pour un exemple plus complexe, voir le mappage **tokenizeString1.mfd** disponible dans le dossier **<Documents>\Altova\MapForce2024\MapForceExamples**.




*tokenizeString1.mfd*

Un fragment du fichier XML de source est affiché ci-dessous. L'élément **Tool** a deux attributs : **Name** et **Code**. Les données d'élément **Tool** consistent en un texte délimité par virgule.

```
<?xml version="1.0" encoding="UTF-8"?>
<AltovaTools xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="AltovaTools.xsd">
   <Version>2010</Version>
   <Tool Name="XMLSpy" Code="XS">XML editor, XSLT editor, XSLT debugger, XQuery editor,
XQuery debugger, XML Schema / DTD editor, WSDL editor, SOAP debugger</Tool>
   <Tool Name="MapForce" Code="MF">Data integration, XML mapping, database mapping, text
conversion, EDI translator, Excel mapping, XBRL mapping, Web services</Tool>
   <Tool Name="StyleVision" Code="SV">Stylesheet designer, electronic forms, XSLT design,
XSL:FO design, database reporting, XBRL rendering</Tool>
   <Tool Name="UModel" Code="UM">UML modeling tool, code generation, reverse engineering,
UML, BPMN, SysML, project documentation, XMI interchange</Tool>
   <Tool Name="DatabaseSpy" Code="DS">Multi-database tool, SQL auto-completion, graphical
database design, table browser, content editor, database comparison tool</Tool>
   <!-- ... -->
</AltovaTools>
```

Ce que fait le mappage :

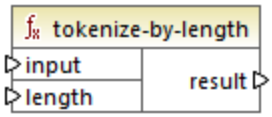
- La fonction **tokenize** reçoit des données provenant de l'item source **Tool** et utilise le délimiteur virgule , pour partager les données en blocs séparés. Le premier bloc est "XML editor", le deuxième est "XSLT editor", etc.
- Pour chaque bloc résultant de la fonction **tokenize**, une nouvelle ligne est générée dans la cible. Cela se produit grâce à la connexion entre le résultat de la fonction et l'item **Rows** dans le composant de cible.
- Le résultat de la fonction **tokenize** est aussi mappé dans la fonction **left-trim**, qui supprime l'espace de début de chaque bloc.
- Le résultat de la fonction **left-trim** (chaque bloc) est écrite dans l'item **Feature** du composant cible.
- Le fichier de sortie du composant cible a été défini en tant que fichier CSV (**AltovaToolFeatures.csv**) et comporte un délimiteur de champ sous la forme d'un point-virgule (double cliquer le composant pour voir les paramètres.

Le résultat du mappage est que, pour chaque bloc créé par la fonction **tokenize**, une nouvelle ligne est créée dans le fichier CSV cible. Un fragment de la sortie de mappage ressemble à :

```
Tool;Feature
XMLSpy;XML editor
XMLSpy;XSLT editor
XMLSpy;XSLT debugger
XMLSpy;XQuery editor
XMLSpy;XQuery debugger
XMLSpy;XML Schema / DTD editor
XMLSpy;WSDL editor
XMLSpy;SOAP debugger
MapForce;Data integration
MapForce;XML mapping
MapForce;database mapping
MapForce;text conversion
MapForce;EDI translator
MapForce;Excel mapping
```

## 6.7.10.12 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

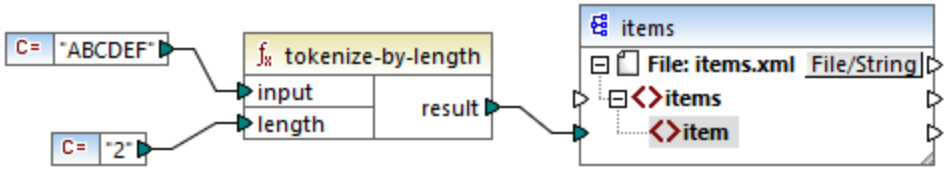
Built-in, C++, C#, Java, XQuery, XSLT 2.0, XSLT 3.0.

### Paramètres

| Nom | Description |
|---|---|
| **input** | Type |
| **length** | Détermine la longueur de chaque string dans la séquence générée des strings. |

### Exemple

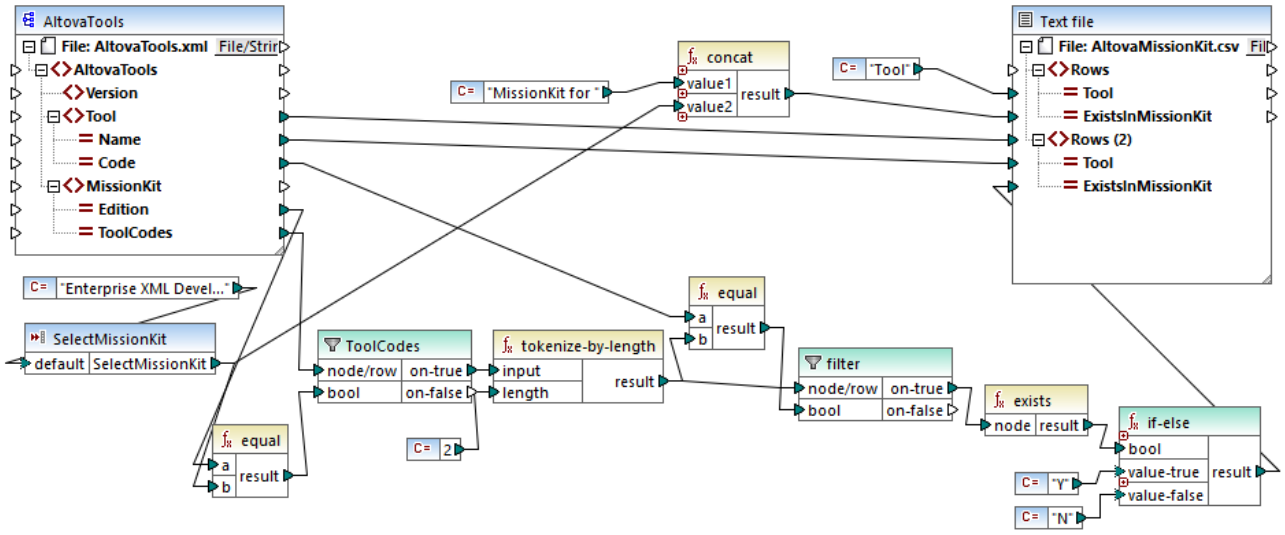
Si le string d'entrée est **ABCDEF** et que la longueur est **2**, alors la fonction retourne une séquence de trois strings : **AB**, **CD**, et **EF**.




Dans le mappage fictif illustré ci-dessus, le résultat de la fonction est une séquence de strings. Conformément au mappage général rules 803, 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>
```

Pour un exemple plus complexe, voir le mappage **tokenizeString2.mfd** disponible dans le dossier **<Documents>\Altova\MapForce2024\MapForceExamples**.




*tokenizeString2.mfd*

Le fichier de source XML est affiché ci-dessous et est le même que celui utilisé dans l'exemple précédent. L'élément **MissionKit** a deux attributs : **Edition** et **ToolCodes**, mais pas de contenu d'élément MissionKit. Veuillez noter que le contenu XML non pertinent pour cet exemple a été supprimé.

```
<?xml version="1.0" encoding="UTF-8"?>
<AltovaTools xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="AltovaTools.xsd">
   <Version>2010</Version>
   <Tool Name="XMLSpy" Code="XS"><!--...--></Tool>
   <Tool Name="MapForce" Code="MF"><!--...--></Tool>
```

```
   <Tool Name="StyleVision" Code="SV"><!-...--></Tool>
   <Tool Name="UModel" Code="UM"><!-...--></Tool>
   <Tool Name="DatabaseSpy" Code="DS"><!-...--></Tool>
   <Tool Name="DiffDog" Code="DD"><!-...--></Tool>
   <Tool Name="SchemaAgent" Code="SA"><!-...--></Tool>
   <Tool Name="SemanticWorks" Code="SW"><!-...--></Tool>
   <Tool Name="Authentic" Code="AU"><!-...--></Tool>
   <MissionKit Edition="Enterprise Software Architects" ToolCodes="XSMFSVUMDSDDSASW"/>
   <MissionKit Edition="Professional Software Architects" ToolCodes="XSMFSVUMDS"/>
   <MissionKit Edition="Enterprise XML Developers" ToolCodes="XSMFSVDDSASW"/>
   <MissionKit Edition="Professional XML Developers" ToolCodes="XSMFSV"/>
</AltovaTools>
```

L'objectif du mappage est de générer une liste montrant quels outils Altova font partie des éditions MissionKit respectives

Comment fonctionne le mappage :

- Le composant d'entrée **SelectMissionKit** agit comme un paramètre dans le mappage, il reçoit sa valeur par défaut depuis une constante, dans ce cas "Enterprise XML Developers".
- La fonction **equal** compare l'édition fournie en tant que paramètre avec l'item**Edition** depuis le fichier XML de source et transmet le résultat vers le paramètre **bool** du filtre **ToolCodes**.
- L'entrée **node/row** du filtre **ToolCodes** est fourni par l'item **ToolCodes** du fichier de source. La valeur pour l'édition "Enterprise XML Developers" est : **XSMFSVDDSASW**.
- La valeur **XSMFSVDDSASW** est transmise dans le paramètre **on-true**, puis vers le paramètre **input** de la fonction **tokenize-by-length** .
- La fonction**tokenize-by-length** partage la valeur **XSMFSVDDSASW** en plusieurs blocs de deux caractères chacun. Le paramètre **length** est **2** ; c'est pourquoi 6 blocs sont créés.
- Chaque bloc est comparé à la valeur **Code** de 2 caractères depuis le fichier de source (dont il y a 9 items au total). Le résultat de la comparaison (true/false) est transmis dans le paramètre **bool** du filtre. Veuillez noter que *tous* les blocs produits par la fonction **tokenize-by-length** sont transmis dans le paramètre **node/row** du filtre.
- Les fonctions **exists** vérifient maintenant les nœuds existants/non-existants transmis vers le paramètre **on-true** du composant de filtre. Les nœuds existants sont ceux dans lesquels *il y a une correspondance* entre le bloc **ToolCodes** et la valeur de **Code**. Des nœuds non-existants se trouvent là où il n'y avait pas de bloc **ToolCodes** pour correspondre à une valeur **Code**.
- Chaque résultat **bool** de la fonction **exists** est transmis dans le composant **if-else** qui génèer un "Y" dans la cible si le nœud existe, ou un "N" si le nœud n'existe pas.

Résultat du mappage :

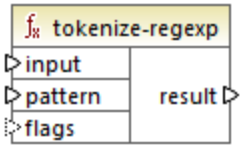
```
Tool;MissionKit for Enterprise XML Developers
XMLSpy;Y
MapForce;Y
StyleVision;Y
UModel;N
DatabaseSpy;N
DiffDog;Y
SchemaAgent;Y
```

```
SemanticWorks;Y
Authentic;N
```

## 6.7.10.13 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

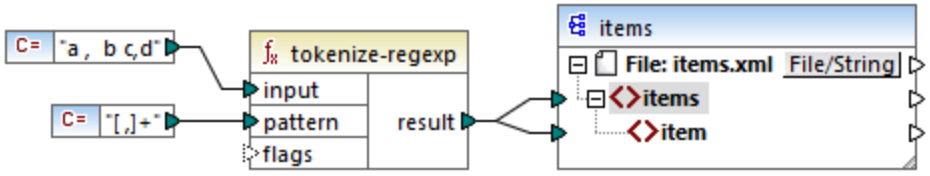
Built-in, C++, C#, Java, XQuery, XSLT 2.0, XSLT 3.0.

### 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 543. |
| **flags** | Paramètre optionnel. Fournit l'expression régulière flags 545 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 :

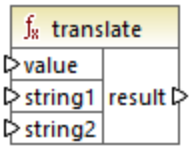
- Il correspond à un des caractères dans la classe de caractère `[ ,]`. Ainsi, un partage se produira à chaque fois qu'une virgule ou qu'une espace est rencontrée dans le string d'entrée.
- Le quantificateur `+` spécifie qu'une ou plusieurs occurrences de la classe du caractère précédent doivent trouver 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>
```

## 6.7.10.14 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.

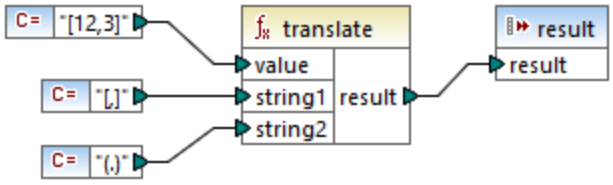
### Paramètres

| 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** `(.)`

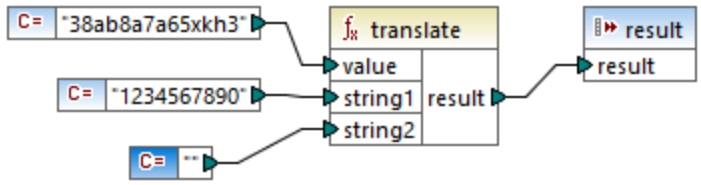
Veuillez noter que les deux **string1** et **string2** ont le même nombre de caractères. Pour chaque caractère dans **string1**, le caractère équivalent à la même position depuis **string2** sera utilisé en tant que remplacement. Par conséquent, le remplacement suivant aura lieu :

- 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 :

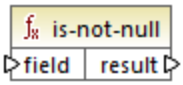
```
abaaxkh
```

## 6.7.11 db

La bibliothèque **db** contient des fonctions qui vous permettent de définir les résultats de mappage lorsque vous rencontrez des champs null dans des bases de données.

### 6.7.11.1 is-not-null

Retourne **false** si le champ est null, sinon, retourne **true**.




#### Langages

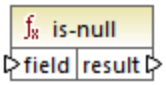
Built-in, C++, C#, Java.

#### Paramètres

| Nom | Description |
|---|---|
| **field** | Le champ de base de données. |

### 6.7.11.2 is-null

Retourne **true** si le champ est null, sinon, retourne **false**.



### Langages

Built-in, C++, C#, Java.

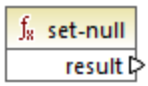
### Paramètres

| Nom | Description |
|---|---|
| **field** | Le champ de base de données. |

## 6.7.11.3 set-null

Définit un champ de base de données sur null. Cette fonction écrasera aussi une valeur par défaut avec null. Si elle est connectée à autre chose, c'est à dire, n'est pas un champ de base de données, elle se comportera comme une séquence vide. Veuillez noter les points suivants :

- Une connexion **`set-null`** à une fonction différente résultera généralement dans le fait que l'autre fonction ne sera pas être appelée du tout. Une connexion **`set-null`** à une fonction de séquence, par ex. **`count`**, appellera la fonction avec une séquence vide.
- Une connexion **`set-null`** à des filtres et des conditions if-else fonctionnent comme prévu, les champs sont définis sur null. Pour les filtres cela signifie l'entrée "node/row".
- L'utilisation **`set-null`** en tant qu'une entrée pour un élément `simpleType` ne créera pas cet élément dans le composant cible

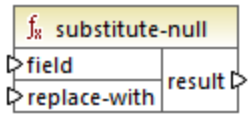



### Langages

Built-in, C++, C#, Java.

## 6.7.11.4 substitute-null

Retourne le champ lui-même s'il n'est pas null ; sinon, **replace-with** est retourné.




### Langages

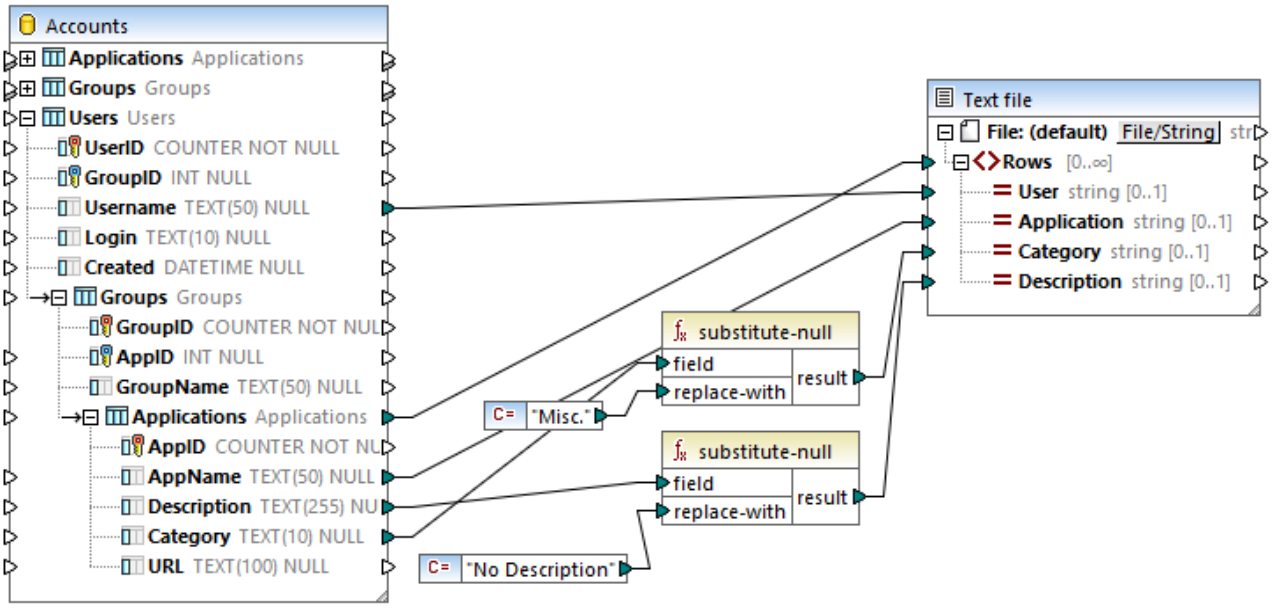
Built-in, C++, C#, Java.

## Paramètres

| Nom | Description |
|---|---|
| **field** | Le champ de base de données. |
| **replace-with** | La valeur de remplacement. |

## Exemple

Le mappage ci-dessous montre un exemples de la fonction **`substitute-null`** en cours d'utilisation. Ce mappage est appelé **DB-ApplicationList.mfd** et est disponible dans le dossier **<Documents>\Altova\MapForce2024\MapForceExamples**.




Le mappage lit des données depuis une base de données SQLite qui contient un table "Applications".

| | AppID | AppName | Description | Category | URL |
|---|---|---|---|---|---|
| 1 | 1 | Altova MapForce | The premier data mapping tool. | IDE | www.altova.com/xmlspy |
| 2 | 2 | Notepad | *[NULL]* | *[NULL]* | *[NULL]* |

La première fonction vérifie si le champ **Category** est null dans la table "Applications". Puisque ce champ est null pour l'application Notepad, la valeur de substitution "Misc" est mappée dans l'item **Category** du fichier de texte de cible.

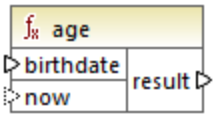
La seconde fonction vérifie si le champ **Description** est null. À nouveau, ce champ est null pour l'application Notepad, la valeur de substitution "No description" est donc mappée dans l'item **Description** du fichier de cible.

## 6.7.12 lang | datetime functions

Les fonctions de date et heure depuis la bibliothèque **lang** peuvent être utilisées pour manipuler des dates, heures et durées. Contrairement aux fonctions date et heure depuis la bibliothèque **core**, ces fonctions sont disponibles uniquement lorsque vous sélectionnez les langages Built-in, Java, C#, ou C++.

### 6.7.12.1 age

Retourne le nombre des années pleines qui se sont écoulées entre la date de naissance fournie en tant que l'argument et maintenant.




#### Langages

Built-in, C++, C#, Java.

#### Paramètres

| Nom | Type | echo Hello, World! |
|---|---|---|
| **birthdate** | `xs:date` | Obligatoire. Fournit la fate de naissance en tant que valeur `xs:date`. |
| **now** | `xs:date` | Paramètre optionnel. Le défaut est la date de système actuel. Si une valeur est mappée à l'argument now, la fonction retourne la différence entre la date de naissance et maintenant, en années pleines. |

### 6.7.12.2 convert-to-utc

Convertit la valeur d'heure fournie en tant qu'argument au UTC (Coordinated Universal Time). La fonction prend en compte le composant de fuseau horaire (par exemple, "+5:00").


#### Langages

Built-in, C++, C#, Java.

### Paramètres

| Nom | Type | echo Hello, World! |
|---|---|---|
| **time** | **xs:dateTime** | Fournit la valeur `xs:dateTime` à convertir. |

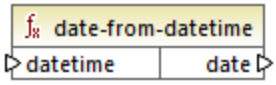
### Exemple

Si la valeur d'entrée est **2001-12-17T09:30:02+05:00**, le résultat de la fonction est **2001-12-17T04:30:02**.

Si la valeur d'entrée est **2001-12-17T09:30:02Z**, le résultat de la fonction est **2001-12-17T09:30:02**. Dans ce cas, aucune conversion ne s'est produite, parce que le "Z" en fin de ligne a déjà défini cette heure pour être l'heure "Zero" (ou "Zulu"), qui est le même que l'UTC.

## 6.7.12.3 date-from-datetime

Retourne la partie *date* provenant de la valeur `xs:dateTime` fournie en tant qu'rgument. La partie *time* est définie sur zéro. Le fuseau horaire n'est pas modifié.




### Langages

Built-in, C++, C#, Java.

### Paramètres

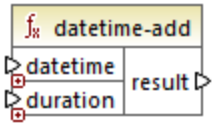
| Nom | Type | echo Hello, World! |
|---|---|---|
| **datetime** | **xs:dateTime** | Fournit la valeur `xs:dateTime` à traiter. |

### Exemple

Si la valeur d'entrée est **2001-12-17T09:30:02+05:00**, le résultat de la fonction est **2001-12-17+05:00**.

## 6.7.12.4 datetime-add

Retourne une valeur `xs:dateTime` obtenue en ajoutant une durée (le second argument) dans un datetime (le premier argument).




### Langages

Built-in, C++, C#, Java.

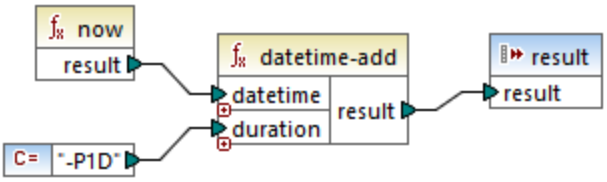
### Paramètres

| Nom | Type | echo Hello, World! |
|---|---|---|
| **datetime** | `xs:dateTime` | Fournit la valeur `xs:dateTime` à utiliser en tant qu'entrée. |
| **duration** | `xs:duration` | Fournit la valeur `xs:duration`.<br><br>Un exemple de durée est **`P1Y2M3DT04H05M59S`**, où :<br><br>• "P" est de désignateur de la période et est obligatoire ;<br>• Le reste des caractères dénote, dans cet ordre : 1 an, 2 mois, 3 jours, T (Désignateur de temps), 04 heures, 05 minutes, 59 secondes.<br><br>Si le caractère "moins" apparaît avant le désignateur "P", cela indique une durée négative, par exemple : **`-P1D`**. |

### Exemple

Partons du principe que la valeur d'entrée **datetime** est **`2001-12-17T09:30:02+05:00`**. Si la **durée** est **`P10D`** (10 jours), le résultat de la fonction est **`2001-12-27T09:30:02+05:00`**.

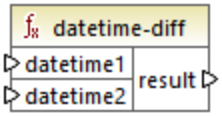
Pour obtenir la date d'hier, connecter la fonction **now** à l'entrée **datetime** . Dans le mappage ci-dessous, la période **`-P1D`** signifie "moins 1 jour", pour que le mappage retourne la date de la veille.




## 6.7.12.5 datetime-diff

Retourne la durée obtenue en soustrayant **datetime2** (second argument) depuis **datetime1** (premier argument). Le résultat peut être mappé dans un type de données string ou durée.




### Langages

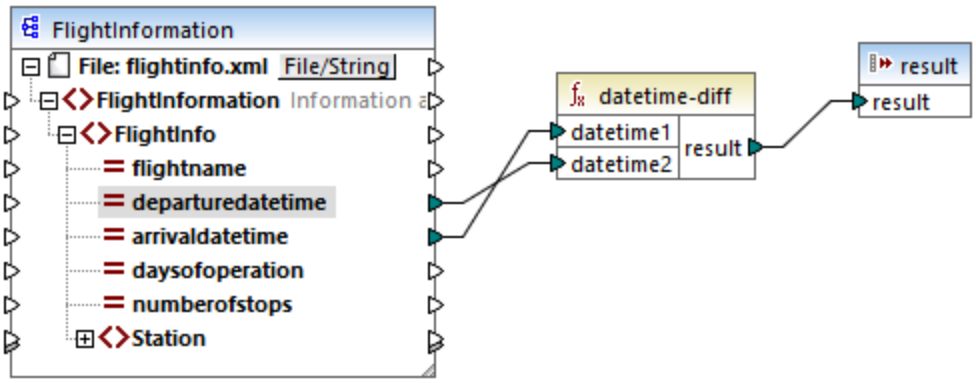
Built-in, C++, C#, Java.

### Paramètres

| Nom | Type | echo Hello, World! |
|---|---|---|
| **datetime1** | `xs:dateTime` | Fournit la première valeur `xs:dateTime`. |
| **datetime2** | `xs:duration` | Fournit la seconde valeur `xs:dateTime`. |

### Exemple

Dans le mappage illustré ci-dessous, la fonction **`datetime-diff`** soustrait le datetime de départ de vol **`2001-12-17T09:30:02+05:00`** depuis le datetime d'arrivée **`2001-12-17T19:30:02+05:00`**. Veuillez noter que le datetime d'arrivée est la valeur supérieure, il est donc connecté à la première entrée de la fonction.

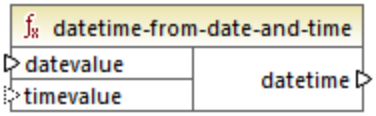



La sortie de mappage est la différence entre les deux (une période de 10 heures) :

```
PT10H
```

## 6.7.12.6 datetime-from-date-and-time

Retourne une valeur `xs:dateTime` contruite depuis une valeur `xs:date` (premier argument) et une valeur `xs:time` (second argument). Le résultat peut être mappé dans un string ou un type de données `xs:dateTime`.




### Langages

Built-in, C++, C#, Java.

### Paramètres

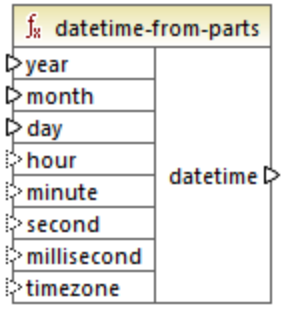
| Nom | Type | echo Hello, World! |
|---|---|---|
| **datevalue** | `xs:date` | Fournit une valeur de type `xs:date` |
| **timevalue** | `xs:time` | Fournit une valeur de type `xs:time` |

### Exemple

Dans le premier argument est `2012-06-29` et le second argument est `11:59:55`, la fonction retourne `2012-06-29T11:59:55`.

## 6.7.12.7 datetime-from-parts

Retourne une valeur de type `xs:dateTime` construite depuis n'importe quelle combinaison des parties suivantes en tant qu'arguments : annéer, mois, jour, heure, minute, seconde, milliseconde et fuseau horaire. Cette fonction normalise automatiquement les paramètres fournis. Par exemple, le 32 janvier sera changé automatiquement en 1er février.




### Langages

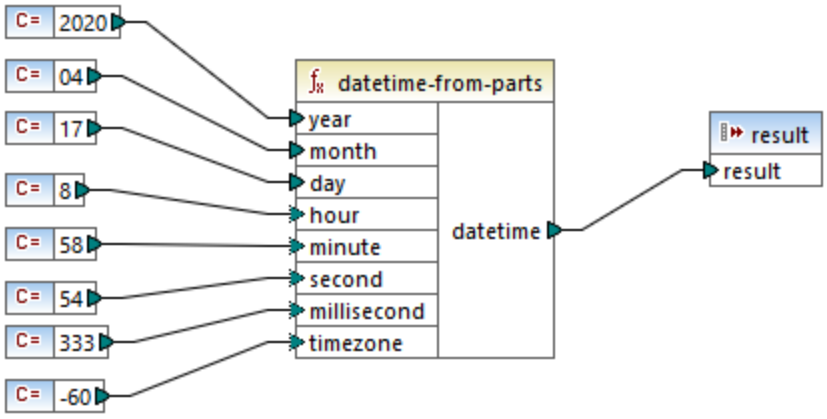
Built-in, C++, C#, Java.

## Paramètres

| Nom | Type | echo Hello, World! |
|---|---|---|
| **year** | `xs:int` | Fournit l'année. |
| **month** | `xs:int` | Fournit le mois. |
| **day** | `xs:int` | Fournit le jour du mois. |
| **hour** | `xs:int` | Optionnel Fournit l'heure. |
| **minute** | `xs:int` | Optionnel Fournit la minute. |
| **second** | `xs:int` | Optionnel Fournit la seconde. |
| **millisecond** | `xs:decimal` | Optionnel Fournit la milliseconde. |
| **timezone** | `xs:int` | Optionnel Fournit le fuseau horaire, en minutes. Cette valeur peut être négative. |

## Exemple

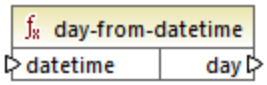
Le mappage suivant construit une valeur `xs:dateTime` provenant de parties qui sont fournies par des constantes.




La sortie de mappage est **`2020-04-17T08:58:54.333-01:00`**.

## 6.7.12.8 day-from-datetime

Retourne le jour en tant que valeur d'entier, depuis la valeur `xs:dateTime` fournie en tant qu'argument .




### Langages

Built-in, C++, C#, Java.

### Paramètres

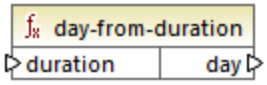
| Nom | Type | echo Hello, World! |
|---|---|---|
| **datetime** | `xs:dateTime` | Fournit la valeur d'entrée de type `xs:dateTime`. |

### Exemple

Si **datetime** est `2001-12-17T10:30:03+01:00`, la fonction retourne `17`.

## 6.7.12.9 day-from-duration

Retourne le jour en tant que valeur d'entier, depuis la valeur `xs:duration` fournie en tant qu'argument .




### Langages

Built-in, C++, C#, Java.

### Paramètres

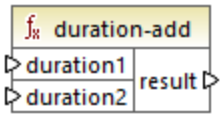
| Nom | Type | echo Hello, World! |
|---|---|---|
| **duration** | `xs:duration` | Fournit la valeur d'entrée de type `xs:duration`. |

### Exemple

Si **duration** est `P1Y2M3DT10H30M`, la fonction `day-from-duration` retourne `3`.

## 6.7.12.10 duration-add

Retourne la durée obtenue en ajoutant deux durées.




### Langages

Built-in, C++, C#, Java.

### Paramètres

| Nom | Type | echo Hello, World! |
|---|---|---|
| **duration1** | `xs:duration` | Fournit la première valeur d'entrée de type `xs:duration`. |
| **duration2** | `xs:duration` | Fournit la deuxième valeur d'entrée de type `xs:duration`. |

### Exemple

Si la première durée est **`P0Y0M3DT03H0M`** (3 jours et 3 heures) et la seconde durée est **`P0Y0M3DT01H0M`** (3 jours et 1 heure), la fonction retourne **`P6DT4H`** (6 jours et 4 heures).

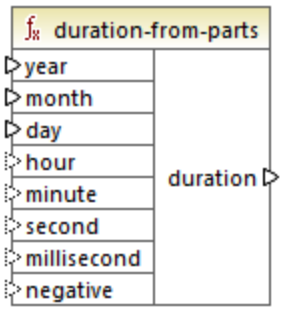
## 6.7.12.11 duration-from-parts

Retourne une valeur de type `xs:duration` calculée en combinant les parties suivantes fournies en tant qu'arguments : année, mois, heure, minute, seconde, milliseconde, négative.

Un exemple de durée est **`P1Y2M3DT04H05M59S`**, où :

- "P" est de désignateur de la période et est obligatoire ;
- Le reste des caractères dénote, dans cet ordre : 1 an, 2 mois, 3 jours, T (Désignateur de temps), 04 heures, 05 minutes, 59 secondes.

Si le caractère "moins" apparaît avant le désignateur "P", cela indique une durée négative, par exemple : **`-P1D`**.




## Langages

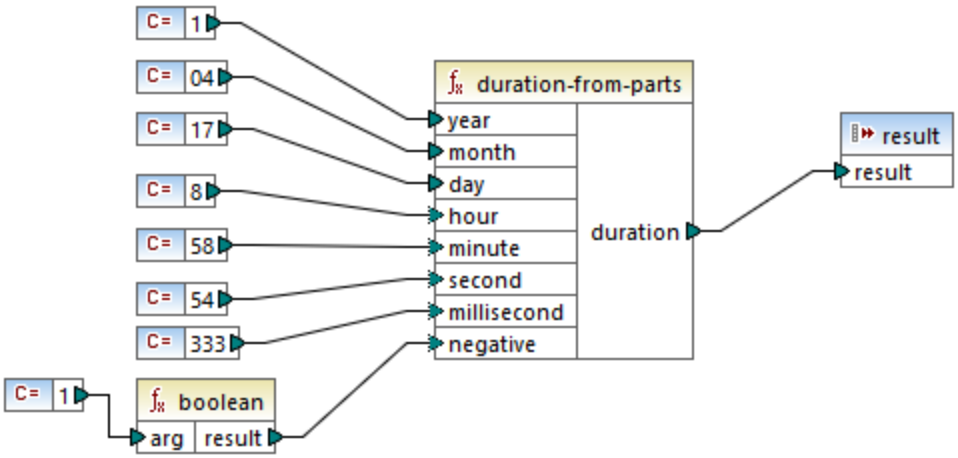
Built-in, C++, C#, Java.

## Paramètres

| Nom | Type | echo Hello, World! |
|---|---|---|
| **year** | `xs:int` | Fournit l'année. |
| **month** | `xs:int` | Fournit le mois. |
| **day** | `xs:int` | Fournit le jour du mois. |
| **hour** | `xs:int` | Optionnel Fournit l'heure. |
| **minute** | `xs:int` | Optionnel Fournit la minute. |
| **second** | `xs:int` | Optionnel Fournit la seconde. |
| **millisecond** | `xs:decimal` | Optionnel Fournit la milliseconde. |
| **negative** | `xs:boolean` | Optionnel Doit être **true** pour une durée négative ; **false** sinon. |

## Exemple

Le mappage suivant génère une durée négative de 1 an, 4 mois, 17 jours, 8 heures, 58 minutes, et 54.333 secondes.




La sortie de mappage est **`-P1Y4M17DT8H58M54.333S`**.

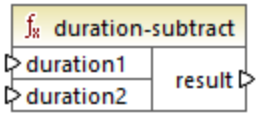
## 6.7.12.12 duration-subtract

Retourne la valeur `xs:duration` obtenue en soustrayant **duration2** depuis **duration1**.

Un exemple de durée est **`P1Y2M3DT04H05M59S`**, où :

- "P" est de désignateur de la période et est obligatoire ;
- Le reste des caractères dénote, dans cet ordre : 1 an, 2 mois, 3 jours, T (Désignateur de temps), 04 heures, 05 minutes, 59 secondes.

Si le caractère "moins" apparaît avant le désignateur "P", cela indique une durée négative, par exemple : **`-P1D`**.




### Langages

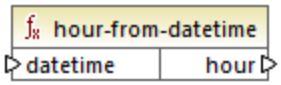
Built-in, C++, C#, Java.

### Exemple

Si **duration1** est **`P0Y0M0DT05H07M`** (5 heures et 7 minutes) et **duration2** est **`PT1H`** (1 heure), la fonction retourne **`PT4H7M`** (4 heures et 7 minutes).

## 6.7.12.13 hour-from-datetime

Retourne l'heure, en tant qu'une valeur d'entier depuis la valeur `xs:dateTime` fournie en tant qu'argument.




### Langages

Built-in, C++, C#, Java.

### Paramètres

| Nom | Type | echo Hello, World! |
|---|---|---|
| **datetime** | **xs:dateTime** | Fournit la valeur d'entrée de type `xs:dateTime`. |

### Exemple

Si **datetime** est **`2001-12-17T09:30:02+05:00`**, la fonction retourne **`9`**.

## 6.7.12.14 hour-from-duration

Retourne l'heure, en tant qu'une valeur d'entier depuis la valeur `xs:duration` fournie en tant qu'argument.


### Langages

Built-in, C++, C#, Java.

### Paramètres

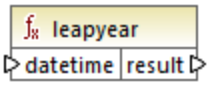
| Nom | Type | echo Hello, World! |
|---|---|---|
| **duration** | **xs:duration** | Fournit la valeur d'entrée de type `xs:duration`. |

### Exemple

Si **duration** est **`P0Y0M0DT05H07M`**, la fonction retourne **`5`**.

## 6.7.12.15 leapyear

Retourne Boolean **true** si l'année de la valeur `xs:dateTime` fournie en tant qu'argument est une année bissextile; **false** sinon.




### Langages

Built-in, C++, C#, Java.

### Paramètres

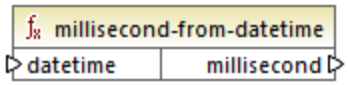
| Nom | Type | echo Hello, World! |
|---|---|---|
| **datetime** | `xs:dateTime` | Fournit la valeur d'entrée de type `xs:dateTime`. |

### Exemple

Si **datetime** est `2020-04-17T09:30:02+02:00`, la fonction retourne **true**, puisque l'année 2020 est une année bissextile.

## 6.7.12.16 millisecond-from-datetime

Retourne les millisecondes, en tant qu'une valeur `xs:decimal`, depuis la valeur `xs:dateTime` fournie en tant qu'argument.




### Langages

Built-in, C++, C#, Java.

### Paramètres

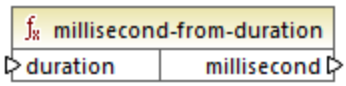
| Nom | Type | echo Hello, World! |
|---|---|---|
| **datetime** | `xs:dateTime` | Fournit la valeur d'entrée de type `xs:dateTime`. |

### Exemple

Si **datetime** est `2001-12-17T09:30:02.544+05:00`, la fonction retourne `544`.

## 6.7.12.17 millisecond-from-duration

Retourne les millisecondes, en tant qu'une valeur `xs:decimal` depuis la valeur `xs:duration` fournie en tant qu'argument.




### Langages

Built-in, C++, C#, Java.

### Paramètres

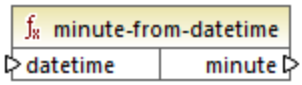
| Nom | Type | echo Hello, World! |
|---|---|---|
| **duration** | `xs:duration` | Fournit la valeur d'entrée de type `xs:duration`. |

### Exemple

Si **duration** est `P0Y0M0DT05H07M02.227S`, la fonction retourne `227`.

## 6.7.12.18 minute-from-datetime

Retourne les minutes, en tant qu'une valeur d'entier depuis la valeur `xs:dateTime` fournie en tant qu'argument.




### Langages

Built-in, C++, C#, Java.

### Paramètres

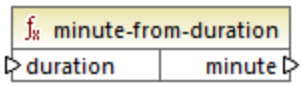
| Nom | Type | echo Hello, World! |
|---|---|---|
| **datetime** | `xs:dateTime` | Fournit la valeur d'entrée de type `xs:dateTime`. |

### Exemple

Si **datetime** est `2001-12-17T09:30:02.544+05:00`, la fonction retourne `30`.

## 6.7.12.19 minute-from-duration

Retourne les minutes, en tant qu'une valeur d'entier depuis la valeur `xs:duration` fournie en tant qu'argument.




### Langages

Built-in, C++, C#, Java.

### Paramètres

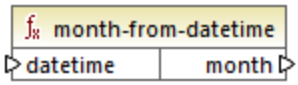
| Nom | Type | echo Hello, World! |
|---|---|---|
| **duration** | `xs:duration` | Fournit la valeur d'entrée de type `xs:duration`. |

### Exemple

Si **duration** est `P0Y0M0DT05H07M02.227S`, la fonction retourne `7`.

## 6.7.12.20 month-from-datetime

Retourne le mois, en tant qu'une valeur d'entier depuis la valeur `xs:dateTime` fournie en tant qu'argument.




### Langages

Built-in, C++, C#, Java.

### Paramètres

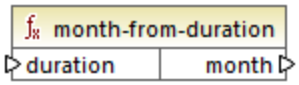
| Nom | Type | echo Hello, World! |
|---|---|---|
| **datetime** | `xs:dateTime` | Fournit la valeur d'entrée de type `xs:dateTime`. |

### Exemple

Si **datetime** est `2001-12-17T09:30:02.544+05:00`, la fonction retourne `12`.

## 6.7.12.21 month-from-duration

Retourne le mois, en tant qu'une valeur d'entier depuis la valeur `xs:duration` fournie en tant qu'argument.




### Langages

Built-in, C++, C#, Java.

### Paramètres

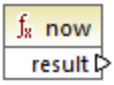
| Nom | Type | echo Hello, World! |
|---|---|---|
| **duration** | `xs:duration` | Fournit la valeur d'entrée de type `xs:duration`. |

### Exemple

Si **duration** est `P0Y04M0DT05H07M02.227S`, la fonction retourne **4**.

## 6.7.12.22 now

Retourne la date et l'heure actuelles (y compris le fuseau horaire), en tant que valeur `xs:dateTime`.

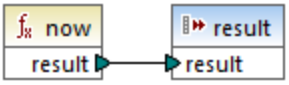



### Langages

Built-in, C++, C#, Java.

### Exemple

Le mappage suivant sort la date et l'heure actuelles. La sortie change à chaque fois que le mappage est exécuté.




Une sortie d'exemple serait `2020-04-17T11:42:34.684+02:00`.

Pour voir un exemple de la manière d'extraire la date d'hier, voir la fonction core | lang | datetime-add 647.

## 6.7.12.23 remove-timezone

Supprime le composant de fuseau horaire depuis le paramètre d'entrée **time** (of type `xs:dateTime`).


### Langages

Built-in, C++, C#, Java.

### Paramètres

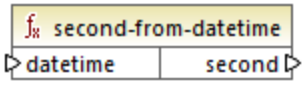
| Nom | Type | echo Hello, World! |
|---|---|---|
| **time** | `xs:dateTime` | Fournit la valeur d'entrée de type `xs:dateTime`. |

### Exemple

Si **time** est `2001-12-17T09:30:02+05:00`, la fonction retourne `2001-12-17T09 30:02`.

## 6.7.12.24 second-from-datetime

Retourne les secondes, en tant qu'une valeur d'entier depuis la valeur `xs:dateTime` fournie en tant qu'argument.




### Langages

Built-in, C++, C#, Java.

### Paramètres

| Nom | Type | echo Hello, World! |
|---|---|---|
| **datetime** | `xs:dateTime` | Fournit la valeur d'entrée de type `xs:dateTime`. |

### Exemple

Si **datetime** est `2001-12-17T09:30:02.544+05:00`, la fonction retourne `2`.

## 6.7.12.25 second-from-duration

Retourne les secondes, en tant qu'une valeur d'entier depuis la valeur `xs:duration` fournie en tant qu'argument.


### Langages

Built-in, C++, C#, Java.

### Paramètres

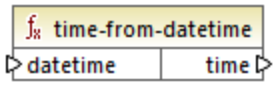
| Nom | Type | echo Hello, World! |
|---|---|---|
| **duration** | `xs:duration` | Fournit la valeur d'entrée de type `xs:duration`. |

### Exemple

Si **duration** est `P0Y04M0DT05H07M02.227S`, la fonction retourne `2`.

## 6.7.12.26 time-from-datetime

Retourne le composant time, en tant qu'une valeur `xs:time` , depuis la valeur `xs:dateTime` fournie en tant qu'argument.




### Langages

Built-in, C++, C#, Java.

### Paramètres

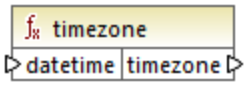
| Nom | Type | echo Hello, World! |
|---|---|---|
| **datetime** | `xs:dateTime` | Fournit la valeur d'entrée de type `xs:dateTime`. |

### Exemple

Si **datetime** est `2001-12-17T09:30:02+05:00`, la fonction retourne `09:30:02+05:00`.

## 6.7.12.27 timezone

Retourne l'offset de fuseau horaire, en minutes, depuis la valeur `xs:dateTime` fournie en tant qu'argument. Retourne 0 pour UTC.




### Langages

Built-in, C++, C#, Java.

### Paramètres

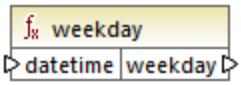
| Nom | Type | echo Hello, World! |
|---|---|---|
| **datetime** | `xs:dateTime` | Fournit la valeur d'entrée de type `xs:dateTime`. |

### Exemple

Si **datetime** est `2001-12-17T09:30:02.544+05:00`, la fonction retourne `300`.

## 6.7.12.28 weekday

Retourne le jour de la semaine depuis la valeur `xs:dateTime` fournie en tant qu'argument. La fonction retournera la valeur **1** pour lundi, la valeur **2** pour mardi, etc.




### Langages

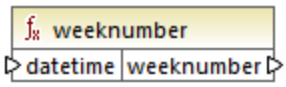
Built-in, C++, C#, Java.

### Paramètres

| Nom | Type | echo Hello, World! |
|---|---|---|
| **datetime** | `xs:dateTime` | Fournit la valeur d'entrée de type `xs:dateTime`. |

## 6.7.12.29 weeknumber

Retourne le numéro de la semaine dans l'année à partir de la valeur `xs:dateTime` fournie en tant qu'argument. La fonction retournera la valeur **1** pour la première semaine de l'année, la valeur **2** pour la deuxième semaine, etc.




### Langages

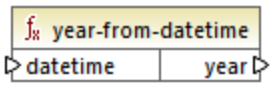
Built-in, C++, C#, Java.

### Paramètres

| Nom | Type | echo Hello, World! |
|---|---|---|
| **datetime** | `xs:dateTime` | Fournit la valeur d'entrée de type `xs:dateTime`. |

## 6.7.12.30 year-from-datetime

Retourne l'année, en tant qu'une valeur d'entier depuis la valeur `xs:dateTime` fournie en tant qu'argument.




### Langages

Built-in, C++, C#, Java.

### Paramètres

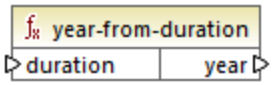
| Nom | Type | echo Hello, World! |
|---|---|---|
| **datetime** | `xs:dateTime` | Fournit la valeur d'entrée de type `xs:dateTime`. |

### Exemple

Si **datetime** est `2001-12-17T09:30:02.544+05:00`, la fonction retourne `2001`.

### 6.7.12.31 year-from-duration

Retourne l'année, en tant qu'une valeur d'entier depuis la valeur `xs:duration` fournie en tant qu'argument.




#### Langages

Built-in, C++, C#, Java.

#### Paramètres

| Nom | Type | echo Hello, World! |
|---|---|---|
| **duration** | `xs:duration` | Fournit la valeur d'entrée de type `xs:duration`. |

#### Exemple

Si **duration** est `P01Y04M0DT05H07M02.227S`, la fonction retourne `1`.

## 6.7.13 lang | file functions

MapForce vous permet de lire des données BLOB (objet binaire large) dans un mappage et puis de le consommer sans devoir changer la structure interne des données binaires (raw). Par exemple, vous pouvez enregistrer les données binaires dans un champ BLOB de base de données, vers un champ de type `xs:base64Binary` dans un fichier XML ou l'envoyer dans un service Web*.

** les appels de service Web sont uniquement pris en charge dans MapForce Enterprise Edition.*

Vous pouvez également créé des mappages qui lisent les données binaires d'une source (telle que le champ BLOB dans une base de données, un champ de type `xs:base64Binary` dans un fichier XML, ou un service Web), puis écrire les fichiers binaires sur un disque. La liste ci-dessous illustre quelques scénarios possibles dans lesquels les fichiers binaires peuvent être utiles :

- Extrayez un contenu binaire encodé comme données base-64 depuis le fichier XML et enregistrez-le sur le disque (par exemple, comme fichier PDF)
- Traitez les fichiers d'image stocés sur votre disque et envoyez-les comme contenu binaire base-64 vers un service Web
- Extrayez le contenu BLOB depuis une table de base de données et enrgistrez-le sur le disque (un fichier d'image pour chaue ligne dans la table de base de données)
- Lisez les fichiers d'image du disque et enregistrez-le dans la table de base de données comme champs de données BLOB.

**Note :** les données de mappage vers ou des fichiers binaires requiert BUILT-IN[22] comme langage de transformation. Vous pouvez prévisualiser le mappage dans MapForce (et enregistrer les fichiers de sortie, le cas échéant) ou choisir de l'exécuter avec MapForce Server (licence uniquement) sur un

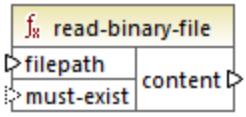
autre ordinateur ou une plateforme différente. Il n'est pas pris en charge pour générer un programme C#, C++ ou Java exécutable pour des mappages qui lisent ou écrivent des fichiers bianires.

*Lire depuis et écrire vers les fichiers binaires*
En tant que tel, il n'y a pas de genre de composant associé aux fichiers binaires dans MapForce, comme c'est le cas, par exemple, avec des fichiers XML, texte ou JSON. À la place, pour aider à accomplir ces objectifs comme ceux ci-dessus, les fonctions built-in suivantes sont disponibles :

- `read-binary-file` 666
- `write-binary-file` 668

## 6.7.13.1 read-binary-file

Retourne le contenu du fichier spécifié en tant qu'un BLOB (binary large object) de type `xs:base64Binary`. Veuillez noter que bien que le type de données est appelé "base64Binary", la représentation interne est juste un BLOB. Ce n'est que lorsque vous mappez le résultat de la fonction dans un nœud XML de type `xs:base64Binary` qu'il sera encodé en base64. Vous pourriez aussi mapper le résultat de la fonction dans `xs:hexBinary`, dans un blob de base de données ou dans un champ binaire dans une structure Protocol Buffers.




Pour lire un fichier binaire dans un mappage, fournir son chemin en tant qu'entrée dans l'argument **filepath**. Si le **filepath** est relatif, MapForce cherchera le fichier dans le même répertoire que le mappage. L'argument **must-exist** est optionnel ; si le fichier ne peut pas être ouvert et que ce paramètre est **vrai**, le mappage lance une erreur. Si le fichier ne peut pas être ouvert et que ce paramètre est **faux**, un binaire vide est retourné.

### Langages

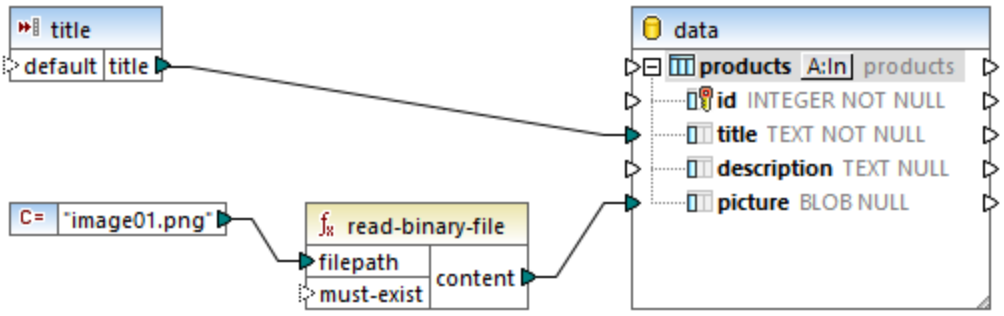
Built-in.

### Paramètres

| Nom | Type | Description |
|---|---|---|
| **filepath** | `xs:string` | Le chemin de fichier. |
| **must-exist** | `xs:boolean` | Paramètre optionnel. Si le fichier ne peut pas être ouvert et que ce paramètre est **true**, le mappage lance une erreur. Si le fichier ne peut pas être ouvert et que ce paramètre est **false**, un binaire vide est retourne.<br><br>La valeur par défaut est **true**. |

## Exemple

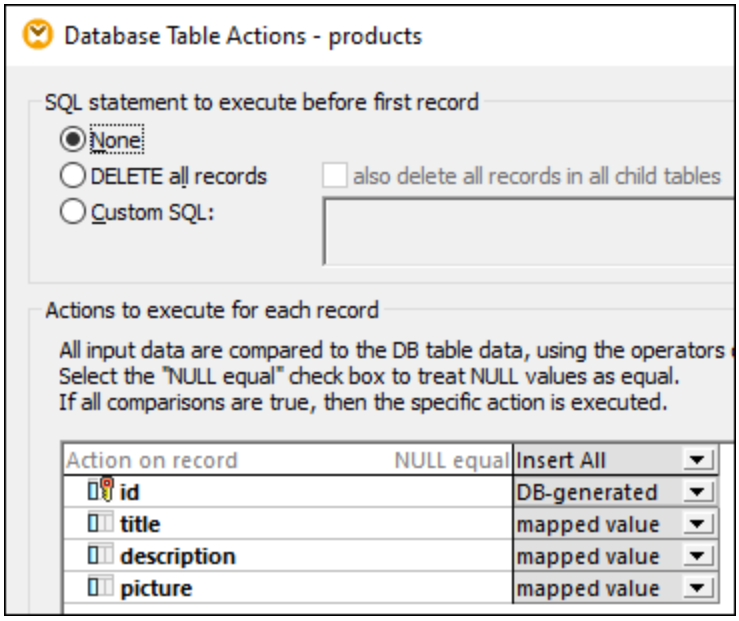
Le mappage illustré ci-dessous lit des données depuis un fichier d'image et l'écrit dans une table de base de données. La base de données de cible est SQLite. Veuillez noter que le type de données du champ de base de données **picture** est BLOB.



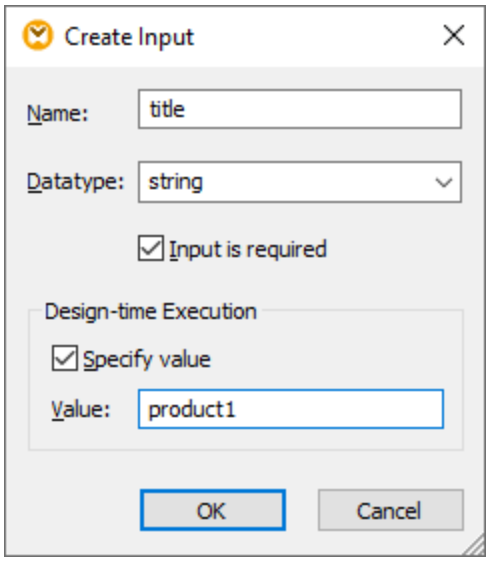

Pour extraire le contenu binaire du fichier, la fonction `read-binary-file` a été utilisée. Dans cet exemple, le premier argument, **filepath** est fourni par une constante Étant donné que le chemin est relatif, MapForce cherchera le fichier d'image dans le même répertoire que le mappage.

Dans la base de données de cible, les champs suivants sont configurés en tant que NOT NULL, le mappage doit donc les fournir :

- **id** - Dans cet exemple, le composant de base de données est configuré de manière à ce que **id** est généré par base de données au lieu d'être fourni par le mappage. Pour plus d'informations, voir Insérer des données dans une table 288.




- **title** - cette valeur est fournie par un simple composant d'entrée portant le même nom. Veuillez noter qu'une valeur d'exécution design-time est définie ("product1") afin de permettre le prévisualiser le mappage. Pour plus d'informations, voir Fournir des paramètres dans le mappage 398.

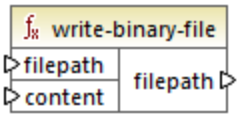



- **picture** - ce champ reçoit la sortie directe de la fonction **`read-binary-file`**.

Étant donné que le composant de cible est une base de données, la prévisualisation du mappage génère un script pseudo-SQL que vous pouvez consulter mais n'envoie pas de modifications à la base de données. Pour exécuter le script par rapport à la base de données, choisir la commande de menu **Sortie | Exécuter SQL-Script**.

## 6.7.13.2 write-binary-file

Cette fonction écrit du contenu binaire dans le chemin de fichier spécifié et retourne le chemin du fichier écrit. Si un fichier binaire est la seule sortie désirée, connecter le résultat de la fonction à un composant sortie simple[410]. Étant donné que cette fonction écrit un fichier à chaque fois qu'une sortie est utilisée dans le mappage, il est recommandé de connecter le résultat de la fonction directement dans un composant cible, sans utiliser un autre traitement entre temps.




Pour écrire des fichiers binaires, fournir leurs chemins en tant qu'entrée dans l'argument **filepath**. Si **filepath** est relatif, MapForce générera le fichier dans le même répertoire que le mappage. L'argument **content** doit être connecté au contenu binaire (par exemple, un champ BLOB dans une base de données).

Lorsque vous consultez le mappage dans MapForce, la fonction génère des fichiers temporaires par défaut, au lieu d'écrire des fichiers directement sur le disque. Pour enregistrer les fichiers temporaires sur le disque, cliquer tout d'abord sur l'onglet **Sortie**, puis cliquer sur la touche de la barre d'outils **Enregistrer sortie générée** ou **Enregistrer toutes les sorties générées**, selon les cas.

Pour configurer MapForce pour écrire une sortie directement dans des fichiers finaux au lieu des fichiers

temporaires, sélectionner la commande de menu **Outils | Options**, cliquer sur **Général**, puis sélectionner l'option **Écrire directement dans les fichiers de sortie finaux**. Veuillez noter que cette option écrase tous les fichiers existants avec le même nom.

La fonction retourne le nom du fichier final (pas temporaire), même si le fichier final n'est pas encore enregistré sur le disque (c'est le cas lorsque vous consultez le mappage et que l'option **Écrire directement dans les fichiers de sortie finaux** est désactivée).

> Veuillez noter qu'il n'est pas possible pour un mappage de relire son propre fichier de sortie.

## Langages

Built-In

## Paramètres

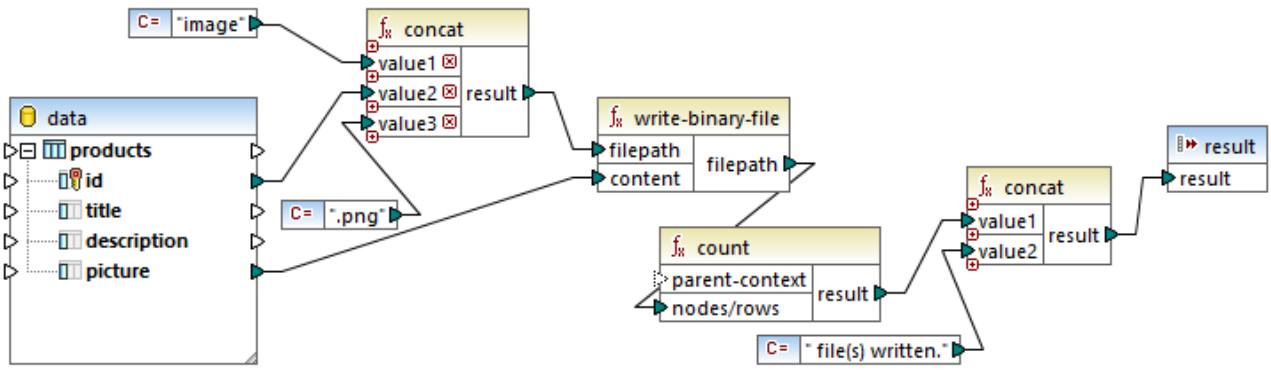
| Nom | Type | Description |
|---|---|---|
| **filepath** | `xs:string` | Le chemin de fichier d'entrée. |
| **contenu** | `xs:base64Binary` | Le contenu binaire de type `xs:base64Binary`. |

## Exemple

Le mappage illustré ci-dessous lit des valeurs BLOB d'une base de données SQLite et écrit des fichiers d'images sur le disque. La base de données a une table appelée **products**, qui présente les colonnes suivantes (champs) :

- **id** (*entier*, le seul nombre de série permanent de l'enregistrement)
- **title** (*texte*, le titre du produit)
- **description** (*texte*, la description du produit)
- **picture** (*blob*, l'image de produit)

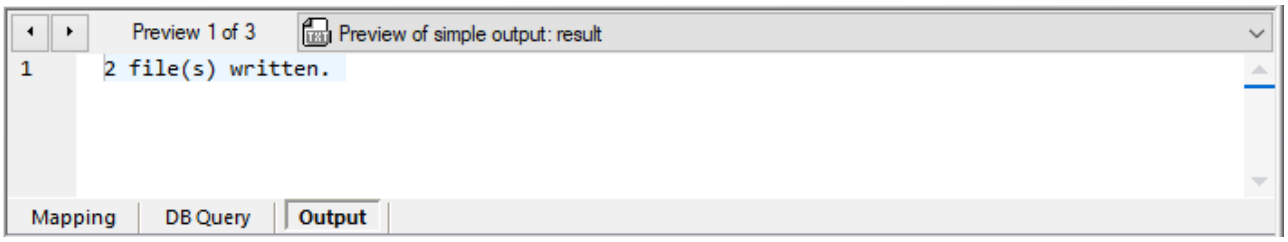
Pour le effectuer cet exemple, seuls les champs **id** et **picture** sont pertinents.



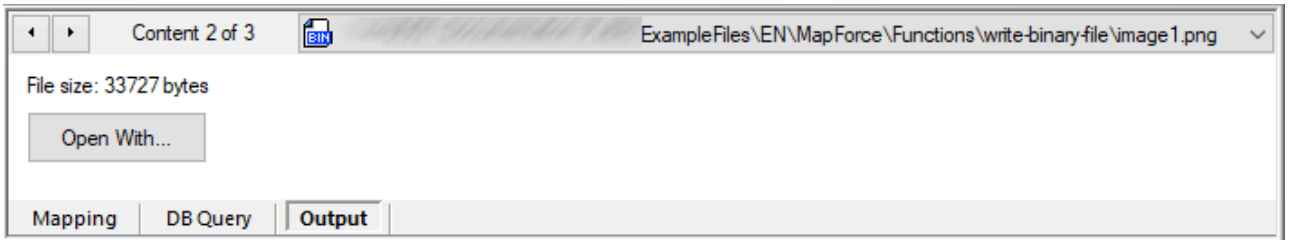
L'objectif du mappage est d'extraire toutes les images provenant de la table de **products** et de les écrire en tant que fichiers dans le disque. Comme illustré ci-dessus, la fonction `write-binary-file` est utilisée à cet escient. Le premier argument, **filepath**, reçoit le chemin de fichier pour chaque image. Le chemin doit être unique pour éviter de contourner des fichiers, il est donc généré en concaténant la base de données unique **id** de chaque enregistrement avec le mot "image" puis en ajoutant l'extension de fichier ".png" (on part du principe que toutes les images sont en format PNG). Veuillez noter que le chemin est relatif, les fichiers sont donc écrits dans le même répertoires que le mappage.

Le second argument, **content**, reçoit le contenu binaire stocké dans la base de données.

La fonction `count` retourne un décompte de tous les fichiers générés et combine ce nombre avec le string "file(s) written". Un rapport est produit indiquant combien de fichiers ont été générés par le mappage. Dans cet exemple, la base de données contient uniquement deux enregistrements de produit, la sortie de mappage réfléchira cela :




Comme évoqué précédemment, cette fonction génère des fichiers temporaires lorsque le mappage est exécuté dans l'exécution de prévisualisation. Pour consulter chaque fichier individuel , utiliser les touches flèchées pour naviguer dans l'enregistrement qui vous intéresse, cliquer sur la touche **Ouvrir avec**, et choisir un éditeur d'image.



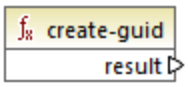

Pour enregistrer tous les fichiers, cliquer sur la touche de barre d'outils **Enregistrer la sortie générée** ou **Enregistrer toutes les sorties générées**, le cas échéant.

## 6.7.14 lang | generator functions

Les fonctions generator provenant de la bibliothèque **lang** sont des fonctions qui génèrent des valeurs (actuellement, `create-guid` est la seule fonction de ce type).

### 6.7.14.1 create-guid

Crée un identificateur global unique (GUID), en tant que string encodé hex. Cette fonction peut être utilisée pour générer des valeurs uniques, directement depuis le mappage, pour des champs de base de données ou d'autres types de composant. Voir aussi la fonction format-guid-string 690.




#### Langages

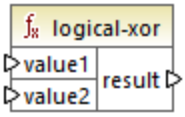
Built-in, C++, C#, Java.

## 6.7.15 lang | logical functions

Les fonctions logiques provenant de la bibliothèque **lang** contiennent des fonctions qui évaluent des types de valeur divers en utilisant une logique booléenne.

### 6.7.15.1 logical-xor

Retourne **true** si **value1** est différent de **value2**; **false** sinon.




#### Langages

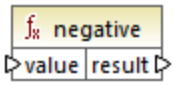
Built-in, C++, C#, Java.

#### Paramètres

| Nom | Type | echo Hello, World! |
|---|---|---|
| **value1** | `xs:boolean` | La première valeur d'entrée. |
| **value2** | `xs:boolean` | La deuxième valeur d'entrée. |

## 6.7.15.2 negative

Retourne **true** si la valeur d'entrée est négative (moins de zéro) ; **false** sinon.




### Langages

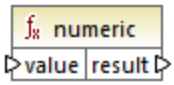
Built-in, C++, C#, Java.

### Paramètres

| Nom | Type | echo Hello, World! |
|---|---|---|
| **value** | `xs:decimal` | La valeur d'entrée. |

## 6.7.15.3 numeric

**mff-lib-lang-read-binary-file.txt**Retourne **true** si la valeur d'entrée est un nombre ou un string qui peut être parsdé en tant que nombre ; **false** sinon.




### Langages

Built-in, C++, C#, Java.

### Paramètres

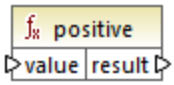
| Nom | Type | echo Hello, World! |
|---|---|---|
| **value** | `xs:decimal` | La valeur d'entrée. |

### Exemple

Si la valeur d'entrée est le string `"4.33"`, la fonction retourne **true**. Si la valeur d'entrée est le string `"4.33 USD"`, la fonction retourne **false**.

### 6.7.15.4 positive

Retourne **true** si la valeur d'entrée est positive (égale ou supérieure à zéro); **false** sinon.




#### Langages

Built-in, C++, C#, Java.

#### Paramètres

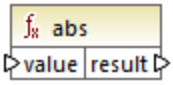
| Nom | Type | echo Hello, World! |
|---|---|---|
| **value** | `xs:decimal` | La valeur d'entrée. |

## 6.7.16 lang | math functions

Les fonctions math provenant de la bibliothèque **lang** peuvent être utilisée pour effectuer diverses opérations mathématiques dans le mappage.

### 6.7.16.1 abs

Retourne la valeur absolue de la valeur numérique fournie en tant qu'argument. Si l'argument n'est pas négatif, l'argument est retourné. Si l'argument est négatif, la négation de l'argument est retournée.




#### Langages

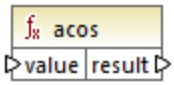
Built-in, C++, C#, Java.

#### Paramètres

| Nom | Type | echo Hello, World! |
|---|---|---|
| **value** | `xs:decimal` | La valeur d'entrée. |

## 6.7.16.2 acos

Retourne l'arc cosinus de **value**, dans la plage de `-pi/2` à `pi/2`.




### Langages

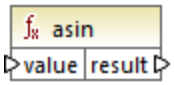
Built-in, C++, C#, Java.

### Paramètres

| Nom | Type | echo Hello, World! |
|---|---|---|
| **value** | `xs:double` | La valeur d'entrée. |

## 6.7.16.3 asin

Retourne l'arc sinus de **value**, dans la plage de `-pi/2` à `pi/2`.




### Langages

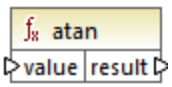
Built-in, C++, C#, Java.

### Paramètres

| Nom | Type | echo Hello, World! |
|---|---|---|
| **value** | `xs:double` | La valeur d'entrée. |

## 6.7.16.4 atan

Retourne l'arc tangente de **value**, dans la plage de `-pi/2` à `pi/2`.



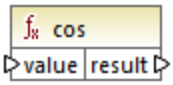
### Langages
Built-in, C++, C#, Java.

### Paramètres

| Nom | Type | echo Hello, World! |
|---|---|---|
| **value** | `xs:double` | La valeur d'entrée. |

## 6.7.16.5 cos

Retourne le cosinus trigonométrique de l'angle donné par **value**. L'unité de la valeur est radian.



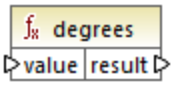

### Langages
Built-in, C++, C#, Java.

### Paramètres

| Nom | Type | echo Hello, World! |
|---|---|---|
| **value** | `xs:double` | La valeur d'entrée. |

## 6.7.16.6 degrees

Convertit un angle mesuré en radian à un angle équivalent approximatif mesuré en degrés.



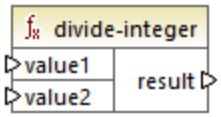

### Langages
Built-in, C++, C#, Java.

### Paramètres

| Nom | Type | echo Hello, World! |
|---|---|---|
| **value** | `xs:double` | La valeur d'entrée. |

## 6.7.16.7 divide-integer

Retourne le résultat entier de la division de **value1** par **value2**.




### Langages

Built-in, C++, C#, Java.

### Paramètres

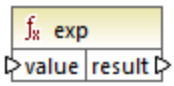
| Nom | Type | echo Hello, World! |
|---|---|---|
| **value1** | `xs:decimal` | La première valeur d'entrée. |
| **value2** | `xs:decimal` | La deuxième valeur d'entrée. |

### Exemple

Si la première valeur est **15** et que la seconde valeur est **2**, la fonction retourne **7**.

## 6.7.16.8 exp

Retourne le nombre d'Euler **e** élévé à la puissance de **value**.




### Langages

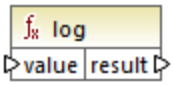
Built-in, C++, C#, Java.

### Paramètres

| Nom | Type | echo Hello, World! |
|---|---|---|
| **value** | `xs:double` | La valeur d'entrée. |

## 6.7.16.9 log

Retourne le logarithme naturel (base e) de **value**.




### Langages

Built-in, C++, C#, Java.

### Paramètres

| Nom | Type | echo Hello, World! |
|---|---|---|
| **value** | `xs:double` | La valeur d'entrée. |

## 6.7.16.10 log10

Retourne le logarithme décimal (base 10) de **value**.


### Langages

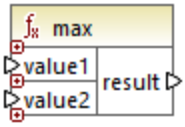
Built-in, C++, C#, Java.

### Paramètres

| Nom | Type | echo Hello, World! |
|---|---|---|
| **value** | `xs:double` | La valeur d'entrée. |

## 6.7.16.11 max

Retourne la valeur numérique de la plus grande valeur fournie en tant qu'argument. 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, voir aussi Ajouter ou supprimer des arguments de fonction 469.




### Langages

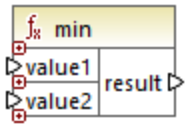
Built-in, C++, C#, Java.

### Paramètres

| Nom | Type | echo Hello, World! |
|---|---|---|
| **value1** | `xs:decimal` | La première valeur d'entrée. |
| **value2** | `xs:decimal` | La deuxième valeur d'entrée. |
| **valueN** | `xs:decimal` | La *n*ème valeur d'entrée. |

## 6.7.16.12 min

Retourne la valeur numérique de la plus petite valeur fournie en tant qu'argument. 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, voir aussi Ajouter ou supprimer des arguments de fonction 469.




### Langages

Built-in, C++, C#, Java.

### Paramètres

| Nom | Type | echo Hello, World! |
|---|---|---|
| **value1** | `xs:decimal` | La première valeur d'entrée. |
| **value2** | `xs:decimal` | La deuxième valeur d'entrée. |
| **valueN** | `xs:decimal` | La *n*ème valeur d'entrée. |

## 6.7.16.13 pi

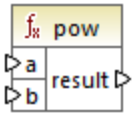
Retourne la valeur de la constante mathématique *pi*.


### Langages

Built-in, C++, C#, Java.

## 6.7.16.14 pow

Retourne la valeur de **a** élevé à la puissance de **b**.




### Langages

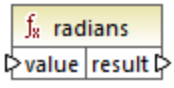
Built-in, C++, C#, Java.

### Paramètres

| Nom | Type | echo Hello, World! |
|---|---|---|
| **a** | `xs:double` | Fournit valeur **a** (la base). |
| **b** | `xs:double` | Fournit valeur **b** (la puissance). |

## 6.7.16.15 radians

Convertit un angle mesuré en degrés à un angle équivalent approximatif mesuré en radians.




### Langages

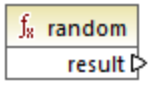
Built-in, C++, C#, Java.

### Paramètres

| Nom | Type | echo Hello, World! |
|---|---|---|
| **value** | `xs:double` | La valeur d'entrée. |

## 6.7.16.16 random

Retourne une valeur avec un sign positif, supérieur ou égal à 0.0 et inférieur à 1.0. Les valeurs retournées sont choisies pseudo-aléatoirement avec une distribution uniforme (environ) de cette plage.

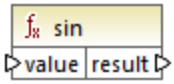



### Langages

Built-in, C++, C#, Java.

## 6.7.16.17 sin

Retourne le sinus de **value**.




### Langages

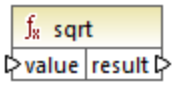
Built-in, C++, C#, Java.

### Paramètres

| Nom | Type | echo Hello, World! |
|---|---|---|
| **value** | `xs:double` | La valeur d'entrée. |

## 6.7.16.18 sqrt

Retourne la racine carrée positive arrondie correctement de **value**.




### Langages

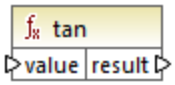
Built-in, C++, C#, Java.

### Paramètres

| Nom | Type | echo Hello, World! |
|---|---|---|
| **value** | `xs:double` | La valeur d'entrée. |

## 6.7.16.19 tan

Retourne la tangente de **value**.




### Langages

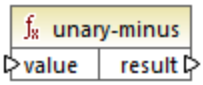
Built-in, C++, C#, Java.

### Paramètres

| Nom | Type | echo Hello, World! |
|---|---|---|
| **value** | `xs:double` | La valeur d'entrée. |

## 6.7.16.20 unary-minus

Retourne la négation de la valeur d'entrée signée.



### Langages

Built-in, C++, C#, Java.

### Paramètres

| Nom | Type | echo Hello, World! |
|---|---|---|
| **value** | `xs:decimal` | La valeur d'entrée. |

### Exemple

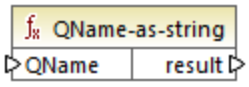
Si la valeur d'entrée est **3**, la fonction retourne **-3**. Si la valeur d'entrée est **-3**, la fonction retourne **3**.

## 6.7.17 lang | QName functions

Les fonctions QName de la bibliothèque **lang** convertissent les valeurs Qualified Name (QName) en strings, et vice-versa. Contrairement aux fonctions provenant de la bibliothèque **core**, ces fonctions sont disponibles uniquement dans les langages Built-in, Java, C# ou C++.

### 6.7.17.1 QName-as-string

Retourne la représentation de string de la valeur Qname fournie en tant qu'argument.




### Langages

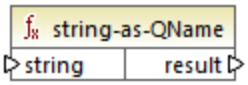
Built-in, C++, C#, Java.

### Paramètres

| Nom | Description |
|---|---|
| **QName** | La valeur d'entrée de `xs:QName`. |

### 6.7.17.2 string-as-QName

Reconvertit la représentation de string d'un QName en un QName.




#### Langages

Built-in, C++, C#, Java.

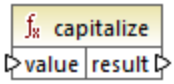
| Nom | Description |
|---|---|
| **Paramètres** | La valeur de `string` d'entrée. |

## 6.7.18 lang | string functions

Les fonctions string provenant de la bibliothèque `lang` vous permettent de traiter des strings (par ex., trim, pad, replace, convertir des strings en casse majuscule ou minuscule, etc.).

### 6.7.18.1 capitalize

Retourne le string d'entrée **value,** où la première lettre de chaque mot est une majuscule.




#### Langages

Built-in, C++, C#, Java.

#### Paramètres

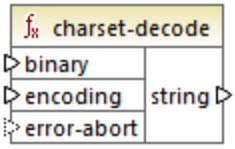
| Nom | Type | echo Hello, World! |
|---|---|---|
| **value** | `xs:string` | La valeur d'entrée. |

#### Exemple

Si la valeur d'entrée est `the quick  brown fox`, la fonction retourne `The Quick Brown Fox`.

## 6.7.18.2 charset-decode

La fonction **charset-decode** prend en tant qu'entrée des données binaires encodées en tant que texte Base64. Elle décode des données conformément à l'ensemble de caractères spécifié (par exemple, "utf-8") et retourne la valeur de string résultante. Si vous souhaitez encoder des données binaires en tant que texte Base64, utilsier la fonction charset-encode 686.




### Langages

Built-in.

### Paramètres

| Nom | Type | echo Hello, World! |
|---|---|---|
| binary-data | xs:base64Binary | Les données binaires en tant que texte Base64. |
| encoding | xs:string | L'ensemble de caractère utilisé pour l'encodage (par exemple, "utf-8"). |
| error-abort | xs:boolean | Argument optionnel qui spécifie comment le traitement doit continuer en cas d'erreurs. Valeurs valides :<br>• **true** - Terminer le traitement avec une exception sur le caractère invalide<br>• **false** - Continuer le traitement et remplacer des caractères valides avec le caractère de remplacement .<br>La valeur par défaut est **true**. |

### Exemple

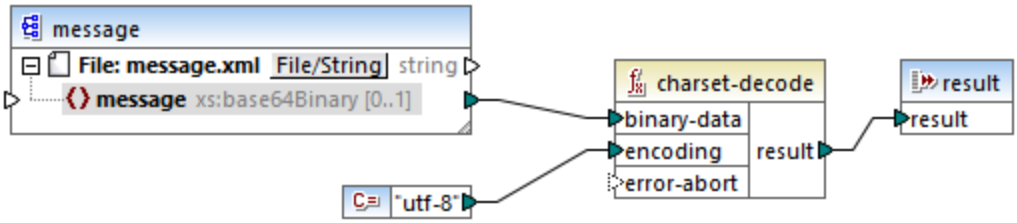
Supposons que vous souhaitez décoder des données binaires provenant du fichier XML source suivant. Veuillez noter que l'élément de **message** contient des données binaires encodées en tant que texte Base64.

```
<?xml version="1.0" encoding="UTF-8"?>
<message xsi:noNamespaceSchemaLocation="message.xsd"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">TG9yZW0gaXBzdW0=</message>
```

Le type de données de l'élément **message** est `xs:base64Binary`, comme illustré par le schéma :

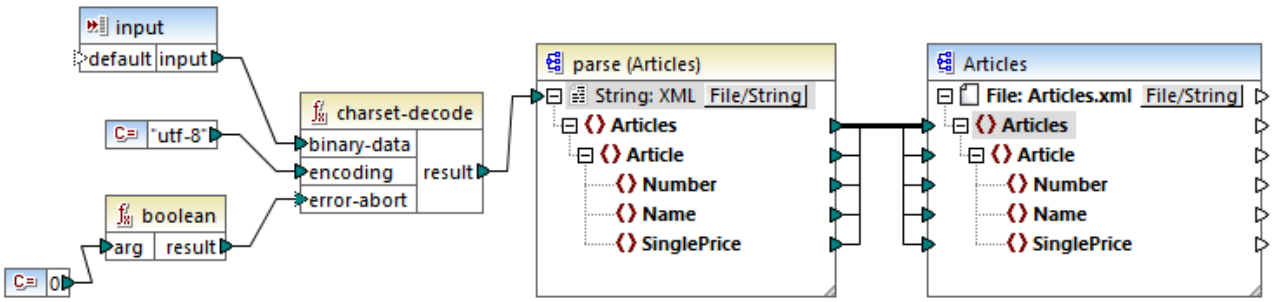
```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
   <xs:element name="message" type="xs:base64Binary"/>
</xs:schema>
```

Un mappage qui décode le message ci-dessus comme suit :




Le mappage dans cet exemple produit le texte "Lorem ipsum".

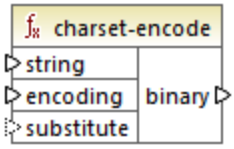
Un mappage peut aussi traiter du texte ou des fichiers XML encodés en tant que données Base64, avec l'aide d'un composant de sérialisation MapForce. Par exemple, le mappage illustré ci-dessous a un paramètre **input** qui nécessite des données de texte Base64. En partant du principe que les données Base64 ont été créées depuis un fichier XML comme indiqué dans l'exemple charset-encode [686], vous pouvez recréer le fichier XML original comme indiqué dans le mappage ci-dessous :




Dans ce mappage, l'argument **error-abort** obtient une valeur **false**, qui a été produite avec l'aide de la fonction intégrée **boolean**. Cela garantit que le traitement se poursuivra même si des caractères invalides surviennent. Le résultat de string de la fonction est ensuite transmis à un composant de parsage XML qui le convertit en un fichier XML. Veuillez noter que, pour que le parsage XML soit possible, vous devez avoir le fichier de schéma XSD. Pour plus d'informations, voir Parser et sérialiser des strings [794].

## 6.7.18.3 charset-encode

La fonction **charset-encode** prend en tant que string d'entrée des données et l'encode en tant que texte Base64. Les données sont encodées dans l'ensemble de caractère spécifique (par exemple, "utf-8") et retournées en tant que type `xs:base64Binary`. Si vous souhaitez décoder des données binaires encodées précédemment en tant que texte Base64, utiliser la fonction charset-decode 684.




### Langages

Built-in.

### Paramètres

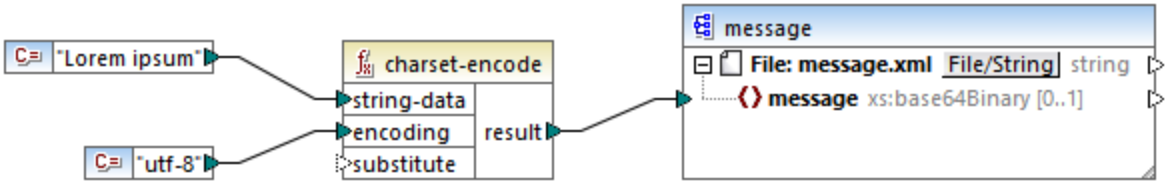
| Nom | Type | echo Hello, World! |
|---|---|---|
| string-data | **xs:string** | Les données de string à encoder. |
| encoding | **xs:string** | L'ensemble de caractère utilisé pour l'encodage (par exemple, "utf-8"). |
| substitute | **xs:string** | Argument optionnel qui spécifie un caractère de remplacement en cas d'émergence de caractères invalides. Cet argument est applicable si vous utilisez un encodage non-Unicode. Pour des encodages Unicode, le caractère de remplacement est . |

### Exemple

Imaginons que vous souhaitez encoder le texte "Lorem ipsum" en tant que données Base64, en utilisant le set de caractère UTF-8 et que vous souhaitez l'écrire dans un fichier XML cible. Celui-ci a un élément **message** de `xs:base64Binary` comme illustré par le schéma :

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
   <xs:element name="message" type="xs:base64Binary"/>
</xs:schema>
```

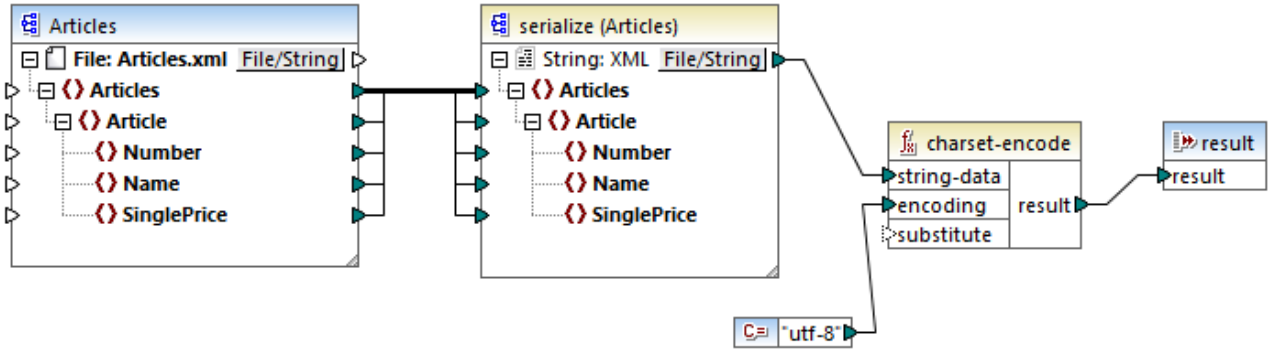
Un mappage qui effectue l'encodage Base64 ressemble à l'exemple suivant :




Ce mappage produit un résultat XML comme celui indiqué ci-dessous (les références de schéma et la déclaration XML ont été sautées) :

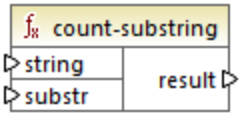
```
<message>TG9yZW0gaXBzdW0=</message>
```

Vous pouvez aussi encoder du texte ou des fichiers XML en tant que Base64, avec l'aide d'un composant de sérialisation MapForce. Par exemple, le mappage illustré ci-dessous sérialise un fichier XML de source dans un string. Le string résultant est ensuite fourni en tant qu'argument dans la fonction **charset-encode**. Enfin, le résultat de fonction est retourné en tant que sortie de mappage avec l'aide d'un composant de sortie simple, voir Retourner les valeurs String depuis un mappage 409. Pour plus d'informations concernant la sérialisation, voir Parser et sérialiser des strings 794.




## 6.7.18.4 count-substring

Retourne une valeur d'entier exprimant le nombre de fois que **substr** se produit dans **string**.




### Langages

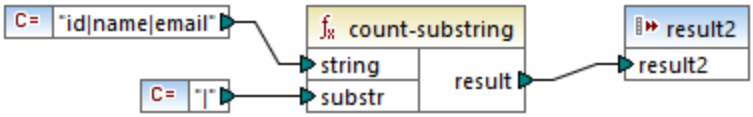
Built-in, C++, C#, Java.

### Paramètres

| Nom | Type | echo Hello, World! |
|---|---|---|
| **Paramètres** | `xs:string` | Type |
| **substr** | `xs:string` | Le sous-string à tester. |

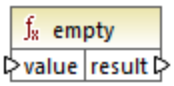
### Exemple

Le mappage suivant retourne **2**. Il s'agit du nombre de fois que la barre verticale se produit dans le cadre du string d'entrée `id|name|email`.




## 6.7.18.5 empty

Retourne **true** si la valeur de string d'entrée est vide ; **false** sinon.




### Langages

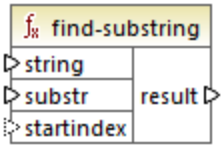
Built-in, C++, C#, Java.

### Paramètres

| Nom | Type | echo Hello, World! |
|---|---|---|
| **value** | `xs:string` | La valeur d'entrée. |

## 6.7.18.6 find-substring

Retourne la position de la première occurrence de **substr** dans **string**. Par défaut, la fonction lance la recherche depuis le premier caractère, qui a la position (index) 1, mais vous pouvez spécifier en option un index de démarrage spécifique. Si **substr** ne peut pas être trouvé, la fonction retourne **0**.




## Langages

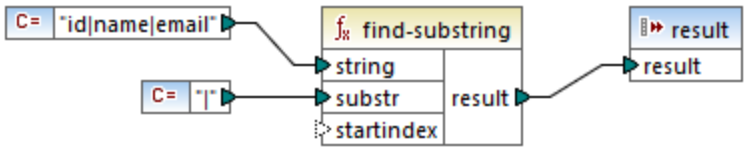
Built-in, C++, C#, Java.

## Paramètres

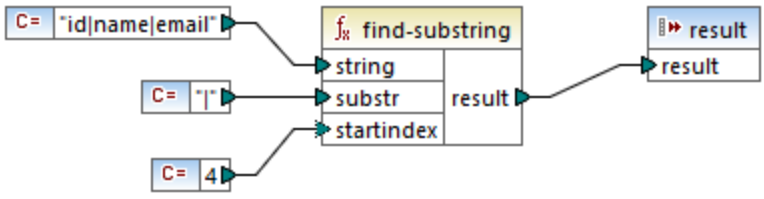
| Nom | Type | echo Hello, World! |
|---|---|---|
| **Paramètres** | `xs:string` | Type |
| **substr** | `xs:string` | Le sous-string à chercher. |
| **startindex** | `xs:int` | Optionnel Spécifie la position de début (index) de la recherche. Si ce paramètre n'est pas spécifié, la recherche commence à la position 1. |

## Exemple

Le mappage suivant sort **3**, qui est la position de la première occurrence du caractère vertical dans le string d'entrée `id|name|email`.

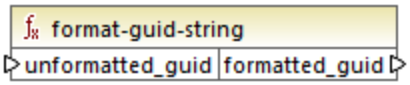



Si vous spécifiez **4** en tant qu'index de démarrage, la fonction commencera la recherche à partir du quatrième caractère. Par conséquent, le mappage ci-dessous produira **8**, qui est la première occurrence du caractère vertical après la recherche à partir du quatrième caractère.



## 6.7.18.7 format-guid-string

Retourne un string d'identifiant unique global (GUID) correctement formaté, typique pour l'utilisation dans des champs de base de données. Voir aussi la fonction create-guid 671.




### Langages

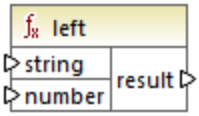
Built-in, C++, C#, Java.

### Paramètres

| Nom | Type | echo Hello, World! |
|---|---|---|
| **unformatted_guid** | `xs:string` | Le string d'entrée encodé en HEX à formater. |

## 6.7.18.8 left

Retourne un string contenant les premiers caractères **number** du string d'entrée.




### Langages

Built-in, C++, C#, Java.

### Paramètres

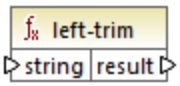
| Nom | Type | echo Hello, World! |
|---|---|---|
| **Paramètres** | `xs:string` | Type |
| **number** | `xs:int` | Spécifie combien de caractères doivent être retournés, en commençant avec le début du string. |

### Exemple

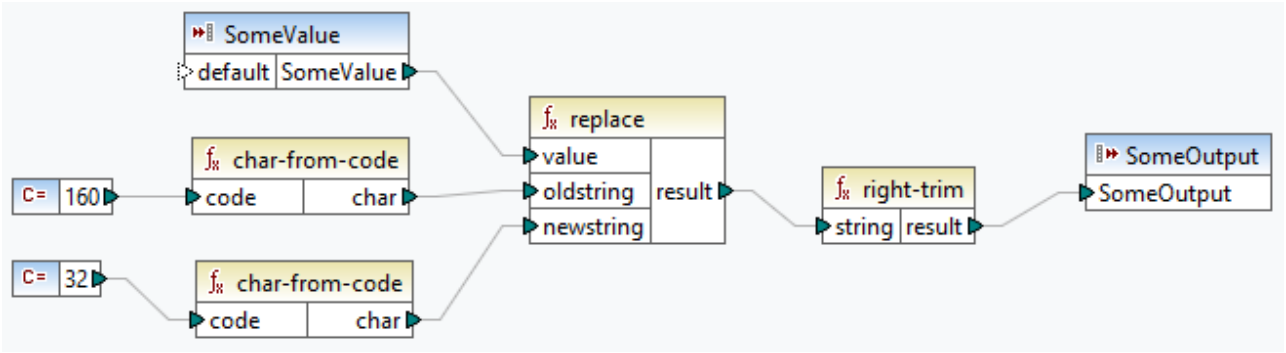
Si le string d'entrée est `This is a sentence` et que le nombre est **4**, la fonction retourne `This`.

## 6.7.18.9 left-trim

La fonction **left-trim** (*voir la capture d'écran ci-dessous*) supprime l'espace blanc à gauche d'un string. 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

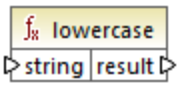
Built-in, C++, C#, Java.

### Paramètres

| Nom | Type | Description |
|---|---|---|
| **string** | `xs:string` | Le string d'entrée. |

## 6.7.18.10 lowercase

Convertit le **string** d'entrée en casse minuscule. Pour des caractère Unicode, les caractères en casse minuscule correspondants (définis par le groupe Unicode) sont utilisés.




### Langages

Built-in, C++, C#, Java.

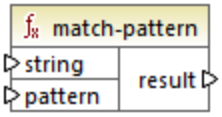
### Paramètres

| Nom | Type | echo Hello, World! |
|---|---|---|
| **Paramètres** | `xs:string` | Type |

## 6.7.18.11 match-pattern

Retourne booléenne **true** si le string d'entrée correspond à l'expression régulière définie par **pattern** ; **false** sinon. Voir aussi Expressions régulières 543.

**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

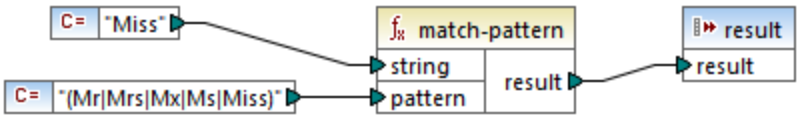
Built-in, C++, C#, Java.

### Paramètres

| Nom | Type | echo Hello, World! |
|---|---|---|
| **Paramètres** | `xs:string` | Type |
| **pattern** | `xs:string` | L'expression régulière à faire correspondre. |

### Exemple

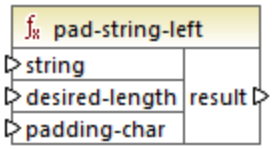
Le mappage suivant valide plusieurs titres de personnes. En particulier, le mappage produira **true** pour tous les titres suivants : Mr, Mrs, Mx, Ms, Miss.




SI le string d'entrée est différent de tous les titres listés ci-dessus, le mappage produira **false**.

## 6.7.18.12 pad-string-left

Retourne le string qui est complété à sa gauche par un seul caractère spécifique, jusqu'à une longueur requise. La longueur de string désirée et le caractère de remplissage sont fournis en tant qu'arguments.




### Langages

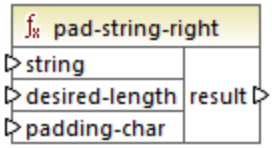
Built-in, C++, C#, Java.

### Paramètres

| Nom | Type | echo Hello, World! |
|---|---|---|
| **Paramètres** | `xs:string` | Spécifie le string d'entrée. |
| **desired-length** | `xs:int` | Définit la longueur désirée du string après le remplissage. |
| **padding-char** | `xs:string` | Définit le caractère à utiliser en tant que caractère de remplissage. |

## 6.7.18.13 pad-string-right

Retourne le string qui est complété à sa droite par un seul caractère spécifique, jusqu'à une longueur requise. La longueur de string désirée et le caractère de remplissage sont fournis en tant qu'arguments.



### Langages

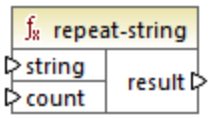
Built-in, C++, C#, Java.

### Paramètres

| Nom | Type | echo Hello, World! |
|---|---|---|
| **Paramètres** | `xs:string` | Spécifie le string d'entrée. |
| **desired-length** | `xs:int` | Définit la longueur désirée du string après le remplissage. |
| **padding-char** | `xs:string` | Définit le caractère à utiliser en tant que caractère de remplissage. |

## 6.7.18.14 repeat-string

Répète le string fourni en tant qu'argument *n* fois. L'argumet **count** spécifie le nombre de fois qu'il faut répéter le string.




### Langages

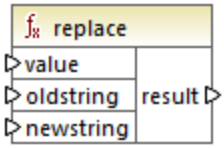
Built-in, C++, C#, Java.

### Paramètres

| Nom | Type | echo Hello, World! |
|---|---|---|
| **Paramètres** | `xs:string` | Type |
| **count** | `xs:int` | Le nombre de fois qu'il faut répéter le string. |

## 6.7.18.15 replace

Le résultat est un nouveau string où chaque instance de **oldstring**, dans le string d'entrée **value**, est remplacé par **newstring**.




### Langages

Built-in, C++, C#, Java.

### Paramètres

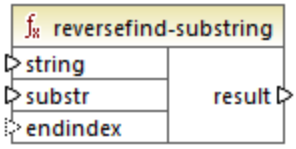
| Nom | Type | echo Hello, World! |
|---|---|---|
| **value** | `xs:string` | La valeur d'entrée. |
| **oldstring** | `xs:string` | L'ancien string à remplacer. |
| **newstring** | `xs:string` | Le nouveau string pour utiliser en tant que remplacement. |

### Exemple

Voir Remplacer les caractères spéciaux 626.

## 6.7.18.16 reversefind-substring

Retourne la position de la dernière occurrence de **substr** dans **string**. Par défaut, la fonction lance la recherche depuis le premier caractère, dont la position est (index) 1, et termine la recherche au niveau du dernier caractère, mais vous pouvez spécifier en option un index de fin. Si **substr** ne peut pas être trouvé, la fonction retourne **0**.




### Langages

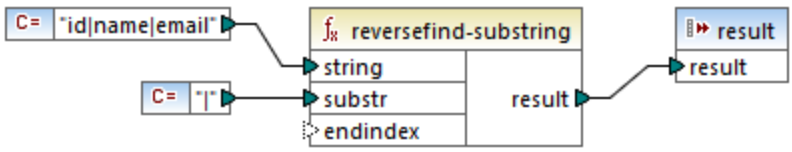
Built-in, C++, C#, Java.

### Paramètres

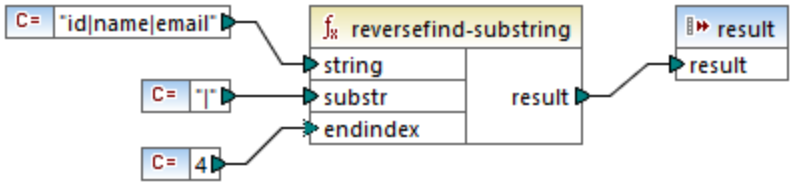
| Nom | Type | echo Hello, World! |
|---|---|---|
| **Paramètres** | `xs:string` | Type |
| **substr** | `xs:string` | Le sous-string à chercher. |
| **endindex** | `xs:int` | Optionnel Spécifie la position de fin (index) de la recherche. Si ce paramètre n'est pas spécifié, la recherche se termine après le dernier caractère dans **string**. |

### Exemple

Le mappage suivant sort **8**, qui est la position de la dernière occurrence du caractère vertical dans le string d'entrée `id|name|email`.

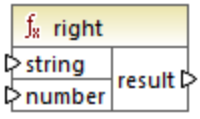



Si vous spécifiez **4** en tant qu'index de fin, la fonction recherchera jusqu'au quatrième caractère. Par conséquent, le mappage ci-dessous produira **3**.




## 6.7.18.17 right

Retourne un string contenant le dernier caractÈre **number** du string d'entrée.




### Langages

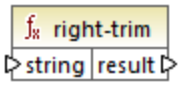
Built-in, C++, C#, Java.

### Exemple

Si le string d'entrée est **`The brown red fox`** et le nombre est **3**, la fonction retourne **`fox`**.

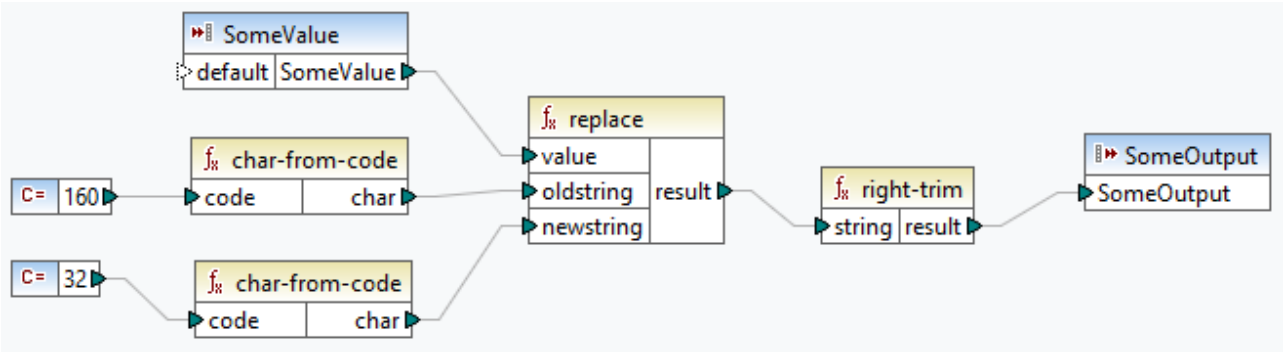
## 6.7.18.18 right-trim

La fonction **`right-trim`** (*voir la capture d'écran ci-dessous*) supprime l'espace blanc à droite d'un string. 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

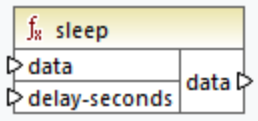
Built-in, C++, C#, Java.

### Paramètres

| Nom | Type | Description |
|---|---|---|
| **string** | `xs:string` | Le string d'entrée. |

## 6.7.18.19 sleep

La fonction **sleep** (*capture d'écran ci-dessous*) retarde la transmission de données pour N secondes. Passer une séquence à travers la fonction retardera chaque item de la séquence pour un moment spécifié. La fonction **sleep** est compatible avec les langages de transformation suivants : Java, C#, C++, et Built-In. La génération de code est prise en charge dans Java, C# et C++. Pour plus d'information sur la génération de code, voir le Générateur de Code[935].




### Paramètres

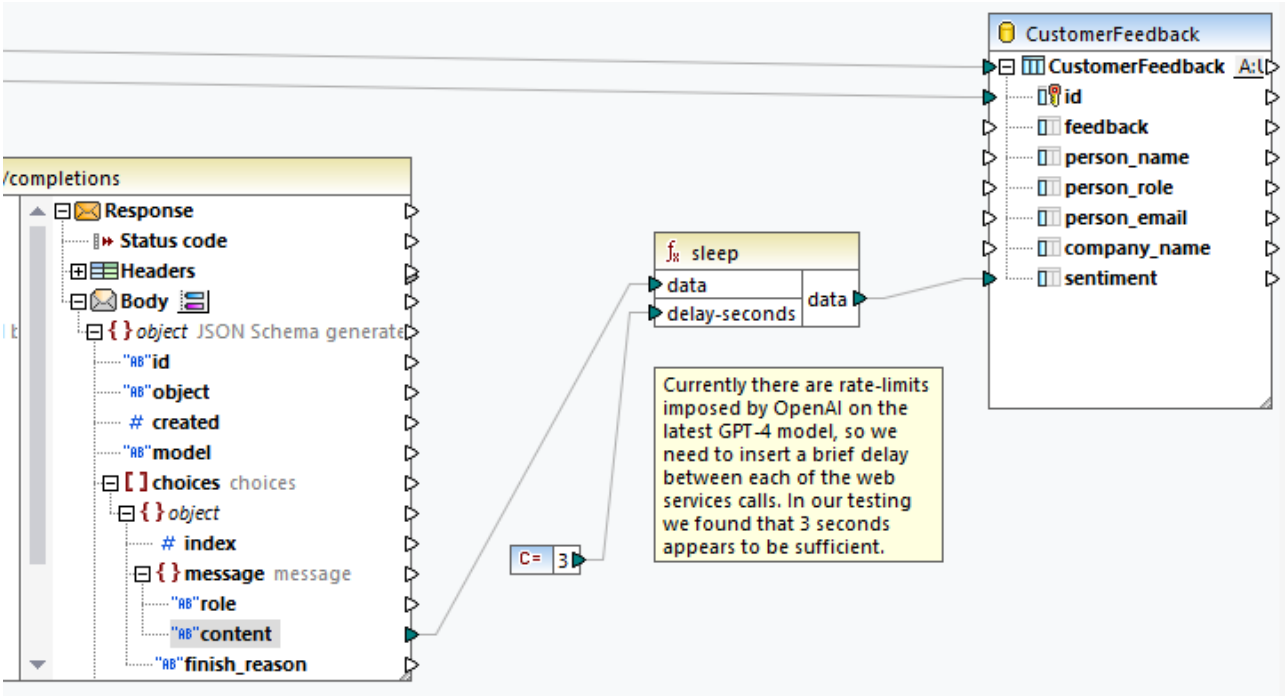
| Nom | Type | Description |
|---|---|---|
| données | tout nœud ou type atomique | Le paramètre d'entrée data accepte toute valeur (par ex., string). |
| delay-seconds | xs:double | Le paramètre d'entrée delay-seconds retarde la transmission de données pour N secondes. Secondes fractionnelles sont aussi acceptables. |
| données | tout nœud ou type atomique | Le paramètre de sortie data obtient les données de l'entrée et passe ces données au nœud cible. |

### Exemple

Pour un scénario use-case possible dans lequel la fonction **sleep** est utilisée, voir le mappage suivant : **MapForceExamples\SentimentAnalysis.mfd**. un extrait de ce mappage est illustré ci-dessous. Pour pouvoir tester le mappage, vous aurez besoin de vos identifiants de connexion de l'organisation.

Puisque OpenAI impose des limites de taux de réponses sur les requêtes API que vous pouvez faire, vous pouvez rencontrer une erreur comme Trop de requêtes. La fonction **sleep** vous permet de passer outre les limites de taux de réponses en configurant un délai.

Dans la structure de réponse de l'appel de service web ci-dessous, le nœud content obtient des données comme résultat de la requête envoyée à l'API OpenAI. Avant chaque appel de service web, il existe un délai pour 3 secondes, puis la valeur du nœud content est mappée vers la colonne sentiment de la base de données CustomerFeedback.

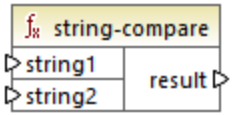



Pour plus d'information sur cet exemple et la fonctionnalité IA dans MapForce, voir les articles suivants :

- *Intégration de données avec l'IA*
- *Analyse des sentiments de la requête de prise en charge basée sur l'IA utilisant MapForce et GPT-4*
- *Classification d'image de la base de données basée sur l'IA avec Altova MapForce*

## 6.7.18.20 string-compare

La fonction **`string-compare`** (*voir la capture d'écran ci-dessous*) retourne le résultat d'un caractère en comparant les caractères de deux strings d'entrée : `string1` et `string2`. La comparaison est basée sur des codes ASCII. Les deux `string1` et `string2` sont de type `xs:string`. Le fonction est sensible à la casse. Si les strings sont égaux, le résultat est 0. Si `string1` est inférieur à `string2`, le résultat sera -1. Si `string1` est supérieur à `string2`, le résultat sera 1.




**Exemple :**

```
string1 : hi
string2 : Hit
```

La fonction **`string-compare`** compare les strings caractère par caractère. La comparaison se termine une fois que la fonction a détecté que le premier caractère de `string1` et le premier caractère de `string2` sont

différents. Le résultat est basé sur la comparaison du premier caractère de chaque string. Puisque `h` est représenté comme numéro de code ASCII supérieur (104 selon le système décimal) à `H` (72 dans le système décimal), `string1` est supérieur au `string2`, et le résultat de la comparaison de string est 1. Si le premier caractère du `string1` et le premier caractère du `string2` étaient les mêmes, la fonction poursuivrait l'analyse du second caractère et ainsi de suite.

Pour une comparaison de string simple avec un résultat booléen, voir `core | logical functions | equal` 579.

### Langages

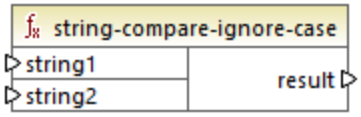
Built-in, C++, C#, Java.

### Paramètres

| Nom | Type | Description |
|---|---|---|
| `string1` | `xs:string` | Le premier string d'entrée. |
| `string2` | `xs:string` | Le second string d'entrée. |

## 6.7.18.21 string-compare-ignore-case

La fonction **`string-compare-ignore-case`** (*voir la capture d'écran ci-dessous*) retourne le résultat d'un caractère en comparant les caractères de deux strings d'entrée : `string1` et `string2`. Les deux `string1` et `string2` sont de type `xs:string`. La focntion ignore la casse. La comparaison est basée sur des codes ASCII. Si les strings sont égaux, le résultat est 0. Si `string1` est inférieur à `string2`, le résultat sera -1. Si `string1` est supérieur à `string2`, le résultat sera 1.




**Exemple :**

```
string1 : hi
string2 : Hit
```

La fonction **`string-compare-ignore-case`** compare les strings caractère par caractère. Même si `h` est représenté en tant que numéro de code ASCII supérieur à `H`, ces deux caractères sont traités en égal dans cette fonction. Le deuxième caractère dans les deux strings est le même. Toutefois, le `string2` a un troisième caractère, alors que le `string1` n'en a pas. Le troisième caractère dans le `string1` a une valeur vide. La valeur `t` dans le `string2` est supérieure à la valeur vide dans le `string1`. C'est la raison pour laquelle le `string1` est inférieur au `string2`, et le résultat est égale à -1.

### Langages

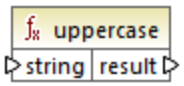
Built-in, C++, C#, Java.

### Paramètres

| Nom | Type | Description |
|---|---|---|
| `string1` | `xs:string` | Le premier string d'entrée. |
| `string2` | `xs:string` | Le second string d'entrée. |

## 6.7.18.22 uppercase

Convertit le **string** d'entrée en majuscule. En ce qui concerne les caractères Unicode, les caractères de majuscule correspondants sont utilisés (tels que définis par le groupe Unicode).




### Langages

Built-in, C++, C#, Java.

### Paramètres

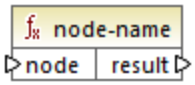
| Nom | Type | echo Hello, World! |
|---|---|---|
| **Paramètres** | `xs:string` | Type |

# 6.7.19 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.

## 6.7.19.1 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

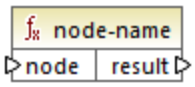
XQuery, XSLT 2.0, XSLT 3.0.

### Paramètres

| Nom | Type | Description |
|---|---|---|
| **node** | `mf:node` | Le nœud d'entrée. |

## 6.7.19.2 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`** 596.




### Langages

XQuery, XSLT 2.0, XSLT 3.0.

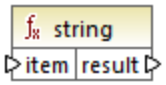
### Paramètres

| Nom | Type | Description |
|---|---|---|
| **node** | `mf:node` | Le nœud d'entrée. |

## 6.7.19.3 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. |

## 6.7.20 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.

### 6.7.20.1 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

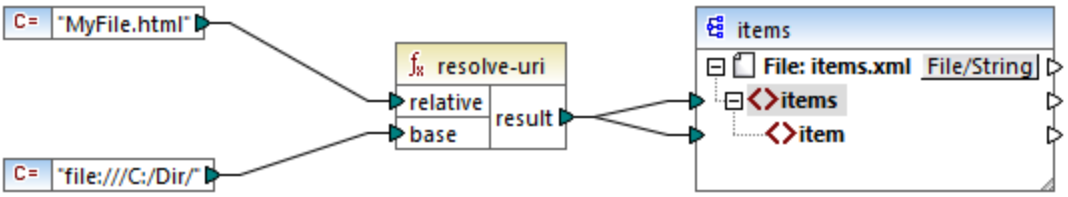
XQuery, XSLT 2.0, XSLT 3.0.

### 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`.

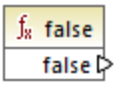



## 6.7.21 xpath2 | boolean functions

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.

### 6.7.21.1 false

Retourne la valeur booléenne **false**.

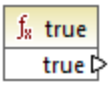



#### Langages

XQuery, XSLT 2.0, XSLT 3.0.

### 6.7.21.2 true

Retourne la valeur booléenne **true**.




#### Langages

XQuery, XSLT 2.0, XSLT 3.0.

## 6.7.22 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.

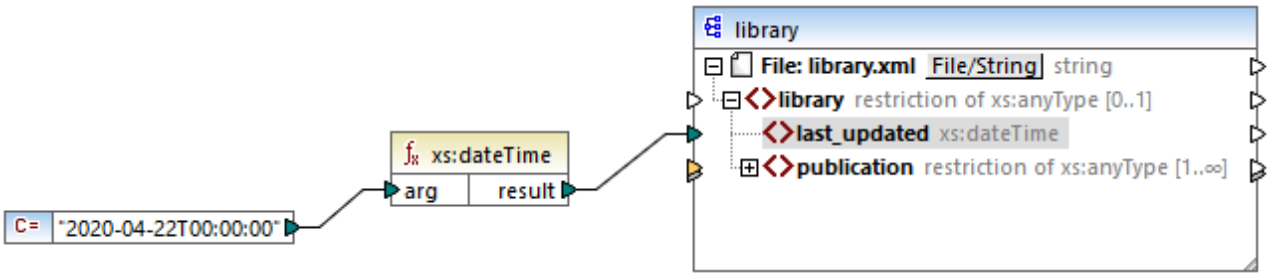
| | | |
|---|---|---|
| **xs:ENTITY** | **xs:double** | **xs:nonPositiveInteger** |
| **xs:ID** | **xs:duration** | **xs:normalizedString** |
| **xs:IDREF** | **xs:float** | **xs:positiveInteger** |
| **xs:NCName** | **xs:gDay** | **xs:short** |
| **xs:NMTOKEN** | **xs:gMonth** | **xs:string** |
| **xs:Name** | **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`.

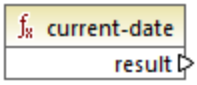
Pour consulter le type de données attendu d'un item de mappage (y compris le type de données des arguments de fonction), déplacer le curseur de la souris sur le connecteur d'entrée ou de sortie respectif.

## 6.7.23 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.

### 6.7.23.1 current-date

Retourne la date actuelle (`xs:date`) depuis l'horloge du système.

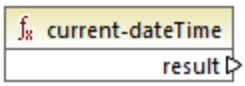



#### Langages

XQuery, XSLT 2.0, XSLT 3.0.

### 6.7.23.2 current-dateTime

Retourne la date et l'heure actuels (`xs:dateTime`) depuis l'horloge du système.

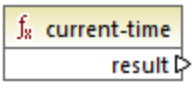



#### Langages

XQuery, XSLT 2.0, XSLT 3.0.

## 6.7.23.3 current-time

Retourne l'heure actuelle (`xs:time`) depuis l'horloge du système.




### Langages

XQuery, XSLT 2.0, XSLT 3.0.

## 6.7.23.4 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.

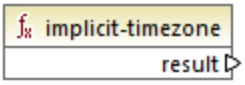
Des comparaisons, y compris pour les fonctions **`max-string`** et **`min-string`** sont basées sur cette collation.


### Langages

XQuery, XSLT 2.0, XSLT 3.0.

## 6.7.23.5 implicit-timezone

Retourne la valeur de la propriété "implicit timezone" depuis le contexte d'évaluation.

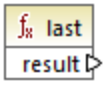



### Langages

XQuery, XSLT 2.0, XSLT 3.0.

## 6.7.23.6 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[804] actuel, tel que décrit dans l'exemple ci-dessous.




## Langages

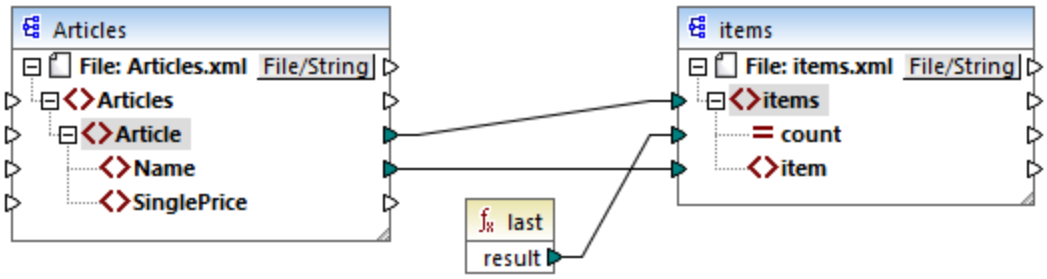
XQuery, XSLT 2.0, XSLT 3.0.

## 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[810] 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**[551] 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.

## 6.7.24 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

## 6.7.24.1 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

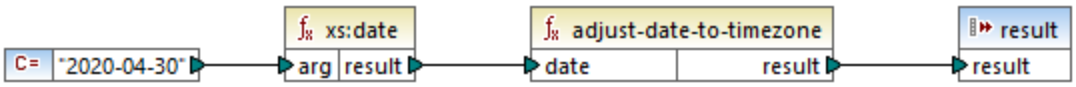
XQuery, XSLT 2.0, XSLT 3.0.

### 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`**.

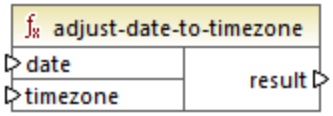



*Mappage XSLT 2.0*

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`**.

## 6.7.24.2 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

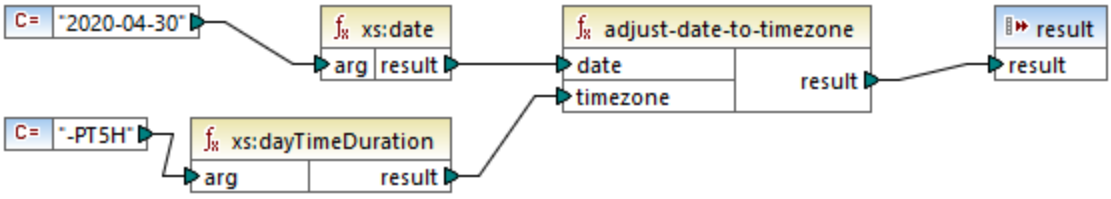
XQuery, XSLT 2.0, XSLT 3.0.

## 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 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[704] correspondantes. L'objectif du mappage est d'ajuster le fuseau horaire à -5 heures. Ce fuseau horaire peut être exprimé en tant que **`-PT5H`**.

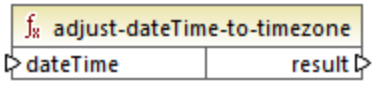



*Mappage XSLT 2.0*

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`**.

# 6.7.24.3 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

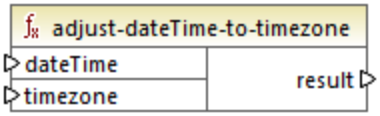
XQuery, XSLT 2.0, XSLT 3.0.

### Paramètres

| Nom | Type | Description |
|---|---|---|
| **dateTime** | `xs:dateTime` | La valeur d'entrée de type `xs:dateTime`. |

## 6.7.24.4 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`. |

## 6.7.24.5 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

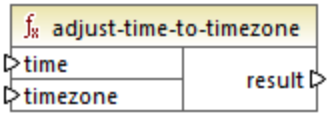
XQuery, XSLT 2.0, XSLT 3.0.

### Paramètres

| Nom | Type | Description |
|---|---|---|
| **time** | **xs:time** | La valeur d'entrée de type `xs:time`. |

## 6.7.24.6 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 |

| Nom | Type | Description |
|---|---|---|
| | | horaire de -5 heures peut être exprimée en tant que **-PT5H**. |

## 6.7.24.7 day-from-date

Retourne un `xs:integer` représentant la partie day de la valeur `xs:date` fournie en tant qu'argument.


### Langages

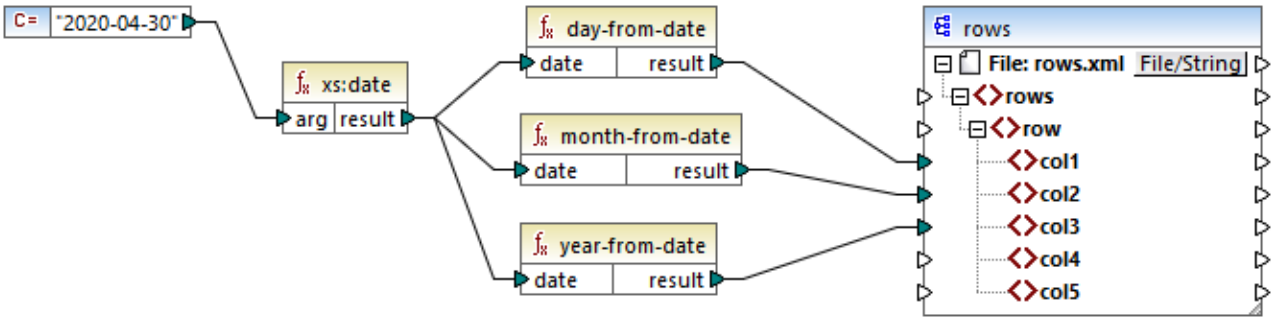
XQuery, XSLT 2.0, XSLT 3.0.

### 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*

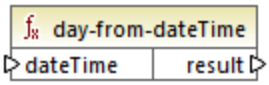
Le sortie de mappage est la suivante :

```
<rows>
   <row>
      <col1>30</col1>
```

```
      <col2>4</col2>
      <col3>2020</col3>
   </row>
</rows>
```

## 6.7.24.8 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`. |

## 6.7.24.9 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

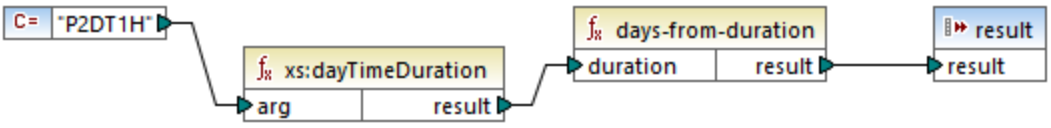
XQuery, XSLT 2.0, XSLT 3.0.

### 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).

## 6.7.24.10 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`. |

## 6.7.24.11 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).

## 6.7.24.12 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`. |

## 6.7.24.13 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`. |

## 6.7.24.14 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).

## 6.7.24.15 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`. |

## 6.7.24.16 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`. |

## 6.7.24.17 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`. |

## 6.7.24.18 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`. |

## 6.7.24.19 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` | |

## 6.7.24.20 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`. |

## 6.7.24.21 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`. |

## 6.7.24.22 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. |

## 6.7.24.23 subtract-dates

Retourne le `xs:dayTimeDuration` qui correspond à la différence entre la valeur normalisée de **date1** et la valeur normalisée de **date2**.

### Langages

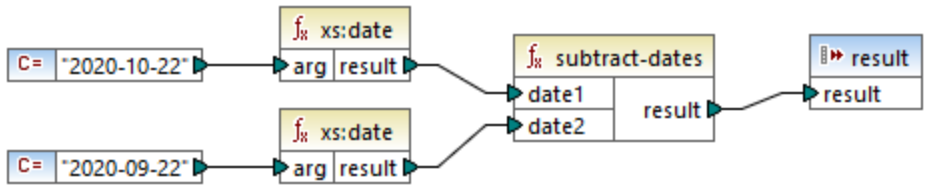
XQuery, XSLT 2.0, XSLT 3.0.

### Paramètres

| Nom | Type | Description |
|---|---|---|
| **date1** | `xs:date` | La première valeur d'entrée. |
| **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.




## 6.7.24.24 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. |

## 6.7.24.25 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`. |

## 6.7.24.26 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`. |

## 6.7.24.27 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`. |

## 6.7.24.28 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`. |

## 6.7.24.29 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`. |

## 6.7.24.30 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`. |

## 6.7.25 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** 727). 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** 725).
- 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** 725).
- Avec un argument qui doit être un nœud : la fonction est appliquée au nœud contextuel.

### 6.7.25.1 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

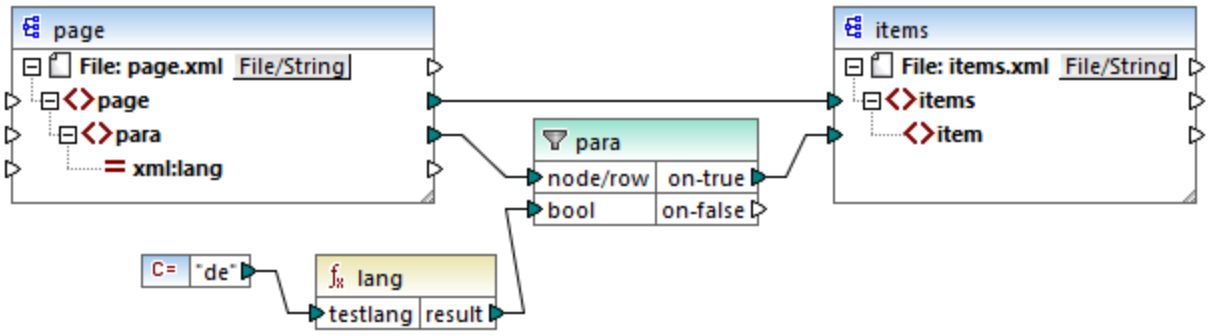
| Nom | Type | Description |
|---|---|---|
| **testlang** | **xs:string** | Le code de langage à vérifier, par exemple, "en". |

### 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 :

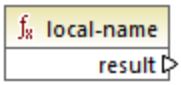
```
<items>
   <item>Grüss Gott!</item>
   <item>Guten Tag!</item>
   <item>Grüezi!</item>
</items>
```

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 804.

## 6.7.25.2 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** 727 qui prend un nœud d'entrée en tant que paramètre.

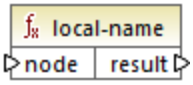



### Langages

XQuery, XSLT 2.0, XSLT 3.0.

## 6.7.25.3 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

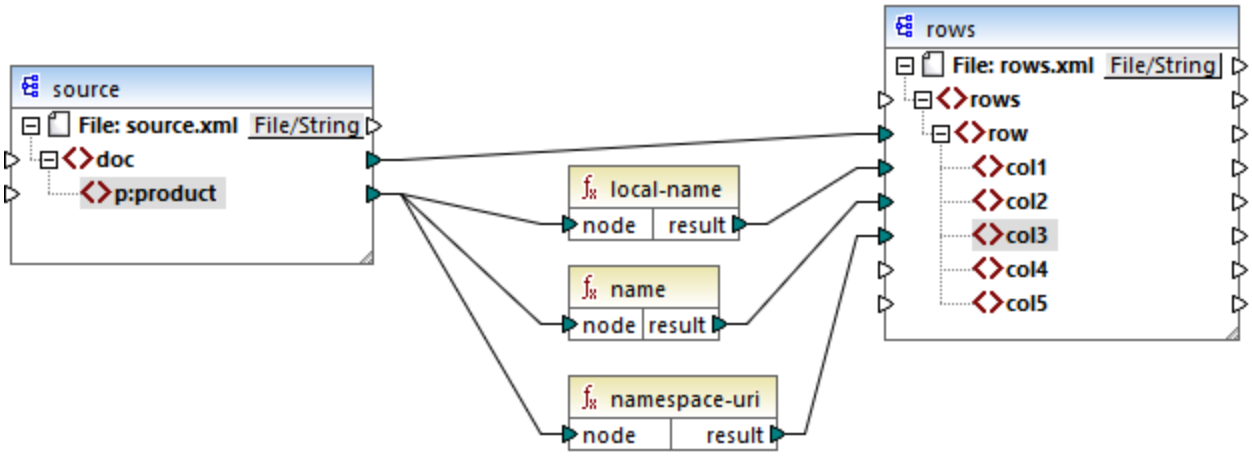
| Nom | Type | Description |
|---|---|---|
| **node** | **node()** | Le nœud d'entrée. |

### 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 :



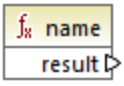

*Mappage XSLT 2.0*

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>
```

## 6.7.25.4 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** 729 qui prend un nœud d'entrée en tant que paramètre.

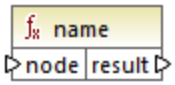



### Langages

XQuery, XSLT 2.0, XSLT 3.0.

## 6.7.25.5 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`** 727.

## 6.7.25.6 namespace-uri

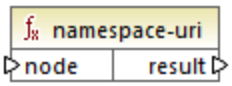
Retourne l'espace de nom URI du QName du nœud contextuel en tant `xs:string`. Il s'agit d'une variante sans paramètres de la fonction **`namespace-uri`** où le nœud contextuel est déterminé par les connexions dans votre mappage. Pour spécifier un nœud explicitement, utiliser la fonction **`namespaces- uri`** 729 qui prend un nœud d'entrée en tant que paramètre.


### Langages

XQuery, XSLT 2.0, XSLT 3.0.

## 6.7.25.7 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

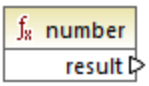
| Nom | Type | Description |
|---|---|---|
| **node** | `node()` | Le nœud d'entrée. |

### Exemple
Voir l'exemple donné pour la fonction **local-name**[727].

## 6.7.25.8 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**[730] qui prend un nœud d'entrée en tant que paramètre.

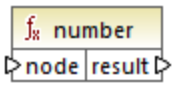
Les seuls types qui peuvent être convertis dans des nombres sont des booléennes, des strings numériques et d'autres types numériques. Les valeurs d'entrée non-numériques (comme un string non-numérique) résultent en NaN (Not a Number - Pas un Nombre).




### Langages
XQuery, XSLT 2.0, XSLT 3.0.

## 6.7.25.9 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

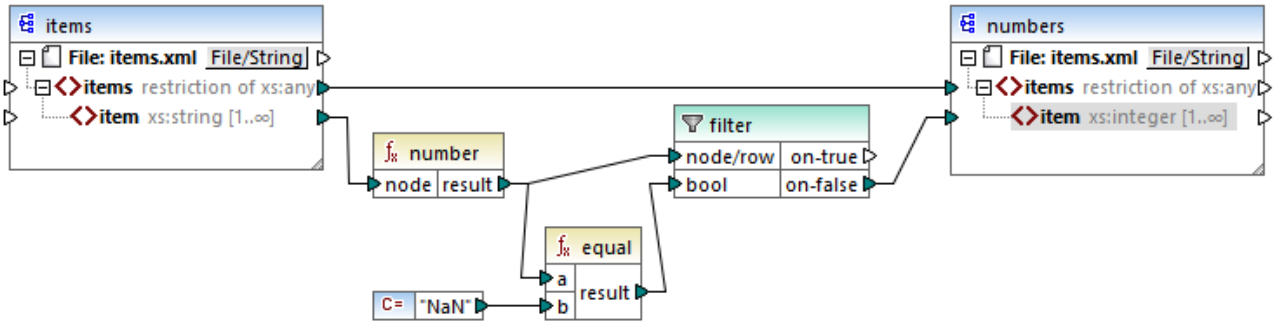
| Nom | Type | Description |
|---|---|---|
| **node** | `mf:atomic` | Le nœud d'entrée. |

### 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 :

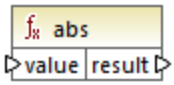
```
<items>
   <item>1</item>
   <item>2</item>
</items>
```

## 6.7.26 xpath2 | numeric functions

Les fonctions numériques de la bibliothèque **xpath2** comprennent les fonctions **`abs`** et **`round-half-to-even`**.

## 6.7.26.1 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.

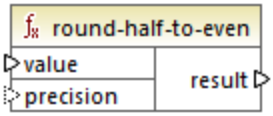
### Paramètres

| Nom | Type | Description |
|---|---|---|
| **value** | `xs:decimal` | La valeur d'entrée. |

## 6.7.26.2 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. |

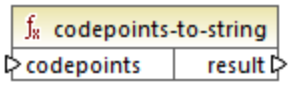
| Nom | Type | Description |
|---|---|---|
| **precision** | `xs:integer` | Argument optionnel qui spécifie le nombre de décimales à arrondir. La valeur par défaut est de **0**. |

## 6.7.27 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.).

### 6.7.27.1 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`[741].




#### 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. |

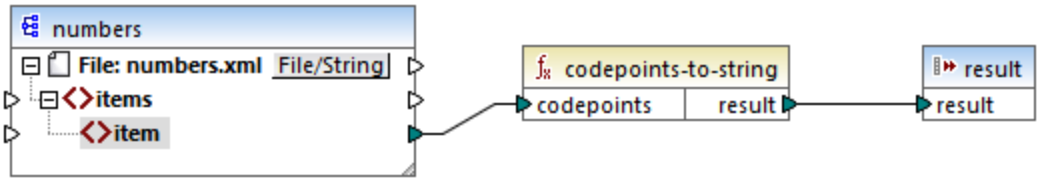
#### Exemple

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**.



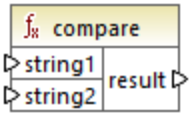

*Mappage XSLT 2.0*

La sortie de mappage est **MapForce**.

## 6.7.27.2 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[735] de cette fonction existe là où vous pouvez fournir la collation en tant qu'argument.




### Langages

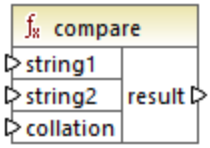
XQuery, XSLT 2.0, XSLT 3.0.

### Paramètres

| Nom | Type | Description |
|---|---|---|
| **string1** | xs:string | Le premier string d'entrée. |
| **string2** | xs:string | Le second string d'entrée. |

## 6.7.27.3 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

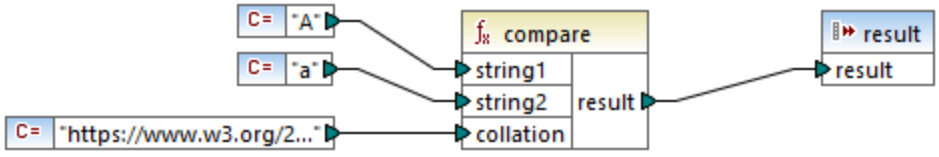
XQuery, XSLT 2.0, XSLT 3.0.

### 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`**[707] ou il peut s'agir d'une collation comme `http://www.w3.org/2005/xpath-functions/collation/html-ascii-case-insensitive`. |

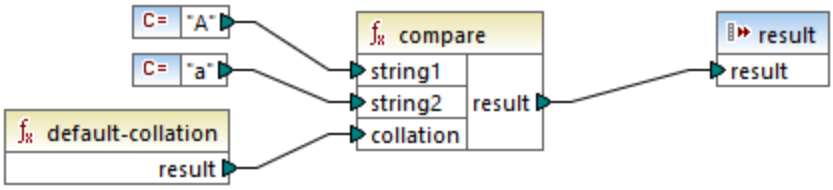
### Exemple

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*

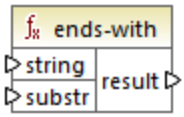
Le résultat du mappage ci-dessus est **0**, signifiant que les deux strings sont traités à égalité. Néanmoins, si vous remplacez la collation avec celle fournie par la fonction **default-collation**, la collation change pour passer à la collation de point de code Unicode par défaut, et le résultat de mappage devient **-1** ("A" est alphabétiquement inférieur à "a").




## 6.7.27.4 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[737] de cette fonction existe là où vous pouvez fournir la collation en tant qu'argument.




### Langages

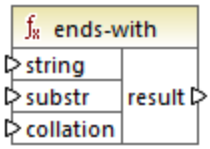
XQuery, XSLT 2.0, XSLT 3.0.

### 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"). |

## 6.7.27.5 ends-with

Retourne **true** si **string** se termine avec **substr**; **false** sinon. La valeur retournée est de type `xs:boolean`.




### Langages

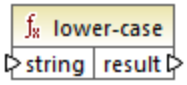
XQuery, XSLT 2.0, XSLT 3.0.

### 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`**[707] ou il peut s'agir d'une collation comme `http://www.w3.org/2005/xpath-functions/collation/html-ascii-case-insensitive`. |

## 6.7.27.6 lower-case

Retourne la valeur de **string** après avoir traduit chaque caractère dans son correspondant en minuscule.




### Langages

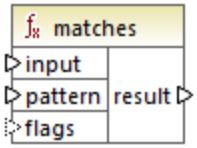
XQuery, XSLT 2.0, XSLT 3.0.

### Paramètres

| Nom | Type | Description |
|---|---|---|
| **string** | `xs:string` | La valeur d'entrée. |

## 6.7.27.7 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.

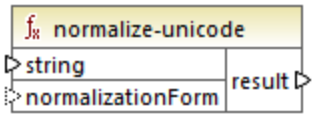
### Paramètres

| Nom | Type | Description |
|---|---|---|
| **input** | `xs:string` | Type |
| **pattern** | `xs:string` | L'expression régulière à faire correspondre, voir Expressions régulière[543]. |
| **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 :<br>`i` Utiliser le mode insensible à la casse. Le défaut est sensible à la casse.<br>`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.<br>x Ignorer l'espace blanc. Par défaut, les caractères d'espace blanc ne sont pas ignorés. |

## 6.7.27.8 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

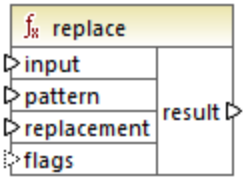
XQuery, XSLT 2.0, XSLT 3.0.

### 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).<br><br>Les formulaires de normalisation NFC, NFD, NFKC et NFKD sont pris en charge. |

## 6.7.27.9 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

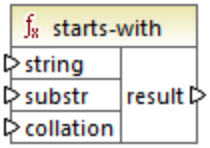
XQuery, XSLT 2.0, XSLT 3.0.

### Paramètres

| Nom | Type | Description |
|---|---|---|
| **input** | `xs:string` | Type |
| **pattern** | `xs:string` | L'expression régulière à faire correspondre, voir Expressions régulière 543. |
| **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` 738. |

## 6.7.27.10 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

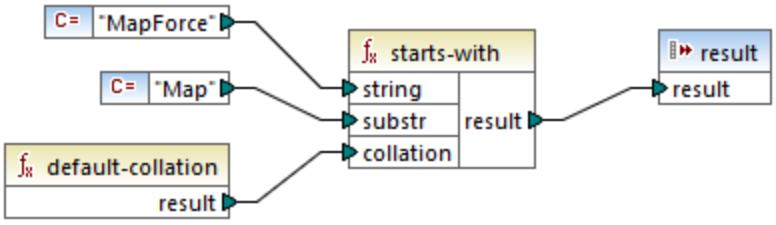
XQuery, XSLT 2.0, XSLT 3.0.

### 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`**[707] ou il peut s'agir d'une collation comme `http://www.w3.org/2005/xpath-functions/collation/html-ascii-case-insensitive`. |

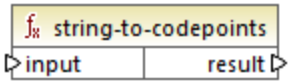
### Exemple

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.




## 6.7.27.11 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`**[733].




### Langages

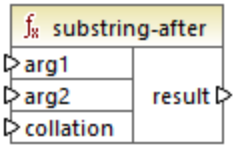
XQuery, XSLT 2.0, XSLT 3.0.

### Paramètres

| Nom | Type | Description |
|---|---|---|
| **input** | `xs:string` | Le string d'entrée |

## 6.7.27.12 substring-after

Retourne la partie du string **arg1** qui se produit après le string **arg2**.




### Langages

XQuery, XSLT 2.0, XSLT 3.0.

### Paramètres

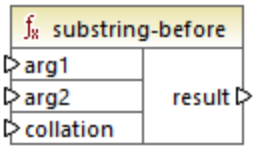
| Nom | Type | Description |
|---|---|---|
| **arg1** | `xs:string` | Le string d'entrée (c'est à dire la "pile de foin"). |
| **arg2** | `xs:string` | Le sous-string (c'est à dire l'"aiguille"). |
| **collation** | `xs:string` | Spécifie la collation à utiliser pour la comparaison de string. Cette entrée peut provenir de la sortie de la fonction **`default-collation`** 707 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`** 707, la fonction retourne "Force".

## 6.7.27.13 substring-before

Retourne la partie du string **arg1** qui se produit avant le string **arg2**.




### Langages

XQuery, XSLT 2.0, XSLT 3.0.

### Paramètres

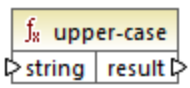
| Nom | Type | Description |
|---|---|---|
| **arg1** | `xs:string` | Le string d'entrée (c'est à dire la "pile de foin"). |
| **arg2** | `xs:string` | Le sous-string (c'est à dire l'"aiguille"). |
| **collation** | `xs:string` | Spécifie la collation à utiliser pour la comparaison de string. Cette entrée peut provenir de la sortie de la fonction **`default-collation`** 707 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`** 707, la fonction retourne "Map".

## 6.7.27.14 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

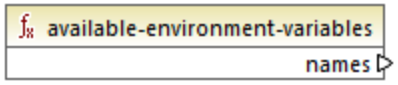
| Nom | Type | Description |
|---|---|---|
| **string** | `xs:string` | Type |

## 6.7.28 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.

### 6.7.28.1 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).

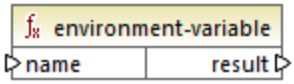



### Langages

XSLT 3.0.

### 6.7.28.2 environment-variable

Retourne la valeur d'une variable d'environnement système, si elle existe. The type de retour est `xs:string`.




### Langages

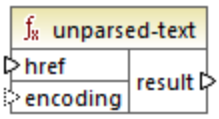
XSLT 3.0.

### Paramètres

| Nom | Type | Description |
|---|---|---|
| **nom** | `xs:string` | Le nom de la varaible d'environnement. |

## 6.7.28.3 unparsed-text

Lit une ressource externe (par exemple, un fichier) et retourne une représentation string de la ressource.




### Langages

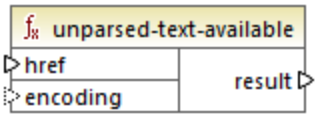
XSLT 3.0.

### 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. |

## 6.7.28.4 unparsed-text-available

Détermine si un appel vers **`unparsed-text`** avec des arguments particuliers réussirait. Le type de retour est `xs:boolean`.



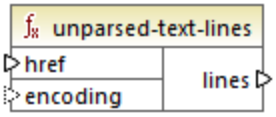
### Langages
XSLT 3.0.

### 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. |

## 6.7.28.5 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

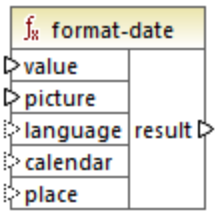
| Nom | Type | Description |
|---|---|---|
| **href** | `xs:string` | Un string sous forme d'une référence URI. |
| **encoding** | `xs:string` | Argument optionnel. Spécifie le nom de l'encodage, par exemple "UTF-8", "UTF-16". Si l'encodage ne peut pas être déterminé automatiquement, alors UTF-8 est présumé. |

## 6.7.29 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.

### 6.7.29.1 format-date

Retourne un string contenant une `xs: date` valeur formatée pour l'affichage.




#### Langages

XSLT 3.0.

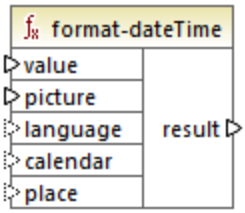
#### Paramètres

| Nom | Type | Description |
|---|---|---|
| **value** | `xs:date` | Fournit la valeur `xs: date` à formater. Paramètre obligatoire. |
| **image** | `xs:string` | Paramètre obligatoire.<br><br>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.<br><br>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. |

## 6.7.29.2 format-dateTime

Retourne un string contenant une `xs: dateTime` valeur formatée pour l'affichage.




### Langages

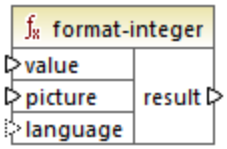
XSLT 3.0.

### Paramètres

| Nom | Type | Description |
|---|---|---|
| **value** | `xs:dateTime` | Fournit la valeur `xs: dateTime` à formater. |
| **image** | `xs:string` | Paramètre obligatoire.<br><br>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.<br><br>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. |

## 6.7.29.3 format-integer

Formate un entier conformément à un string d'image donné, utilisant les conventions d'un langage naturel donné, si spécifié.




### Langages

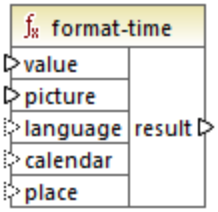
XSLT 3.0.

### Paramètres

| Nom | Type | Description |
|---|---|---|
| **value** | `xs:integer` | La valeur de l'entier de l'entrée à formater. |
| **image** | `xs:string` | Paramètre obligatoire.<br><br>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.<br><br>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). |

## 6.7.29.4 format-time

Retourne un string contenant une `xs:time` valeur formatée pour l'affichage.




### Langages

XSLT 3.0.

### Paramètres

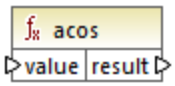
| Nom | Type | Description |
|---|---|---|
| **value** | `xs:time` | Fournit la valeur `xs:time` à formater. |
| **image** | `xs:string` | Paramètre obligatoire.<br><br>Voir section 9.8.4.1 de la Recommandation W3C « XPath and XQuery Functions and Operators 3.1 » (https://www.w3.org/TR/xpath-functions-31). |
| **language** | `xs:string` | Paramètre optionnel.<br><br>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. |

## 6.7.30 xpath3 | math functions

Les fonctions mathématiques de la bibliothèque **xpath3** sont utilisées pour effectuer des calculs trigonométriques ou autres calculs mathématiques.

### 6.7.30.1 acos

Retourne l'arc cosinus d'un angle, allant de **0** à **pi**.




#### Langages

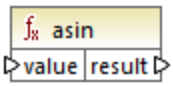
XSLT 3.0.

#### Paramètres

| Nom | Type | Description |
|---|---|---|
| **value** | `xs:double` | La valeur d'entrée. |

### 6.7.30.2 asin

Retourne l'arc sinus d'un angle, allant de **-pi/2** à **pi/2**.




#### Langages

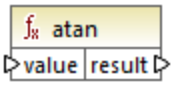
XSLT 3.0.

#### Paramètres

| Nom | Type | Description |
|---|---|---|
| **value** | `xs:double` | La valeur d'entrée. |

## 6.7.30.3 atan

Retourne l'arc tangente d'un angle, allant de **-pi/2** à **pi/2**.




### Langages

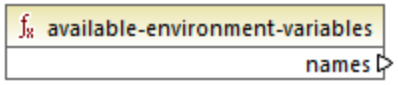
XSLT 3.0.

### Paramètres

| Nom | Type | Description |
|---|---|---|
| **value** | `xs:double` | La valeur d'entrée. |

## 6.7.30.4 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).

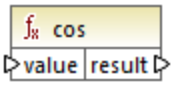



### Langages

XSLT 3.0.

## 6.7.30.5 cos

Retourne le cosinus trigonométrique de l'angle indiqué par la valeur. L'unité de la valeur est radian.




### Langages

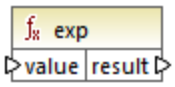
XSLT 3.0.

### Paramètres

| Nom | Type | Description |
|---|---|---|
| **value** | `xs:double` | La valeur d'entrée. |

## 6.7.30.6 exp

Retourne le nombre d'Euler *e* à la puissance de la valeur.




### Langages

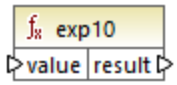
XSLT 3.0.

### Paramètres

| Nom | Type | Description |
|---|---|---|
| **value** | `xs:double` | La valeur d'entrée. |

## 6.7.30.7 exp10

Retourne 10 à la puissance de la valeur.




### Langages

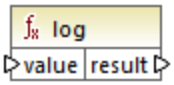
XSLT 3.0.

### Paramètres

| Nom | Type | Description |
|---|---|---|
| **value** | `xs:double` | La valeur d'entrée. |

## 6.7.30.8 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. |

## 6.7.30.9 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. |

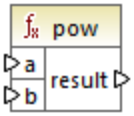
## 6.7.30.10 pi

Retourne une approximation vers la constante mathématique **pi**.

### Langages
XSLT 3.0.

## 6.7.30.11 pow

Retourne la valeur de **a** élevé à la puissance de **b**.



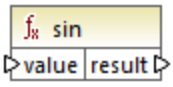

### Langages
XSLT 3.0.

### Paramètres

| Nom | Type | Description |
|---|---|---|
| **a** | `xs:double` | La valeur d'entrée **a**. |
| **b** | `xs:double` | La valeur d'entrée **b**. |

## 6.7.30.12 sin

Retourne le sinus trigonométrique d'un angle indiqué par la valeur. L'unité de valeur est en radian.



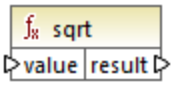

### Langages
XSLT 3.0.

### Paramètres

| Nom | Type | Description |
|---|---|---|
| **value** | `xs:double` | La valeur d'entrée. |

### 6.7.30.13 sqrt

Retourne la racine carrée non négative de l'argument.




#### Langages

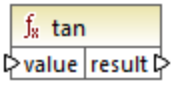
XSLT 3.0.

#### Paramètres

| Nom | Type | Description |
|---|---|---|
| **value** | `xs:double` | La valeur d'entrée. |

### 6.7.30.14 tan

Retourne la tangente trigonométrique de l'angle indiquée par la valeur. L'unité de valeur est en radian.




#### Langages

XSLT 3.0.

#### Paramètres

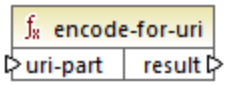
| Nom | Type | Description |
|---|---|---|
| **value** | `xs:double` | La valeur d'entrée. |

## 6.7.31 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.

## 6.7.31.1 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

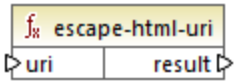
XSLT 3.0.

### Paramètres

| Nom | Type | Description |
|---|---|---|
| **uri-part** | `xs:string` | La valeur d'entrée de l'URI à encoder. |

## 6.7.31.2 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

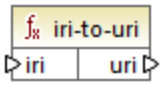
XSLT 3.0.

### Paramètres

| Nom | Type | Description |
|---|---|---|
| **uri** | `xs:string` | La valeur d'entrée de l'URI à échapper. |

### 6.7.31.3 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. |

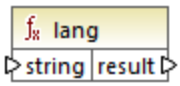
## 6.7.32 xslt | xpath functions

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[804] actuel qui est déterminé par les connexions dans votre mappage.

**Note :** vous trouverez des fonctions XPath 1.0 supplémentaires dans la bibliothèque **core**.

### 6.7.32.1 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

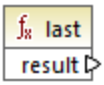
| Nom | Type | echo Hello, World! |
|---|---|---|
| **Paramètres** | `xs:string` | Le code de langage à vérifier, par exemple, "en". |

### Exemple

Voir l'exemple donné pour la fonction **lang**[725] de la bibliothèque **xpath2**.

## 6.7.32.2 last

Retourne le numéro de la position du dernier nœud dans la liste de nœud traitée.




### Langages

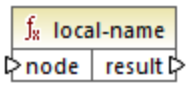
XSLT 1.0.

### Exemple

Voir l'exemple donné pour la fonction **last**[707] de la bibliothèque **xpath2**.

## 6.7.32.3 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

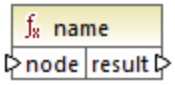
| Nom | Type | Description |
|---|---|---|
| **node** | `node()` | Le nœud d'entrée. |

### Exemple

Voir l'exemple donné pour la fonction **local-name**[727] de la bibliothèque **xpath2**.

## 6.7.32.4 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**[727] de la bibliothèque **xpath2**.

## 6.7.32.5 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

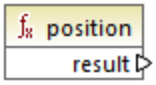
| Nom | Type | Description |
|---|---|---|
| **node** | **node()** | Le nœud d'entrée. |

### Exemple

Voir l'exemple donné pour la fonction `local-name`[727] de la bibliothèque **xpath2**.

## 6.7.32.6 position

Retourne la position du nœud actuel dans le nœud qui est traité actuellement.




### Langages

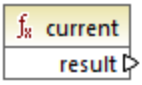
XSLT 1.0.

# 6.7.33 xslt | xslt functions

Les fonctions dans ce groupe sont des fonctions XSLT 1.0 diverses.

## 6.7.33.1 current

La fonction `current` ne prend aucun argument et retourne le nœud actuel.

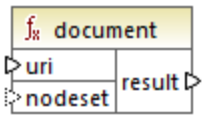



### Langages

XSLT 1.0.

## 6.7.33.2 document

Accède aux nœuds depuis un document XML externe. Le résultat est sorti dans un nœud dans le document de sortie.



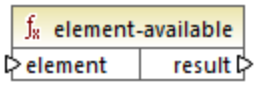
### Langages
XSLT 1.0.

### 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. |

## 6.7.33.3 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.



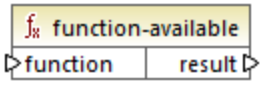

### Langages
XSLT 1.0.

### Paramètres

| Nom | Type | echo Hello, World! |
|---|---|---|
| **element** | `xs:string` | Le nom d'élément. |

## 6.7.33.4 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

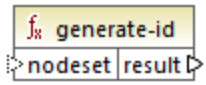
XSLT 1.0.

### Paramètres

| Nom | Type | echo Hello, World! |
|---|---|---|
| **fonction** | `xs:string` | Le nom de la fonction. |

## 6.7.33.5 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

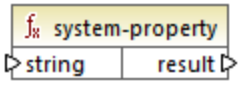
XSLT 1.0, XSLT 2.0, XSLT 3.0.

### Paramètres

| Nom | Type | Description |
|---|---|---|
| **nodeset** | `node()` | Argument optionnel qui fournit le nœud d'entrée. |

## 6.7.33.6 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

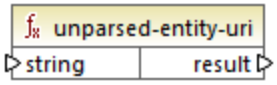
XSLT 1.0, XSLT 2.0, XSLT 3.0.

### 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`. |

## 6.7.33.7 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** 592 et **static-node-name** 594. 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.

**Note :** Dans le cas de CSV/FLF, l'accès dynamique implique l'accès à "fields" au lieu de "nodes", puisque les structures CSV/FLF n'ont pas de "nœuds".

Lorsque la cible de mappage est un fichier CSV ou FLF (champ de longueur fixe), les champs doivent être définis dans les paramètres de composant (et il n'est pas possible de changer le nom, l'ordre ou le nombre des champs de cible). Contrairement à XML, le format des fichiers de texte est fixe, donc seule la valeur de champ peut être manipulée, et pas le nom de champ, le nombre ou l'ordre.

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 767. Pour un exemple de mappage étape par étape, voir Exemple : Mapper des noms d'élément vers les valeur d'attribut 779.

## 7.1.1 Obtenir l'accès aux noms de nœud

Lorsqu'un nœud dans un composant XML (ou un champ dans un composant CSV/FLF) 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

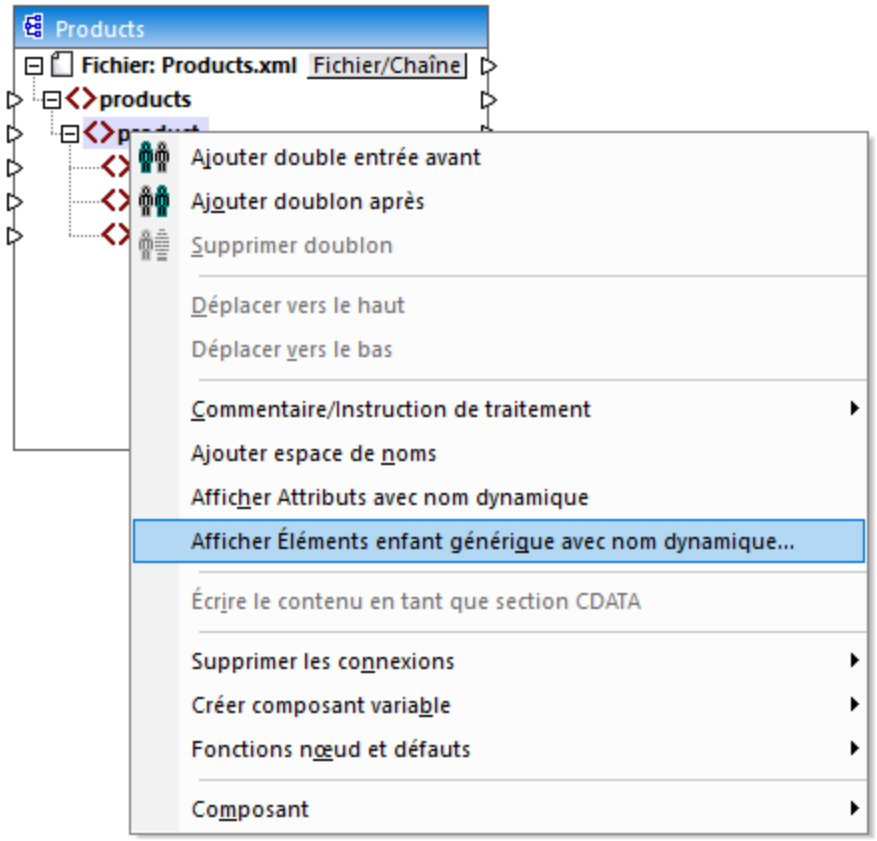



*Fig. 1 Activer des noms de nœud dynamiques (pour les éléments enfant)*

**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 775.

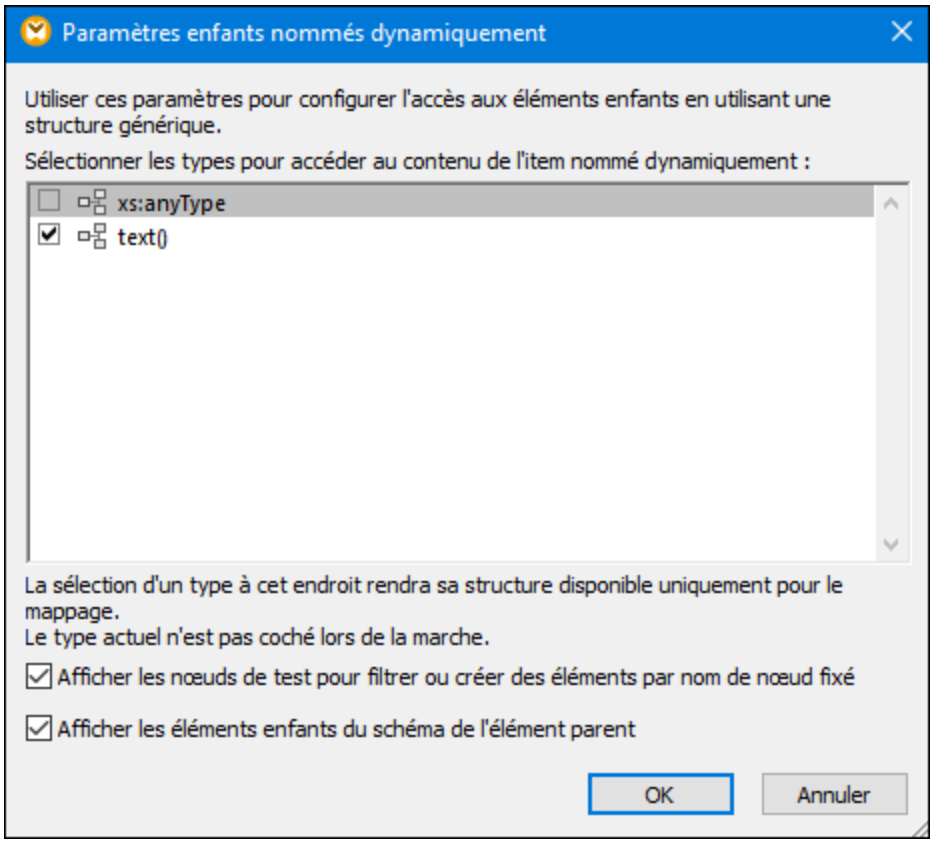



*Fig. 2 Dialogue "Paramètres d'Enfants nommés dynamiquement"*

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.

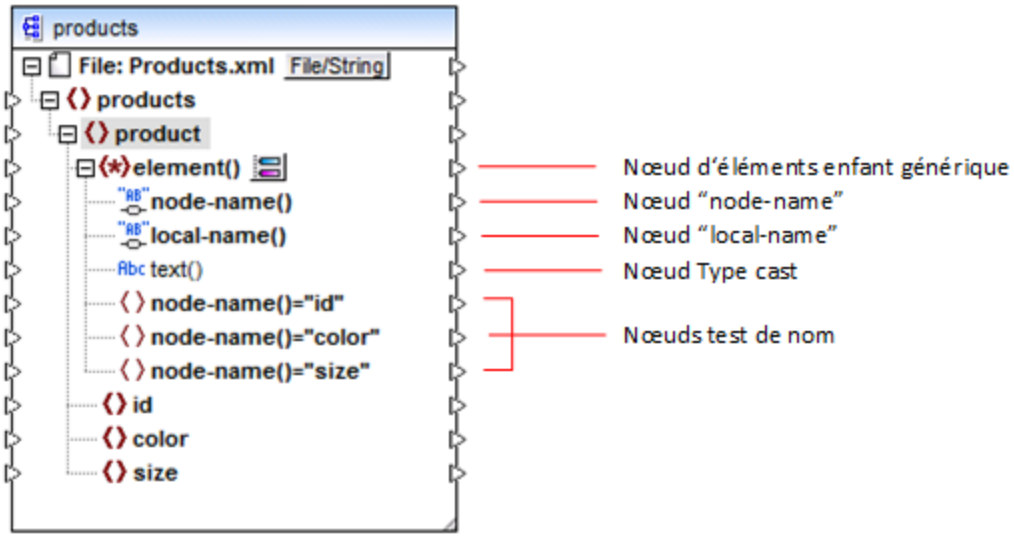



*Fig.3 Noms de nœud dynamiques activés (pour les éléments)*

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.

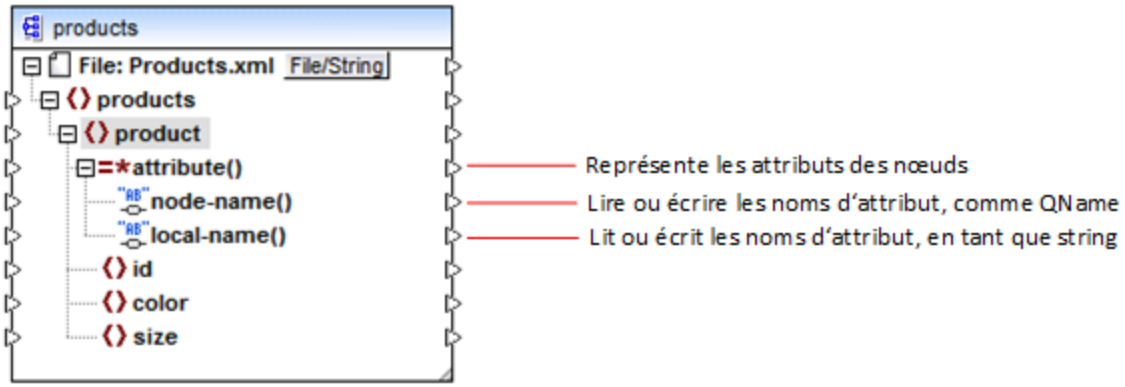



*Fig. 4 Noms de nœud dynamiques activés (pour les attributs)*

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. Par exemple, voir Exemple : Regrouper et filtrer les nœuds par nom 782.

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 :

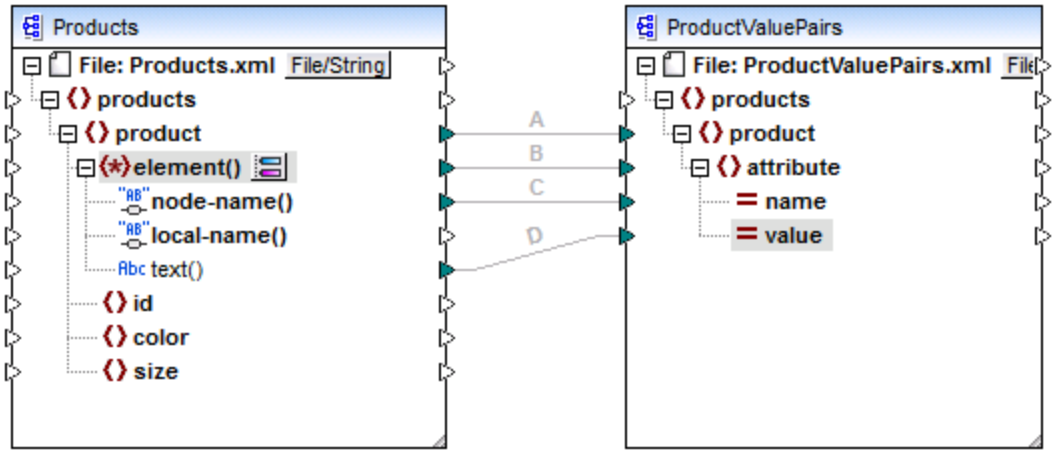



*Fig. 7 Mapper des noms d'élément XML dans les valeurs d'attribut (dans MapForce)*

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 779.

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 :

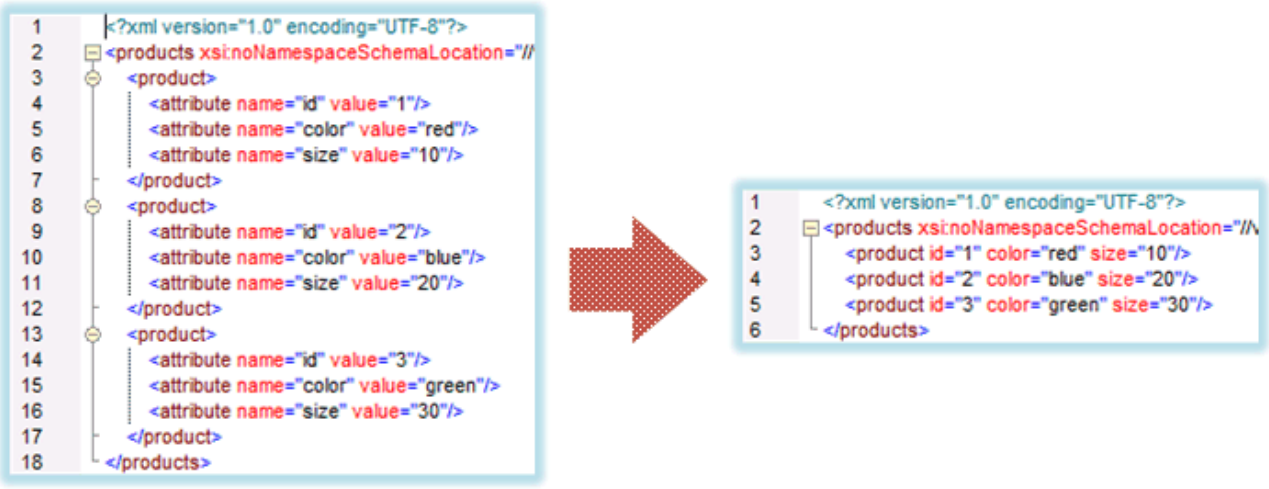



*Fig. 8 Mapper des valeurs d'attribut pour attribuer des noms (exigence)*

Le mappage qui permet d'atteindre cet objectif ressemble à ce qui suit :

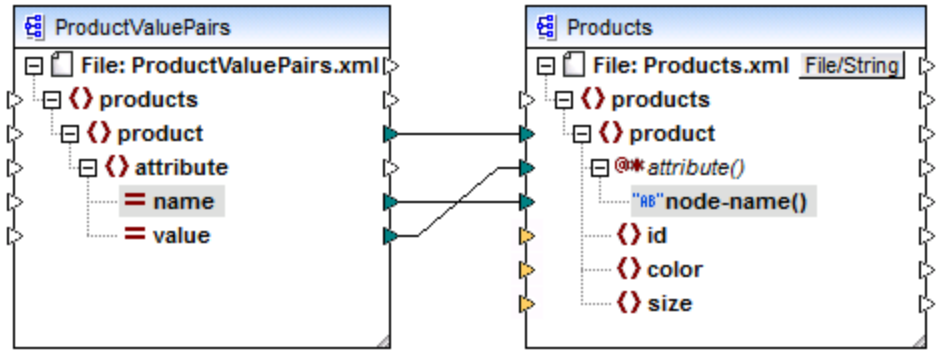



*Fig. 9 Mapper des valeurs d'attribut pour attribuer des noms (dans MapForce)*

**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), ou utiliser la fonction QName-as-string 682.

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 775.

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 775). Pour consulter un exemple, voir Exemple : Regrouper et filtrer les nœuds par nom 782.

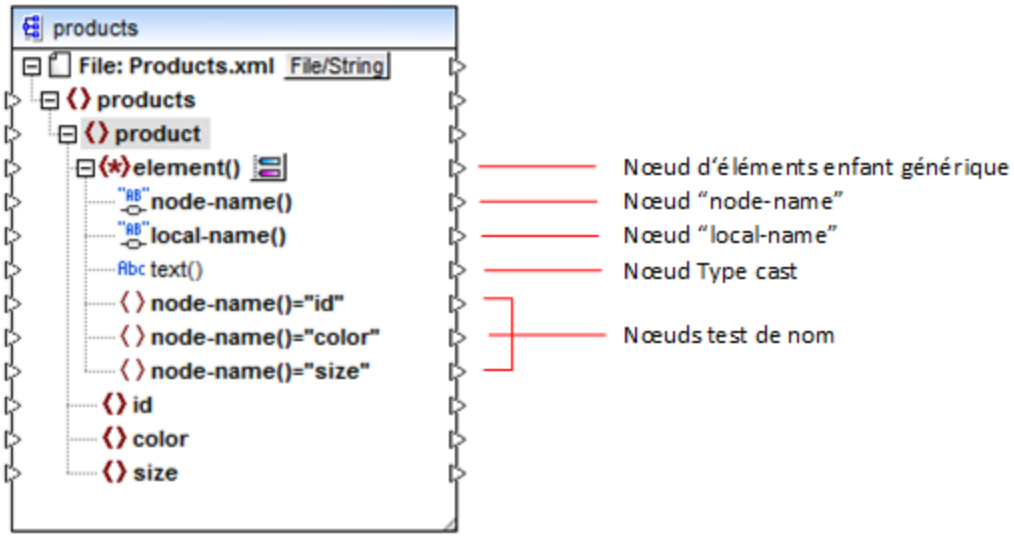
En général, les nœuds de test de nom fonctionnent presque comme des nœuds d'élément normaux pour lire et écrire des valeurs et des structures sous-arborescentes. Néanmoins, étant donné que la sémantique de mappage est différente lorsque l'accès dynamique est activé, il existe des limites. Par exemple, 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()`.

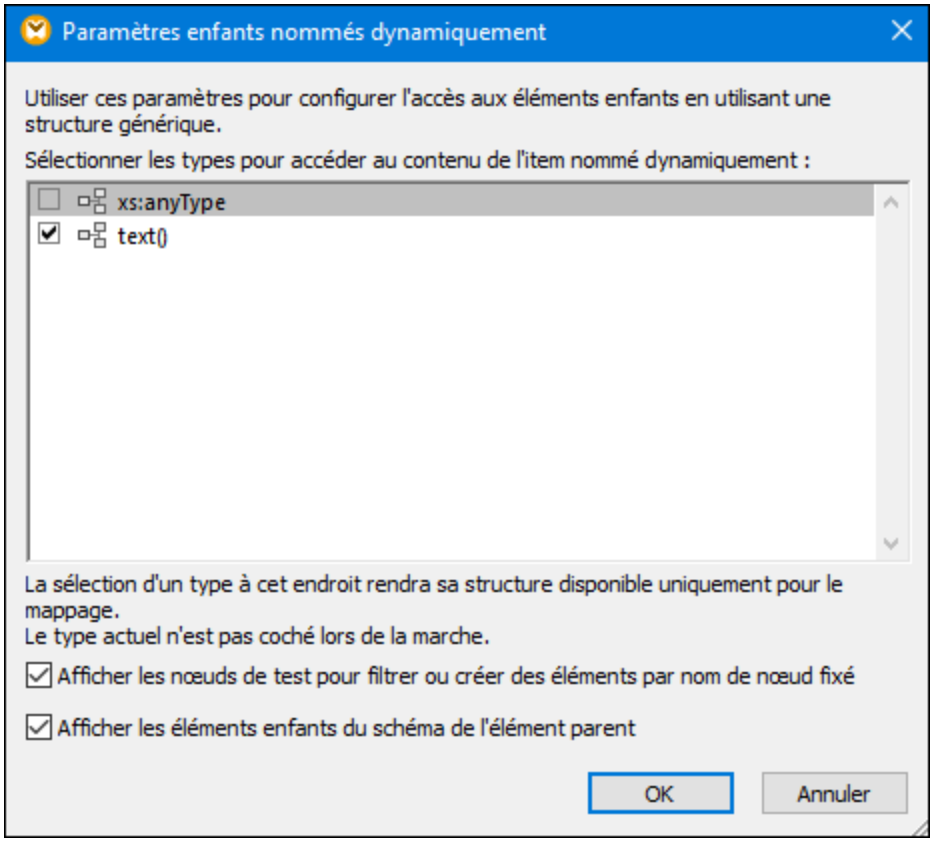
Si nécessaire, vous pouvez dissimuler les nœuds de test de nom depuis le composant. Pour ce faire, cliquer sur la touche **Changer la sélection** ( ) à côté du nœud `element()`. Ensuite, dans le dialogue "*Paramètres Enfants nommés dynamiquement*" (Fig. 2), décocher la case **Afficher les nœuds de test 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 767, 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 779. 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**.

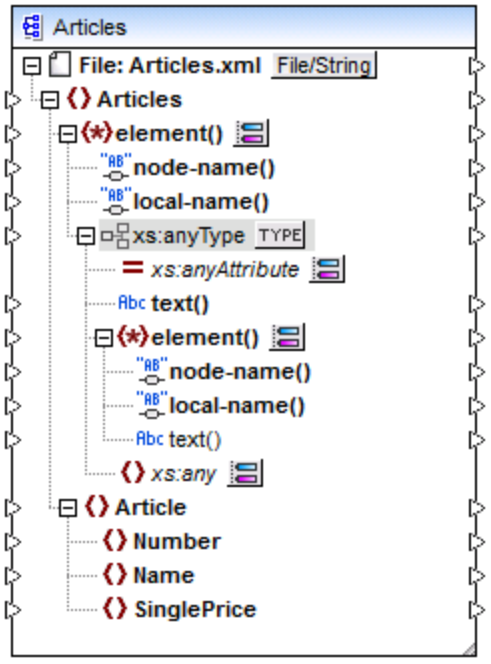
Veuillez noter que lors de l'exécution de mappage, MapForce (par le biais du nœud type cast) n'a pas d'informations par rapport à ce qu'est le type réel du nœud d'instance. C'est pourquoi, votre mappage doit accéder au contenu de nœud en utilisant le type correct. Par exemple, si vous prévoyez que le 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. Cette technique est illustrée dans Exemple : Regrouper et filtrer les nœuds par nom 782.

## 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 779, 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 129).
- 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 136). 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.

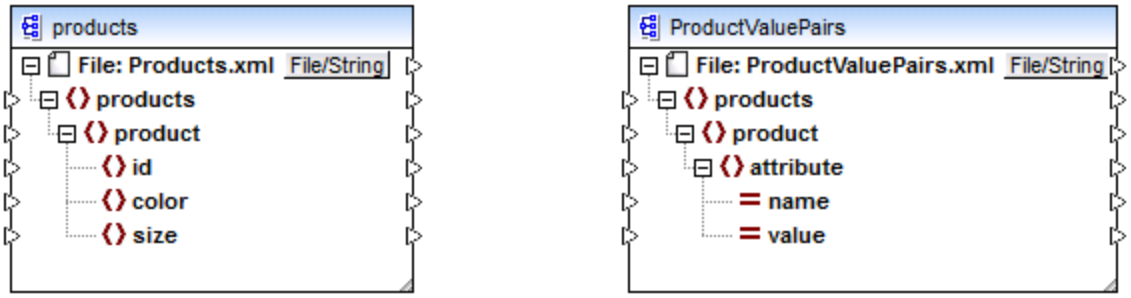
#### Étape 1: Ajouter le composant XML source au mappage

- 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

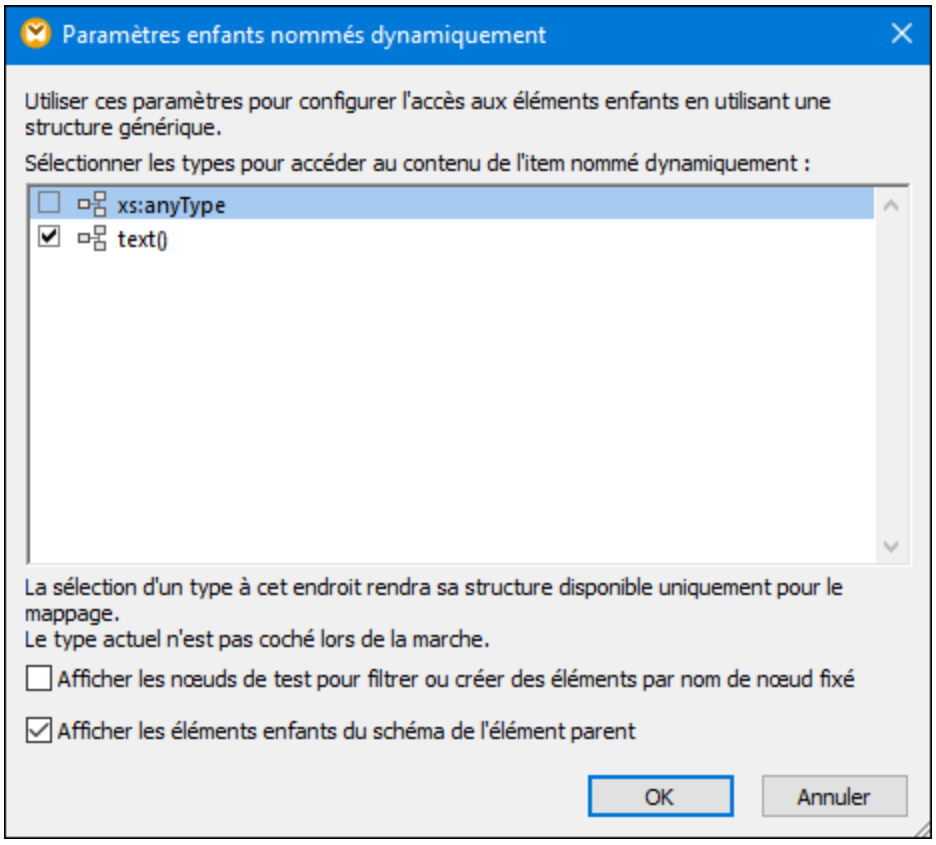
- Dans le menu **Insérer**, cliquer sur **Schéma XML/Fichier**, et chercher le fichier suivant : **<Documents>\Altova\MapForce2024\MapForceExamples\Tutorial\ProductValuePairs.xsd**. Lorsque vous êtes invité à fournir un fichier d'instance, cliquer sur **Sauter**. Lorsque vous êtes invité à choisir un élément racine, choisir `products` en tant qu'élément racine.

Á ce niveau, le mappage devrait ressembler à l'exemple suivant :

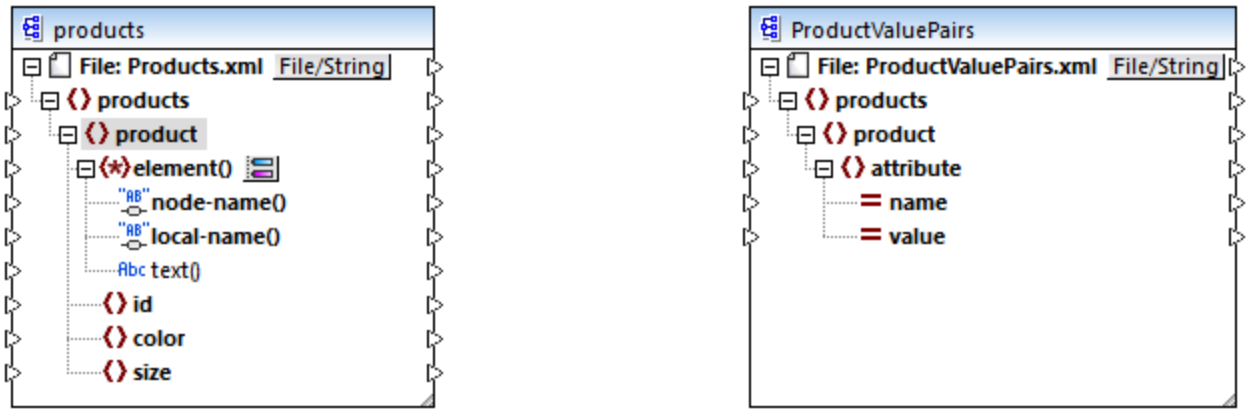



## Étape 3: Activer l'accès dynamique aux nœuds enfants

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.

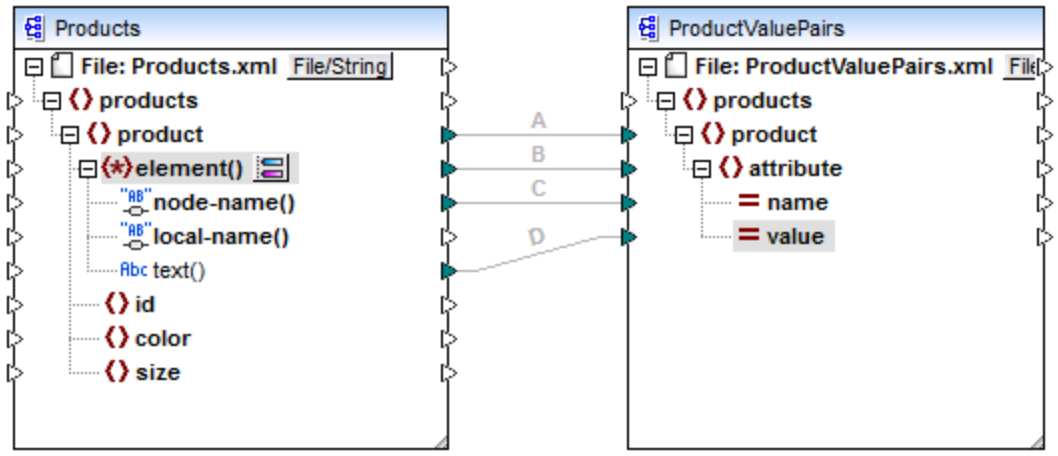



Veuillez noter qu'un nœud `text()` a été ajouté au composant de source. Ce nœud fournira le contenu de chaque item enfant au mappage (dans ce cas, la valeur de "id", "color", et "size").




## 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*

## 7.1.4 Exemple: Regrouper et filtrer les nœuds par nom

Cet exemple vous montre comment concevoir un mappage qui lit des paires clé-valeur depuis une liste de propriété XML (ou plist XML) et les écrit dans un fichier CSV. (Les listes de propriété XML représentent un moyen de stocker des informations d'objet macOS et iOS dans un format XML, voir https://developer.apple.com/library/mac/documentation/Cocoa/Conceptual/PropertyLists/UnderstandXMLPlist/UnderstandXMLPlist.html.) L'exemple est accompagné d'un échantillon de mappage qui est disponible sous le chemin suivant :
**<Documents>\Altova\MapForce2024\MapForceExamples\Tutorial\ReadPropertyList.mfd**.

Le listage de code ci-dessous représente le fichier XML de source.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist SYSTEM "https://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
   <dict>
      <key>First Name</key>
      <string>William</string>
      <key>Last Name</key>
      <string>Shakespeare</string>
      <key>Birthdate</key>
      <integer>1564</integer>
      <key>Profession</key>
      <string>Playwright</string>
      <key>Lines</key>
      <array>
         <string>It is a tale told by an idiot,</string>
         <string>Full of sound and fury, signifying nothing.</string>
      </array>
   </dict>
</plist>
```

L'objectif du mappage est de créer une nouvelle ligne dans le fichier CSV depuis certaines paires clé-valeur trouvée sous le nœud `<dict>` dans le fichier de liste de propriété. En particulier, le mappage doit uniquement filtrer des paires **<key>** - **<string>**. Les autres paires clé-valeurs (par exemple, **<key>** - **<integer>**) doivent être ignorées. Dans le fichier CSV, la ligne doit stocker le nom de la propriété, séparée de la valeur de la propriété par une virgule. Autrement dit, la sortie doit ressembler à l'exemple suivant :

```
Prénom,William
Nom de famille,Shakespeare
Profession,Playwright
```

Pour atteindre cet objectif, le mappage utilise un accès dynamique à tous les nœuds enfants du nœud `dict`. Ensuite, le mappage utilise la fonction group-starting-with[613] pour regrouper les paires clé-valeur extraites du fichier XML. Enfin, le mappage utilise un filtre pour ne filtrer que les nœuds dont le nom de nœud est "string".

Les étapes suivantes montrent comment le mappage requis peut être créé.

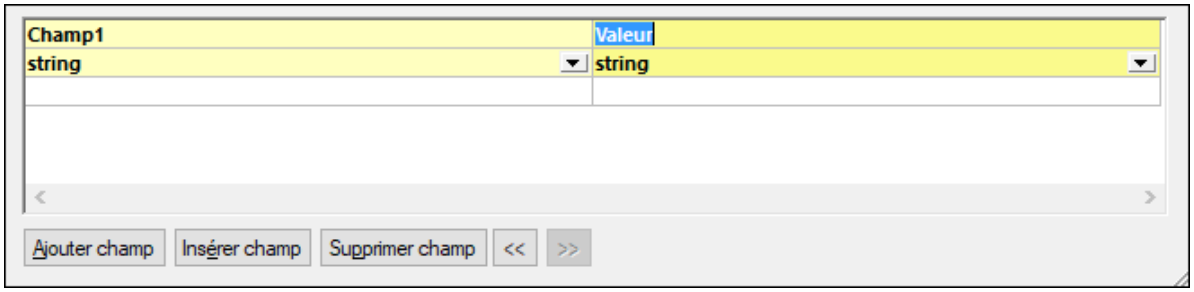
## Étape 1 : Ajouter le composant XML source au mappage

1. Le langage de transformation de mappage est configuré sur BUILT-IN[22].
2. Dans le menu **Insérer**, cliquez sur **Schéma XML/Fichier**, et chercher le fichier suivant : **<Documents>\Altova\MapForce2024\MapForceExamples\Tutorial\plist.xml**. Ce fichier XML pointe vers le schéma **plist.xml** situé dans le même dossier.

## Étape 2 : Ajouter le composant CSV cible au mappage

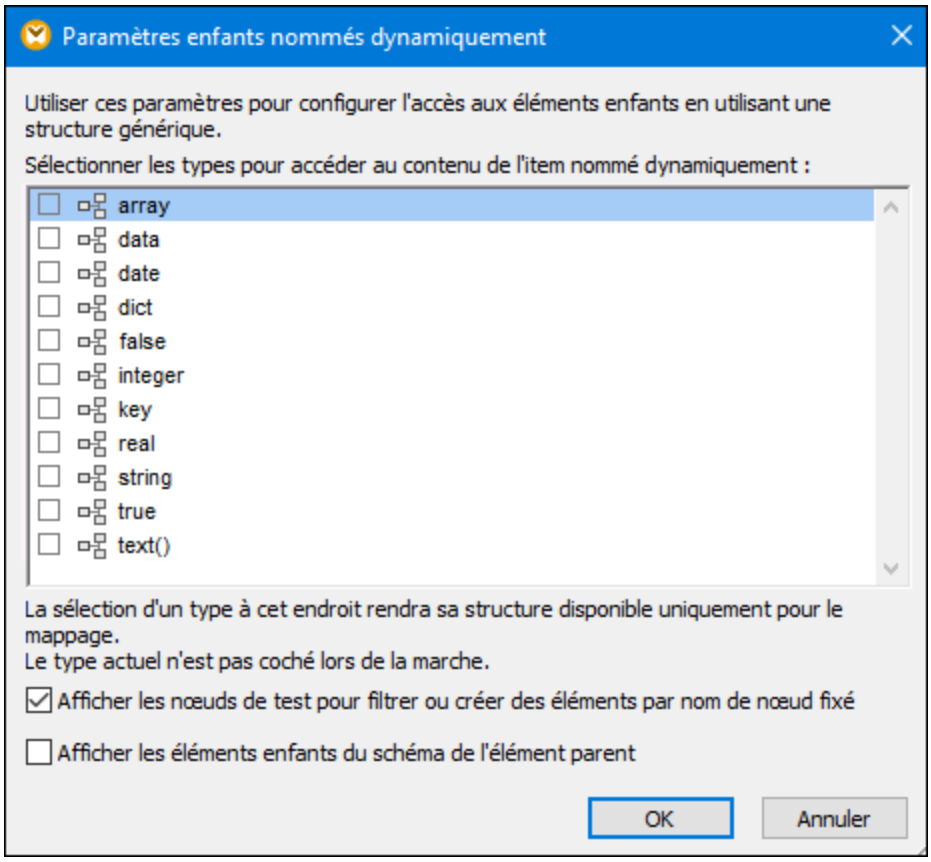
1. Dans le menu **Insérer**, cliquez sur **Fichier texte**. Sur invitation, choisir l'option **Utiliser un traitement simple pour les fichiers CSV standard...** ..
2. Ajouter un champ CSV dans le composant, en cliquant sur **Ajouter champ**.
3. Double-cliquez sur le nom de chaque champ et saisir "Key" en tant que nom du premier champ, et "Value" en tant que nom du second champ. Le champ "Key" stockera le nom de la propriété alors que

le champ "Value" stockera la valeur de la propriété. Pour plus d'informations concernant les composants CSV, voir Fichiers CSV et texte[373].

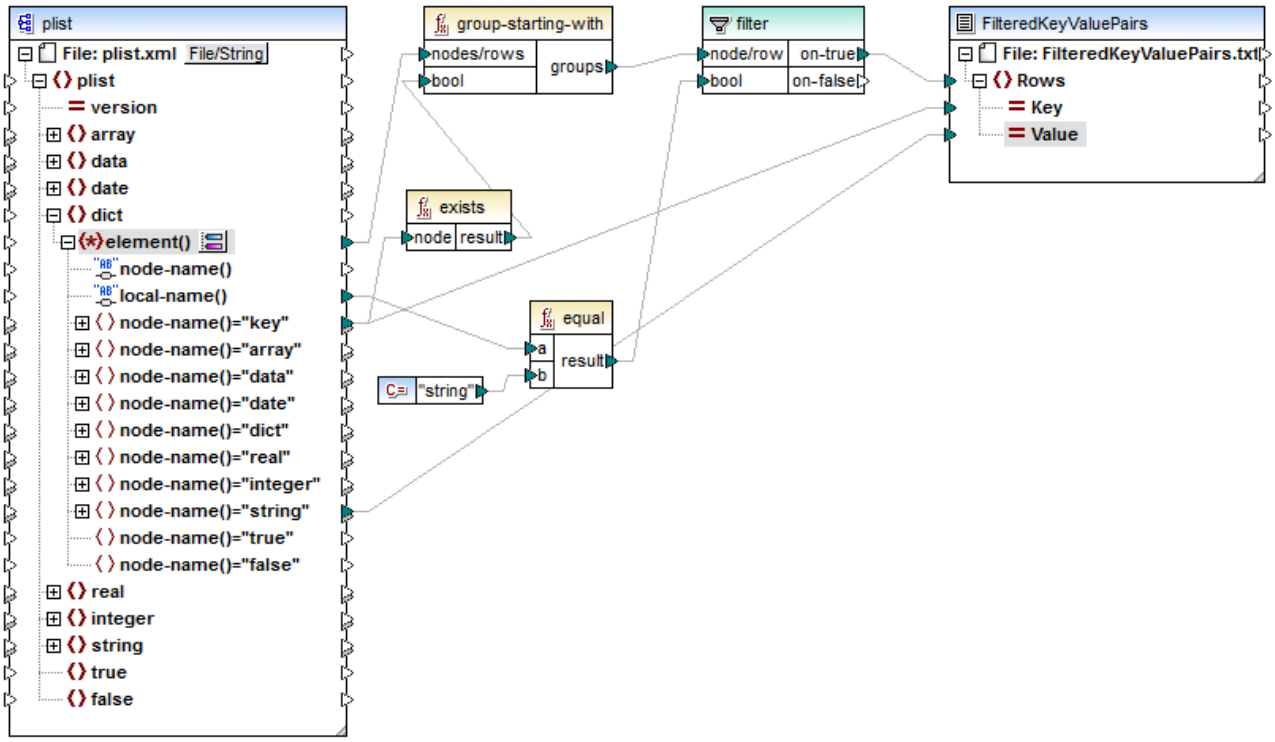



## Étape 3 : Ajouter le filtre et les fonctions

1. Glisser les fonctions **equal**[579], **exists**[599] et **group-starting-with**[613] depuis la fenêtre de bibliothèque dans le mappage. Pour des informations générales concernant les fonctions, voir Fonctions[467].
2. Pour ajouter le filtre, cliquez dans le menu **Insérer**, puis cliquez sur **Filtre : Nœuds/Lignes**. Pour des informations générales concernant les filtres, voir Filtres et conditions[444].
3. Dans le menu **Insérer**, cliquez sur **Constante**, puis saisissez le texte "chaîne".
4. Cliquez avec la touche de droite sur le nœud `dict` dans le composant de source et choisissez **Afficher les éléments enfants avec un Nom Dynamique** depuis le menu contextuel. Dans le dialogue "Paramètres des Enfants nommés dynamiquement", veuillez vous assurer que la case à cocher **Afficher les nœuds de test pour filtrer ou créer les élément par nom de nœud fixé** est bien sélectionnée.




5. Tracer les connexions comme affiché ci-dessous.




*ReadPropertyList.mfd*

## Explication du mappage

L'élément `element()` dans le composant source fournit à tous les enfants du le nœud `dict`, comme séquence à la fonction **`group-starting-with`**. La fonction **`group-starting-with`** crée un nouveau groupe à chaque fois qu'un nœud portant le nom `key` est rencontré. La fonction **`exists`** vérifie cette condition et retourne le résultat en tant que valeur booléenne vrai/faux de la fonction de regroupement.

Pour chaque groupe, le filtre vérifie si le nom du nœud actuel est égal à "string", avec l'aide de la fonction **`equal`**. Le nom lui-même est lu depuis le `local-name()`, qui fournit le nom du nœud en tant que string.

Les connexions vers le composant cible ont le rôle suivant :

- Une nouvelle ligne est créée dans le CSV cible uniquement si la condition de filtre est vraie.
- `Key` (nom de propriété) est prélevé depuis la valeur de l'élément `key` dans la source.
- `Value` (valeur de de propriété) est prélevé depuis le nœud de test de nom `string` name.

## 7.2 Fichiers Batch-Process

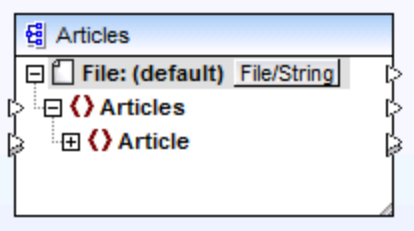
Vous pouvez configurer MapForce pour qu'il traite plusieurs fichiers (par exemple, tous les fichiers dans un répertoire) lorsque le mappage est en cours. L'utilisation de cette fonction vous permettra de résoudre les tâches suivantes :

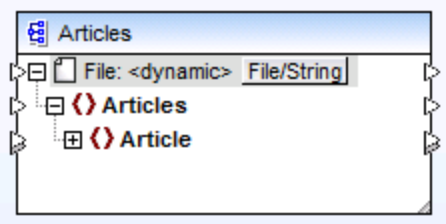
- Fournir dans le mappage une liste des fichiers d'entrée à traiter
- Générer en tant que sortie de mappage une liste des fichiers au lieu du fichier de sortie unique
- Générez une application de mappage où les noms de fichiers d'entrée et de sortie sont définis au moment de l'exécution
- Convertissez un ensemble de fichiers vers un autre format
- Partager un fichier volumineux (ou une base de données) dans des parties plus petites
- Fusionner plusieurs fichiers en un grand fichier (ou les charger dans une base de données)

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 44). 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.<br><br>Si vous choisissez cette option, le nœud de racine ne nécessite pas de connecteur d'entrée, étant donné que cela n'est pas utile.<br><br>  |

| | |
|---|---|
| | 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.<br><br>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.<br><br>  <br><br>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.<br><br>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.<br><br>Lorsque cette option est sélectionnée, le nom du nœud de racine est affiché en tant que **Fichier : <dynamic>**.<br><br>  <br><br>Cette option est mutuellement exclusive avec l'option *Utiliser Noms de fichier depuis les Paramètres de composant*. |
| *Parser strings vers XML, Parser strings vers JSON, Parser strings vers CSV, Parser Strings vers FLF, Parse strings vers EDI* | Lorsqu'elle est activée, cette option permet au composant d'accepter une valeur de string en tant qu'entrée dans le nœud de racine et de la convertir dans une structure XML, |

| | |
|---|---|
| | JSON, CSV, FLF ou EDI, respectivement. Pour plus d'informations, voir Parser et sérialiser des strings 794. |
| *Sérialiser XML vers Strings, Sérialiser JSON vers Strings, Sérialiser CSV vers Strings, Sérialiser FLF vers Strings, Sérialiser EDI vers Strings* | Lorsqu'elle est activée, cette option permet au composant d'accepter une structure en tant qu'entrée, et de la convertir en un string. La structure d'entrée peut être XML, JSON, CSV, Champ de longueur fixe, ou EDI, respectivement. Pour plus d'informations, voir Parser et sérialiser des strings 794. |

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) | * |
| XQuery | * | *(1) | N'est pas pris en charge par XQuery |
| 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 |

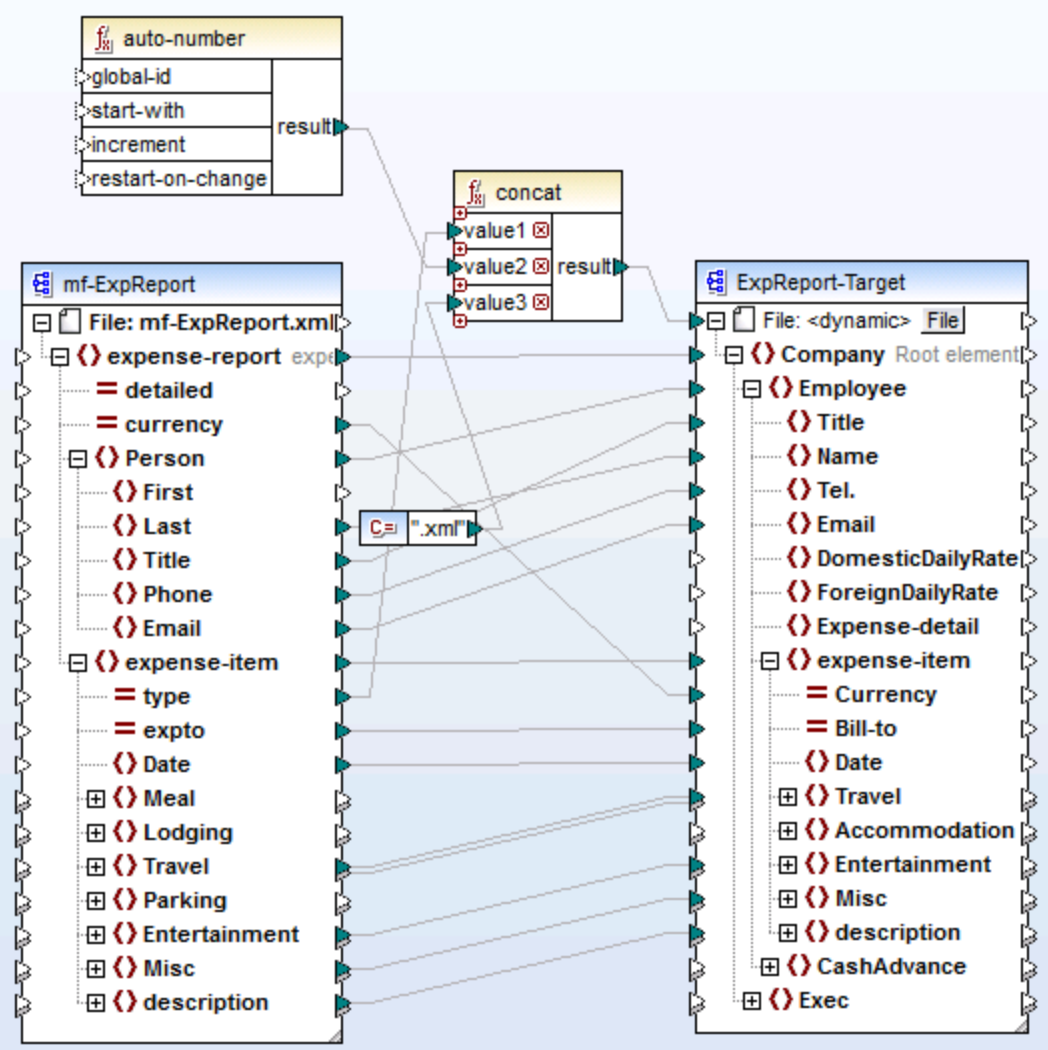
**expense-item** (5)

| | 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 :

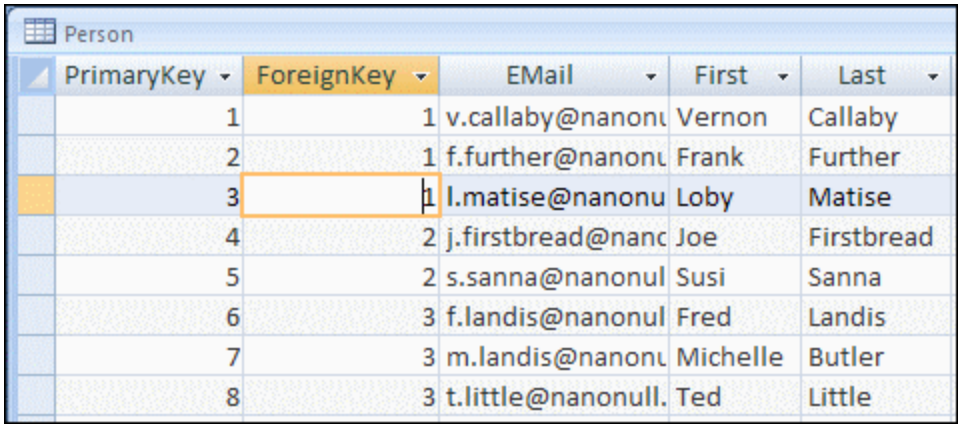
- L'attribut `type` fournit la première partie du nom de fichier (par exemple, "Travel").
- La fonction **`auto-number`** fournit le numéro séquentiel du fichier (par exemple, "Travel1", "Travel2", etc.).
- La constante fournit l'extension de fichier, qui est ".xml", donc "Travel1.xml" est le nom de fichier du premier fichier.

## 7.2.2 Exemple : Séparer une table de base de données dans plusieurs fichiers XML

Cet exemple vous montre comment générer dynamiquement plusieurs fichiers XML, un pour chaque enregistrement d'une table de base de données. Le mappage d'accompagnement pour cet exemple est

disponible sous le chemin suivant :
**<Documents>\Altova\MapForce2024\MapForceExamples\Tutorial\PersonDB-dyn.mfd**.

Le fichier de base de données de source (disponible dans le même dossier que le mappage) inclut une table Person qui contient 21 enregistrements. L'objectif de cet exemple est de générer un fichier XML séparé pour chaque enregistrement dans la table Person.

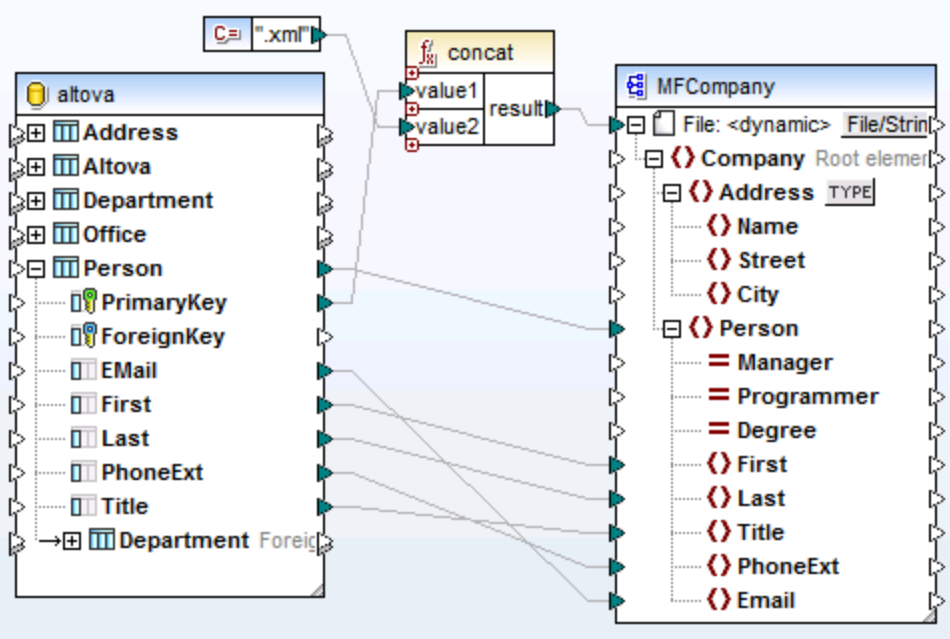


Person

| PrimaryKey | ForeignKey | EMail | First | Last |
|---|---|---|---|---|
| 1 | 1 | v.callaby@nanonu | Vernon | Callaby |
| 2 | 1 | f.further@nanonu | Frank | Further |
| 3 | 1 | l.matise@nanonu | Loby | Matise |
| 4 | 2 | j.firstbread@nanc | Joe | Firstbread |
| 5 | 2 | s.sanna@nanonul | Susi | Sanna |
| 6 | 3 | f.landis@nanonul | Fred | Landis |
| 7 | 3 | m.landis@nanonu | Michelle | Butler |
| 8 | 3 | t.little@nanonull. | Ted | Little |

Étant donné que le champ "PrimaryKey" identifie uniquement chaque personne dans la table, il s'agit de l'item que nous allons utiliser pour séparer la base de données de source dans des fichiers séparés. 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. 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**.
4. Créer les connexions tel qu'affiché ci-dessous puis cliquer sur l'onglet **Sortie** pour voir le résultat du mappage.




*PersonDB-dyn.mfd (MapForce Professional Edition)*

Veuillez noter que les fichiers de sortie résultants sont nommés dynamiquement comme suit :

- Le champ **PrimaryKey** fournit la première partie du nom de fichier (par exemple, "1").
- La constante fournit l'extension de fichier (".xml"), donc "1.xml" est le nom de fichier du premier fichier.

## 7.3 Parser et sérialiser des strings

Le parsage et la sérialisation de string est une technique avancée de mappage qui vous permet de configurer le composant soit pour parser les données depuis un string, soit pour sérialiser les données vers un string. Cette technique peut être considérée comme une alternative à la lecture des données depuis des fichiers (ou d'écrire des données dans des fichiers). Les composants MapForce qui parsent les strings ou sérialisent des données dans des strings peuvent être utiles dans un grand nombre de situations, par exemple si :

- vous souhaitez insérer des structures comme des fichiers XML dans des champs de base de données.
- vous souhaitez convertir des fragments XML stockés dans des champs de base de données dans des fichiers XML autonomes.
- vous avez des données legacy stockées en tant que texte (par exemple, un contenu de longueur fixe dans un seul champ de base de données), et que vous souhaitez convertir ces données dans une structure entièrement triable et basée sur champs

Le parsage de string et la sérialisation est disponible pour les types de composant MapForce suivants :

- Texte (CSV, texte à champ de longueur fixe)
- Fichier de schéma XML

Le parsage de string et la sérialisation sont pris en charge pour les types de composant MapForce suivants :

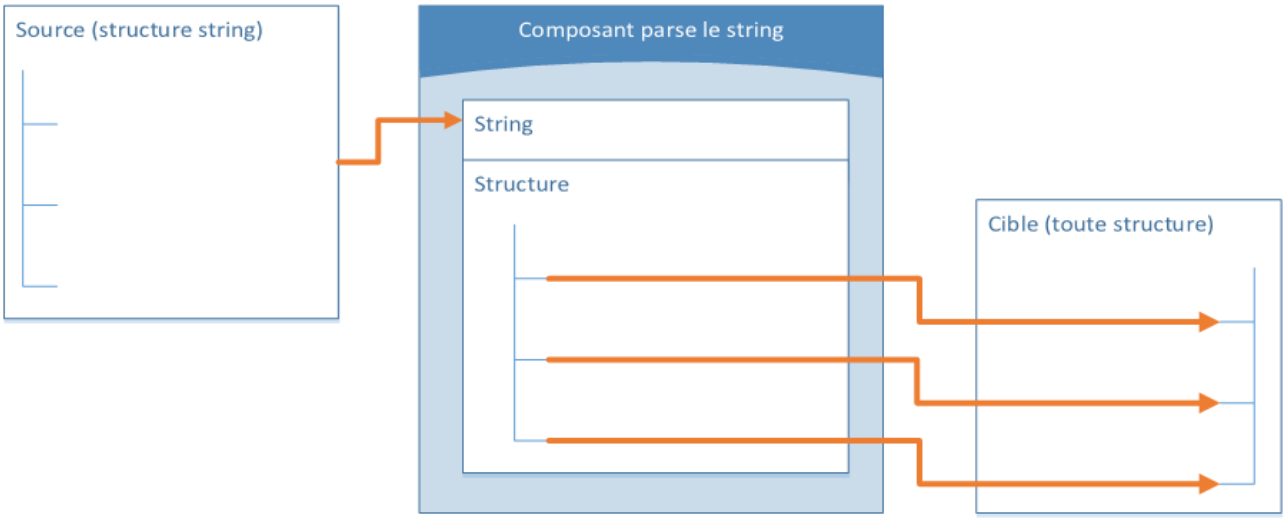
- Texte (CSV, champ de saisie de longueur fixe)
- Fichiers de schéma XML

Pour tous les types de composants ci-dessus, le parsage de string et la sérialisation sont pris en charge dans le langage cible BUILT-IN. De plus, le parsage de strings dans JSON ou la sérialisation de JSON depuis des strings est prise en charge dans BUILT-IN, C# et Java.

### 7.3.1 À propos du composant Parser/Sérialiser

Un composant Parser/Sérialiser dans MapForce est un composant hybride qui n'est ni un composant source ni un composant cible. Selon le rôle qu'ils jouent dans la conception de mappage, ce genre de composants doivent être placés entre d'autres composants source et cible.

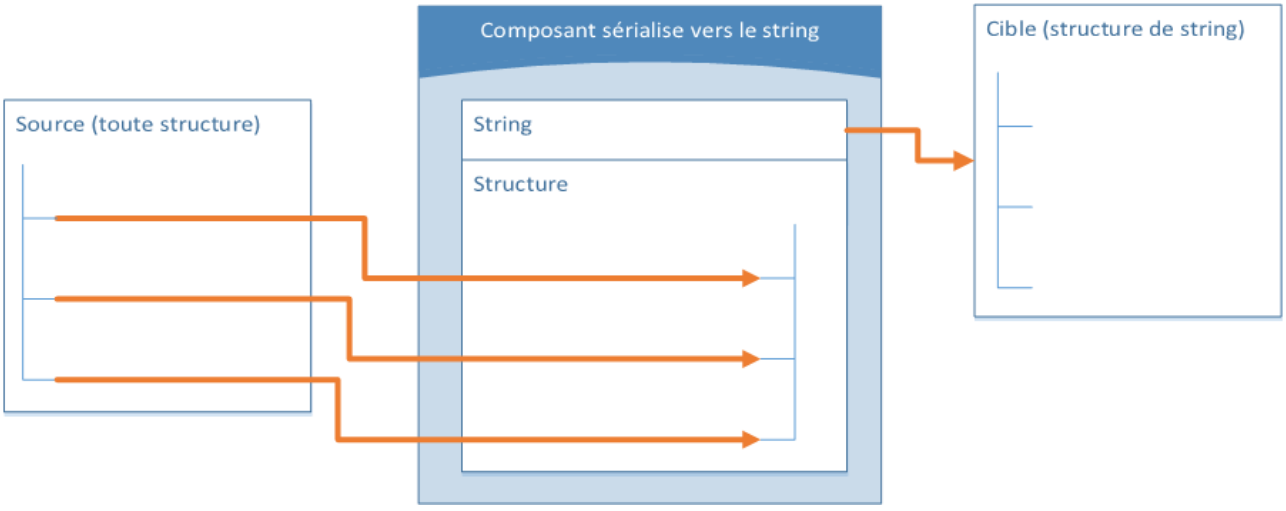
Vous pouvez utiliser le composant "Parser/Sérialiser String" pour le parsage du string lorsque, pour une raison ou une autre, vous devez convertir un string qui a une structure (par exemple, quelque XML stocké en tant que string dans une base de données) en un format différent. Le parsage de données depuis le string de source dans le composant "Parser/Sérialiser" signifie que le string de source est transformé en une structure MapForce ; ainsi, vous obtenez l'accès à tout élément ou attribut du XML de source stocké en tant que string.




*Composant générique "Parser String"*

Le diagramme ci-dessus illustre la structure typique d'un composant MapForce qui parse un string. Veuillez noter que le composant "Parser/Sérialiser String" est placé entre la source et la cible du mappage. Le fonctionnement de ce composant est d'accepter certaines structures de string en tant qu'entrée, par le biais d'un seul connecteur MapForce connecté à son nœud **String** supérieur. La structure de sortie peut être une des cibles de données prises en charge par MapForce.

Lorsque vous sérialisez des données depuis un composant vers un string, c'est l'inverse qui se produit. L'ensemble de la structure du composant MapForce devient une structure de string que vous pouvez manipuler ultérieurement, si vous le souhaitez. Par exemple, ceci permet d'écrire un fichier XML (ou fragment XML) dans un champ de base de données ou dans une seule cellule d'une feuilles de calcul Excel.

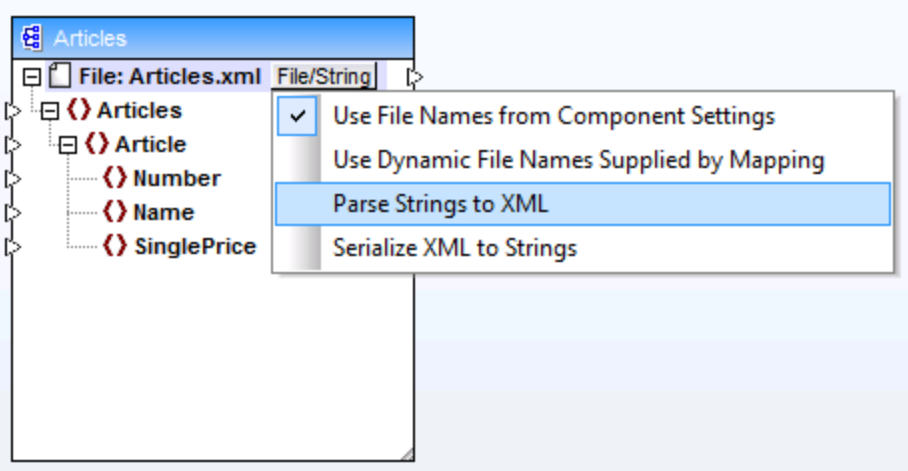



*Composant générique "Sérialiser vers String"*

Le diagramme ci-dessus illustre un composant générique MapForce "Sérialiser vers String". Ce composant en question accepte toute source de données comme entrée prise en charge par MapForce (par le biais de

connecteurs MapForce standard). La structure de sortie est un string que vous pouvez passer par la suite par le biais d'un connecteur MapForce unique appartenant au nœud **String** supérieur du composant à l'item de composant cible (par exemple, une cellule de feuille de calcul). Pour consulter un exemple, voir Exemple : Sérialiser vers un String (XML à base de données) 797.

Vous pouvez désigner un composant pour le parsage ou la sérialisation de string à tout moment depuis la fenêtre de mappage. Pour ce faire, cliquez sur la touche **Fichier/String** ( File/String ) adjacente au nœud de racine, puis sélectionnez l'option désirée.




*Modifier le mode de composant*

**Note :** un composant "Parser/Sérialiser String" ne peut pas lire des données depuis un string et l'écrire dans un string simultanément. C'est pourquoi, le nœud de racine peut soit avoir un connecteur entrant, soit un connecteur sortant (pas les deux). Une erreur sera générée si vous tentez d'utiliser le même composant pour les deux opérations.

Lorsque vous désignez un composant pour le parsage ou la sérialisation de string, l'apparence du composant change comme suit :

- Le composant obtient le préfixe **parse** ou **serialize** dans le titre.
- La barre de titre présente une couleur d'arrière-plan jaune, semblable aux composants de fonction.
- Le nœud supérieur débute avec le préfixe **String:** et est identifié par l'icône .
- Si le composant parse un string, le connecteur de sortie depuis le nœud de racine n'est pas significatif et n'est donc pas disponible.
- Si le composant est sérialisé vers un string, le connecteur d'entrée dans le nœud de racine n'est pas significatif et n'est donc pas disponible.

Lorsqu'un composant se trouve dans le mode "Parser/Sérialiser String", vous pouvez modifier ses paramètres comme s'il se trouvait en mode basé sur fichier (voir Changer les paramètres du composants 44). Veuillez noter que tous les paramètres de composant ne sont pas disponibles lorsqu'un composant se trouve soit en mode "Parser" ou "Sérialiser".

### 7.3.2 Exemple : sérialiser vers un string (XML vers base de données)

Cet exemple vous guide dans les étapes nécessaires pour créer un design de mappage qui sérialise les données dans un string. L'exemple est accompagné d'un fichier d'échantillon. Si vous souhaitez consulter le fichier échantillon avant de vous lancer dans cet exemple, vous pouvez l'ouvrir depuis le chemin suivant : **<Documents>\Altova\MapForce2024\MapForceExamples\Tutorial\SerializeToString.mfd**.

Partons d'un exemple dans lequel vous avez un fichier XML (et son schéma lié) qui consiste en plusieurs éléments `<Person>`. Chaque élément `<Person>` réunit le prénom d'une personne, le nom de famille, le titre professionnel, l'extension téléphonique et l'adresse e-mail, comme suit :

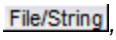
```
<Person>
      <First>Joe</First>
      <Last>Firstbread</Last>
      <Title>Marketing Manager Europe</Title>
      <PhoneExt>621</PhoneExt>
      <Email>j.firstbread@nanonull.com</Email>
</Person>
```

Votre objectif est d'extraire chaque élément `<Person>` depuis le fichier XML et de l'insérer littéralement (y compris les balises XML) en tant que nouvel enregistrement de base de données dans la table `PEOPLE` d'une base de données SQLite. La table `PEOPLE` contient uniquement deux colonnes : `ID` et `PERSON`. Sa définition complète est la suivante :

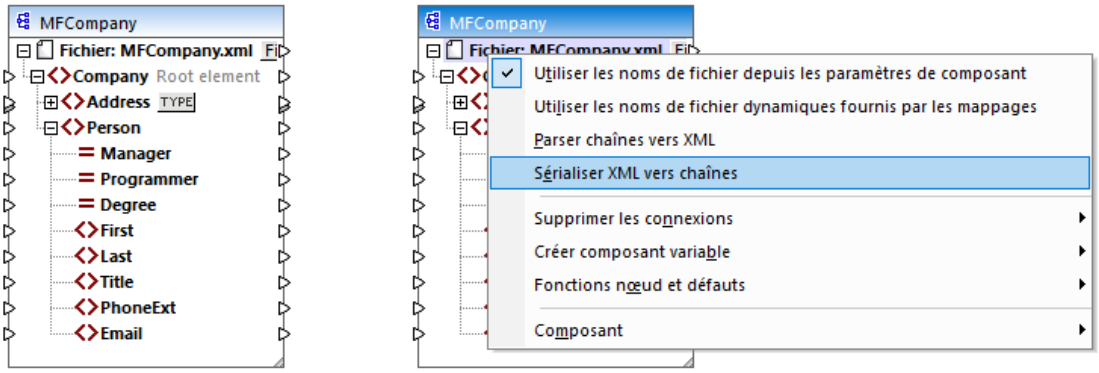
```
CREATE TABLE PEOPLE (ID INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL, PERSON TEXT);
```

Une fois que le mappage a été exécuté, le résultat attendu est que la table `PEOPLE` aura le même nombre de lignes que celui des éléments <Person> dans le fichier XML.

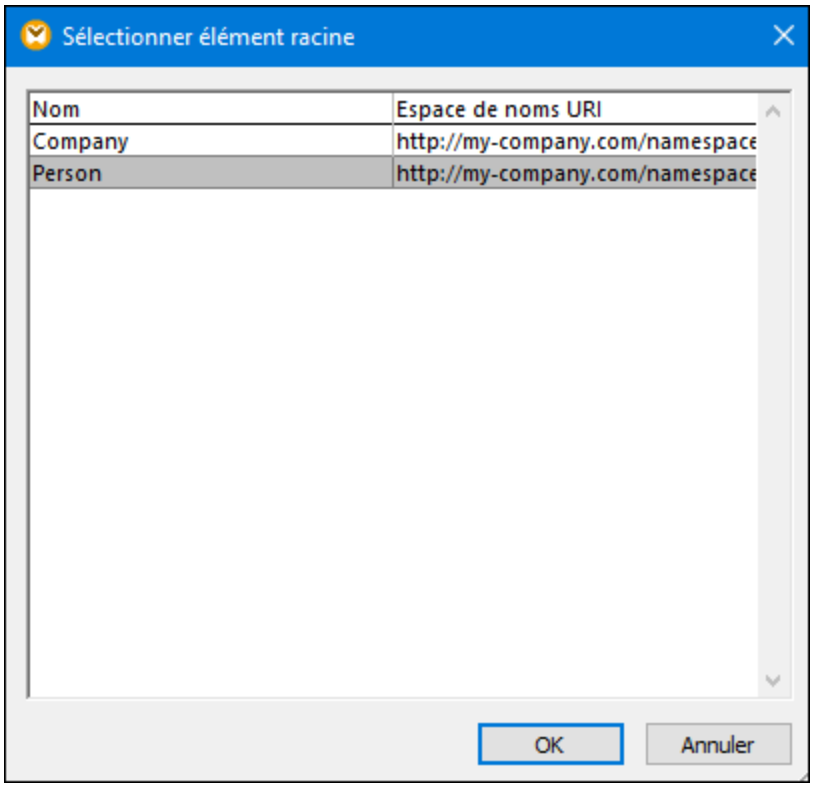
Pour atteindre cet objectif, procéder comme suit :

1. Ajouter à la zone de mappage le composant XML de source (utiliser la commande de menu **Insérer | Schéma XML/Fichier**). Le fichier échantillon est disponible sous : **<Documents>\Altova\MapForce2024\MapForceExamples\Tutorial\MFCompany.xml**.
2. Dupliquer (copier-coller) le composant XML.
3. Sur le composant XML dupliqué, cliquer sur 

, puis sélectionner **Sérialiser XML sur Strings**.

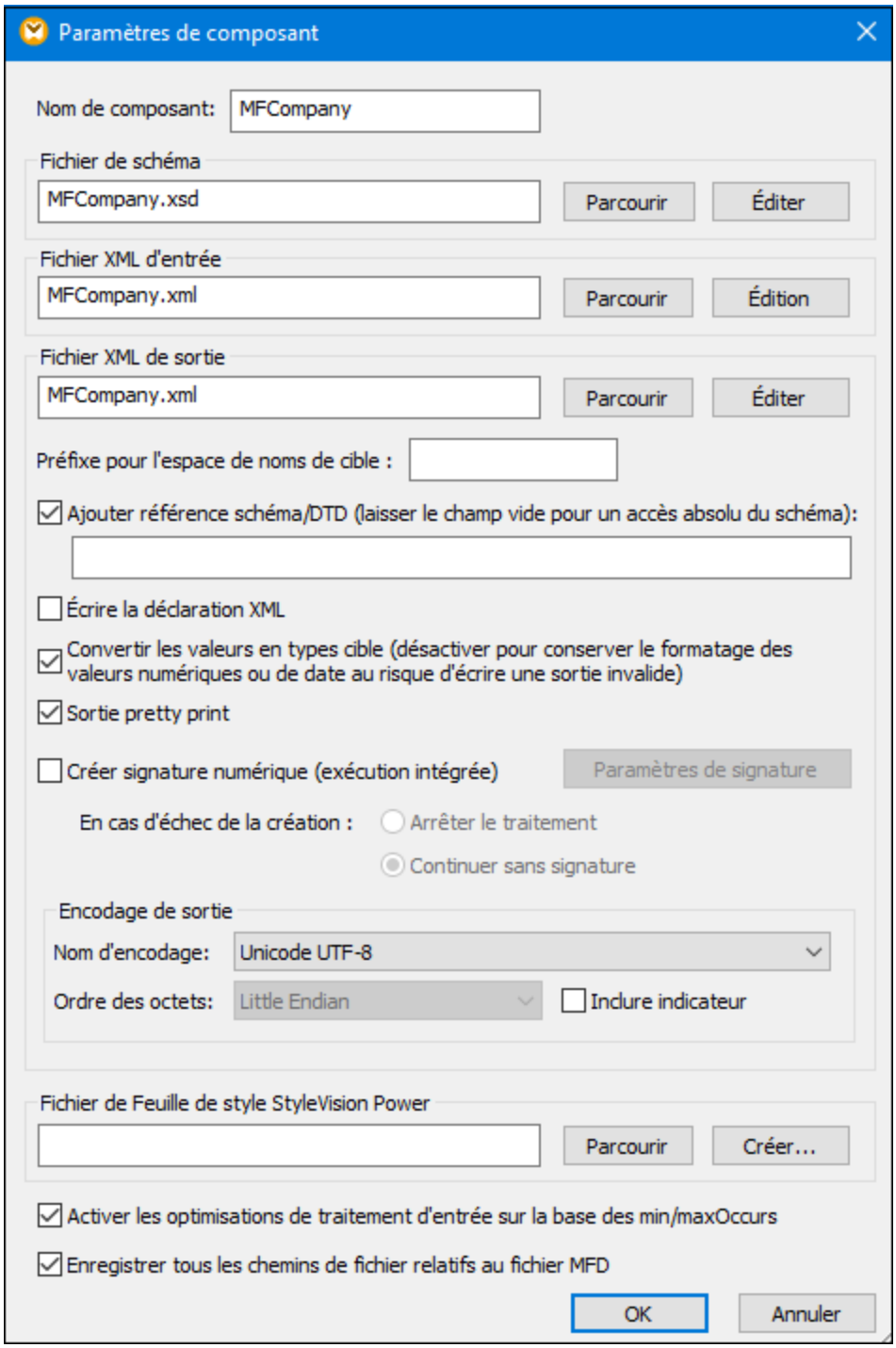



4. Cliquer avec la touche de droite sur le composant dupliqué et choisir **Changer l'élément racine** depuis le menu contextuel. Puis changer l'élément de racine sur `<Person>`.

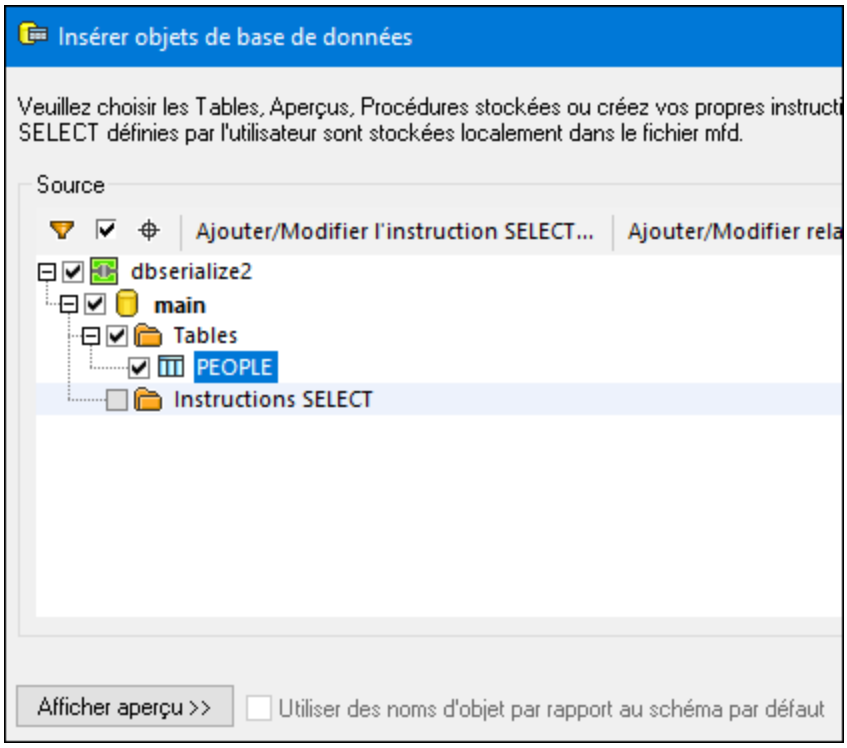



> En général, vous pouvez changer l'élément de racine à tout élément qui a une déclaration globale (pas locale) dans le schéma XML. Tout élément qui n'est pas défini globalement dans votre schéma n'est pas listé dans le dialogue "Sélectionner l'élément racine".

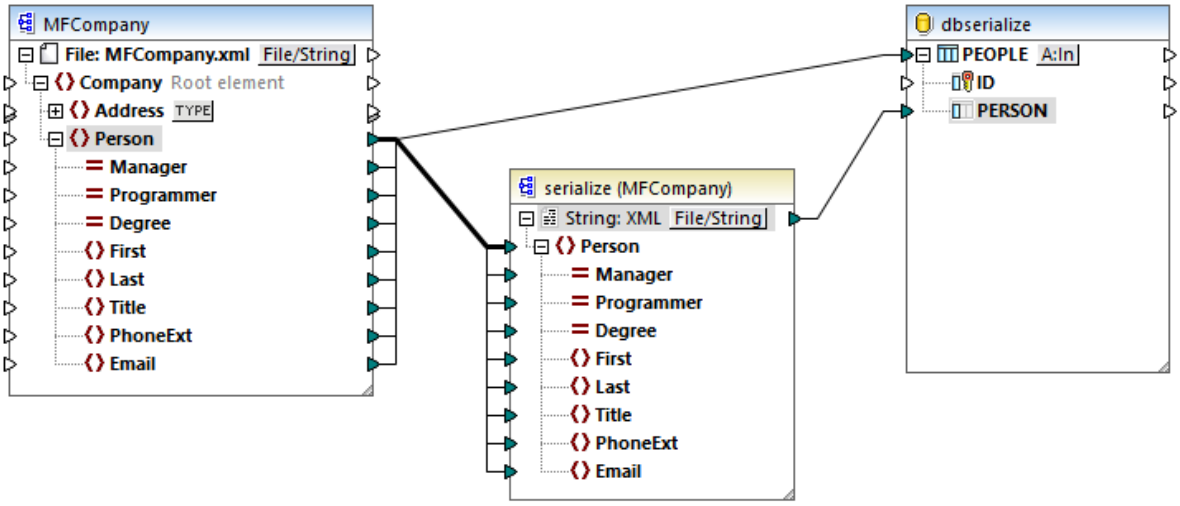
5. Double-cliquer le composant et décocher la case **Écrire déclaration XML**. Cela évite l'écriture de la déclaration XML pour chaque élément `<Person>`.




6. Ajouter à la zone de mappage le composant de base de données SQLite cible, depuis le champ suivant : **<Documents>\Altova\MapForce2024\MapForceExamples\Tutorial\\dbserialize.db**. (Pour ajouter le composant de base de données, utiliser la commande de menu **Insérer | Base de données**, voir aussi Connexion à une base de données[161]). Lorsque vous serez invité à insérer un objet de base de données, choisir la table PEOPLE.



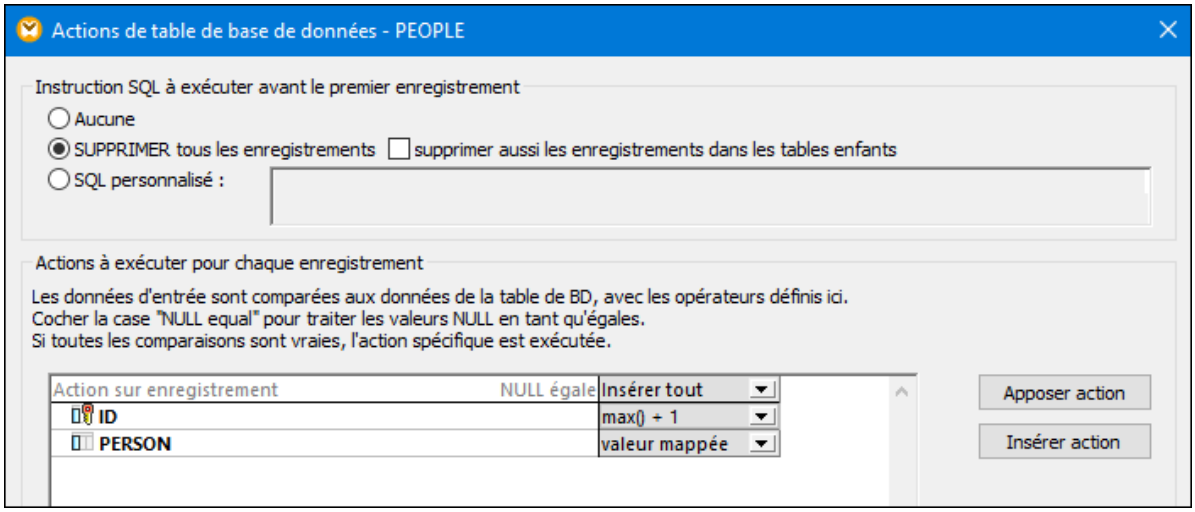

7. Lier les composants comme indiqué ci-dessous. À gauche du mappage, l'élément `<Person>` mappe dans un composant de sérialisation. À droite du mappage, la valeur de string sérialisée est insérée dans la colonne `PERSON` de la table de la base de données `PEOPLE`. Enfin, le connecteur dessiné depuis `<Person>` vers la table `PEOPLE` indique à MapForce de créer un nouvel enregistrement pour chaque élément `<Person>` rencontré.




8. Cliquer sur la touche **A:In** dans le composant de base de données, et procéder comme suit :
   a. Sélectionner l'option **Supprimer tous les enregistrements**. Au moment du mappage, tous les enregistrements existants seront supprimés de la base de données avant d'en insérer de nouveaux
   b. Sélectionner l'option **généré en BD** à côté de la colonne **ID**. Cela garantit que l'ID de l'enregistrement sera généré par la base de données. Noter que l'option **généré en BD** apparaît

aussi si la colonne prend en charge cette option. Pour les colonnes qui ne sont pas une identité ou un champ auto-incrémenté, l'option **max+1** est disponible à la place, cette option analysera quelle est la valeur ID maximum existant dans la base de données et d'insérer l'entier disponible suivant, incrémenté de 1.




Vous venez de créer un design de mappage qui sérialise les données en string. Si vous cliquez sur l'onglet **Sortie**, la requête SQL d'aperçu indique que des enregistrements séparés seront insérés dans la base de données pour chaque élément `<Person>` dans le fichier XML, ce qui était l'objectif de ce mappage.

# 7.4 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

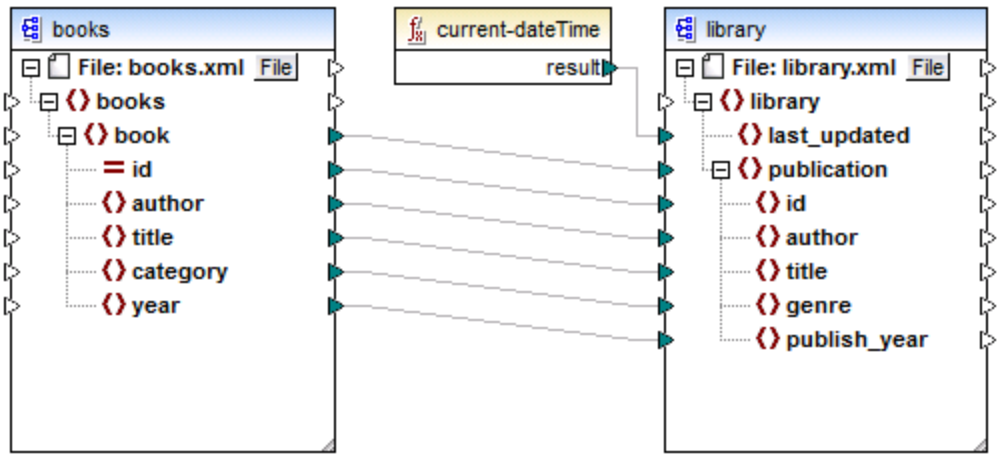
Pour être valide, un mappage doit inclure au moins une source et au moins un composant de cible. Un composant de source est un composant qui lit des données, généralement depuis un fichier ou une base de données. Un composant de cible est un composant qui écrit des données, généralement vers un fichier 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 79, au niveau du composant 44, ou même au niveau de la connexion 54, 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 803.
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 804.

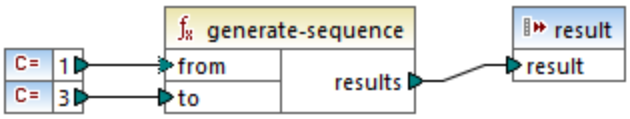
## 7.4.1 Séquences

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 124.

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. En alternative, vous pouvez obtenir le même résultat en utilisant Fonctions Défauts et Nœud 476.

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.

## 7.4.2 Le contexte de mappage

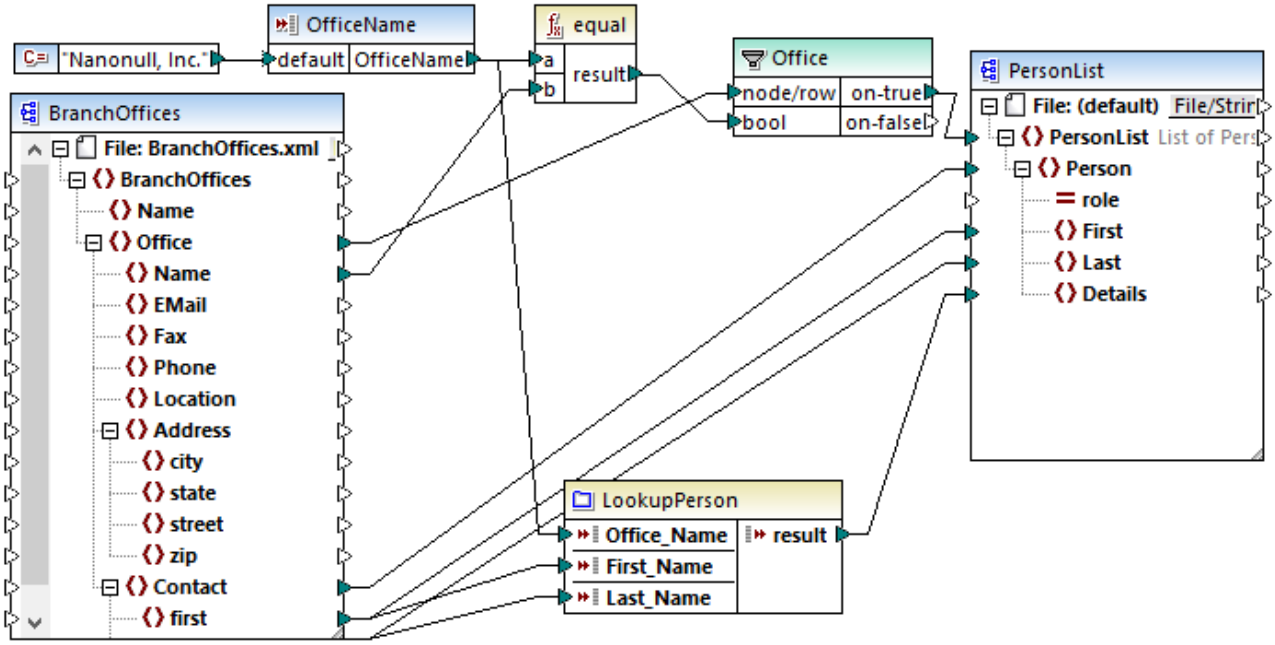
Les composants de mappage sont des structures hiérarchiques qui peuvent contenir plusieurs niveaux de profondeur. D'un autre côté, un mappage peut avoir plusieurs sources et composants, plus des composants intermédiaires comme des fonctions, des filtres, des value-maps, etc. Cela rajoute de la complexité 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**.

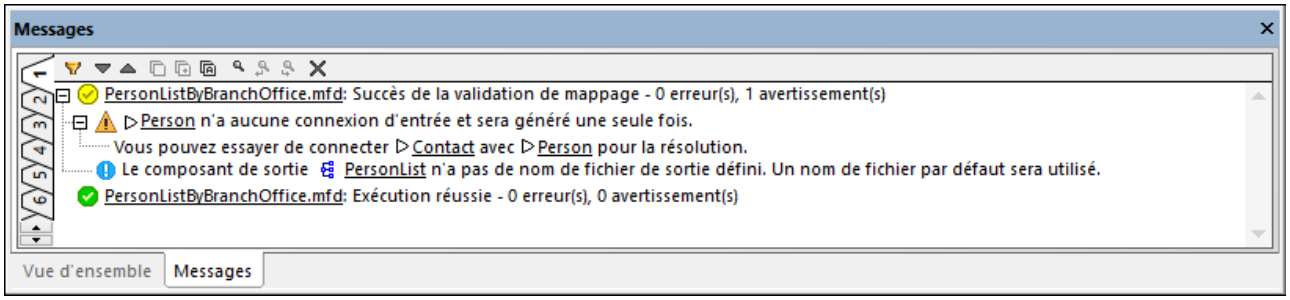
Comme mentionné précédemment, l'exécution de mappage commence toujours depuis le nœud de racine cible (**PersonList**, dans cet exemple). En retraçant la connexion (par le biais du filtre et de la fonction) vers un item de source, vous pouvez conclure que l'item de source est **Office**. (L'autre chemin 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 504.

### 7.4.2.1 Base de données

Afin d'améliorer l'efficacité et de réduire l'usage des ressources en matériel et en réseau, il est recommandé d'éviter d'appeler la même base de données plusieurs fois dans le même mappage. Vous pouvez être confronté à des situations dans lesquelles vous ne pouvez pas éviter d'appeler une base de données plusieurs fois en raison de la nature du mappage, mais voici quelques recommandations :

- Si vous ne nécessitez qu'un seul appel de base de données, évitez de placer le composant de base de données dans un contexte parent qui exigerait d'appeler la base de données plusieurs fois. Cela pourrait se produire par exemple, si vous ajoutez un composant de base de données dans une fonction définie par l'utilisateur qui reçoit une séquence de valeurs en tant qu'entrée et qui reçoit donc un appel pour chaque item dans la séquence, voir aussi Fonctions définies par l'utilisateur 807 ci-dessous. Des variables sont généralement utiles pour rassembler des données dans le même contexte avant le les passer dans le composant cible.
- Si vous souhaitez rassembler des valeurs depuis une base de données (par exemple, pour compter le nombre d'enregistrements en utilisant la fonction **count**), il est recommandé de connecter la sortie de la fonction aggregate dans une variable où **compute-when=once**. Cela évite les appels répétitifs à la base de données, comme décrit dans l'Exemple: Compter les lignes de table de base de données 421.
- Essayer d'extraire toutes les données de base de données en un appel (par exemple, une instruction SQL-SELECT, ou une procédure stockée), contrairement à l'ajout du même composant de base de données plusieurs fois dans le mappage.
- Si vous souhaitez extraire des données depuis plusieurs tables ou modes depuis la même base de données, nous vous conseillons d'utiliser soit un composant Join (dans le mode SQL), soit une instruction SQL-SELECT. Ce dernier est plus pratique si vous préférez écrire l'instruction SQL SELECT vous-même. Si vous devez joindre des données de base de données à des données non-base de données, ou des données provenant de base de données différentes, utiliser des joins non-SQL. Pour optimiser l'exécution de joins non-SQL dans des mappages intenses en données, exécuter les mappages avec MapForce Server Advanced Edition.

- Si vous nécessitez de filtrer des données depuis une base de données, il est plus efficace d'utiliser un composant SQL-WHERE au lieu d'un filtre standard, puisque le composant précédent a été optimisé spécifiquement pour le travail avec des bases de données, dans la grammaire de la base de données respective.

### 7.4.2.2 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.

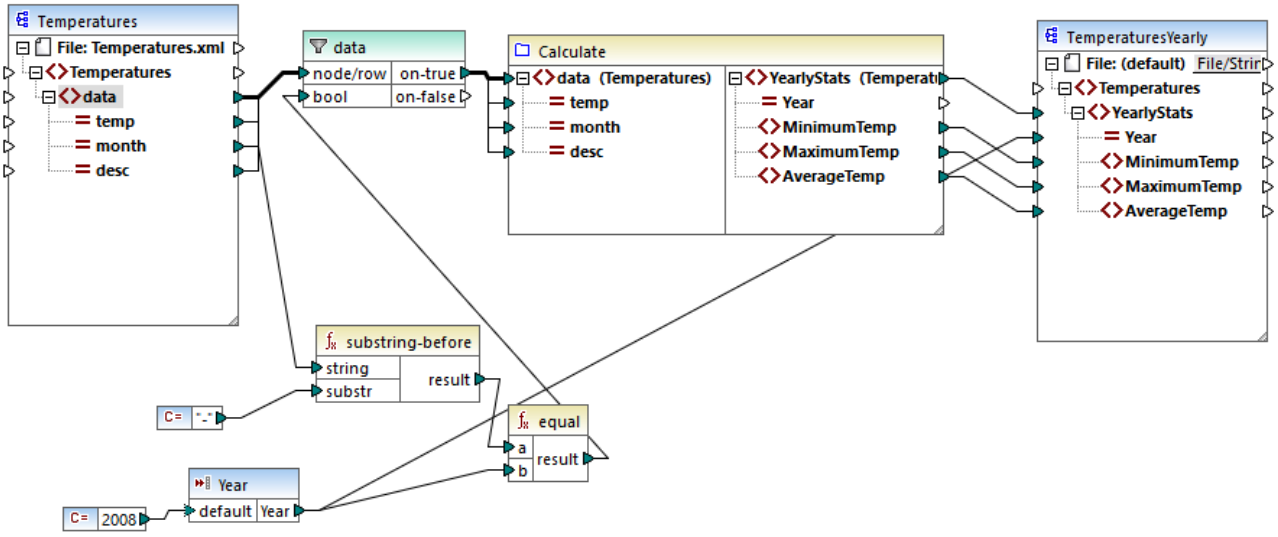
Ce comportement mentionné peut être acceptable pour le type de mappage dans lequel vous avez vraiment besoin que la FDU soit appelée *autant de fois qu'il y a de valeurs d'entrée* et que vous n'avez pas d'alternative.

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 491.) 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**.

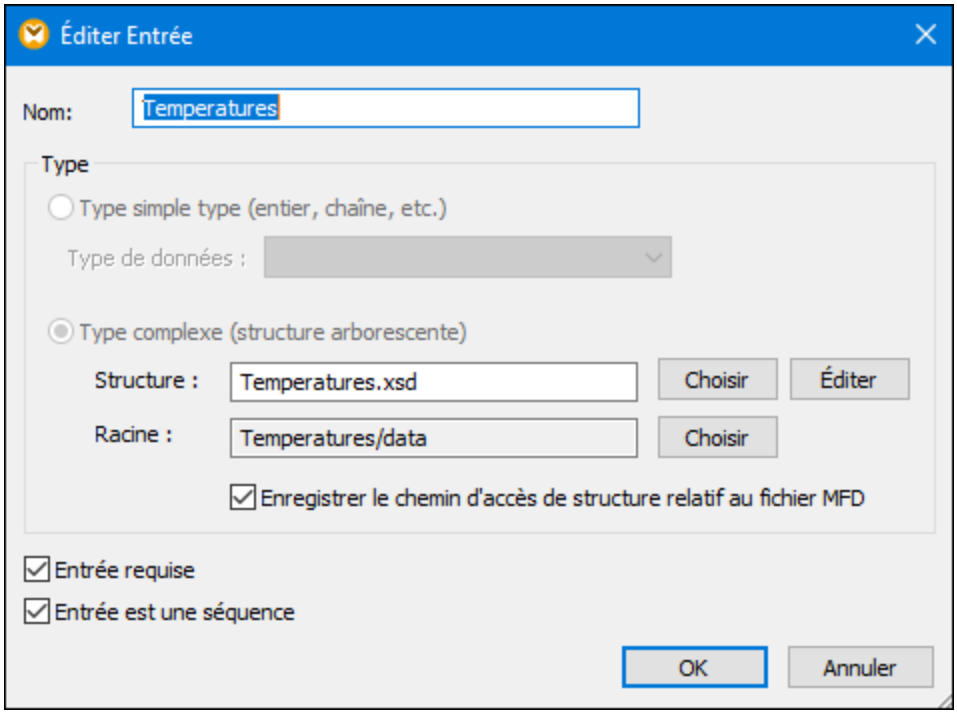



Le mappage ci-dessus illustre un cas typique d'une FDU qui fonctionne sur un ensemble de valeurs et nécessite donc toutes les valeurs d'entrée en un appel. Spécifiquement, la fonction définie par l'utilisateur **`Calculate`** retourne les températures minimum, maximum et moyennes, en prenant en tant 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>
```

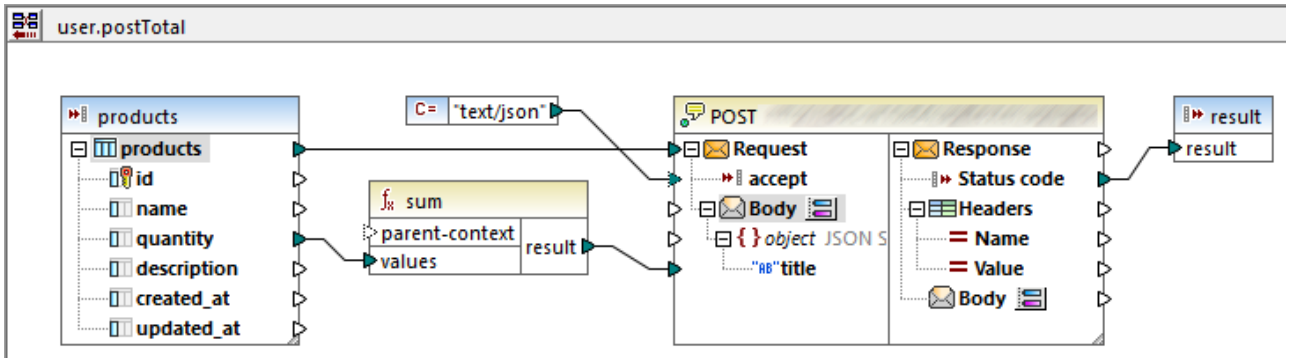
Comme d'habitude, l'exécution de mappage commence avec l'item supérieur du composant cible (**YearlyStats**, dans cette exemple). Pour remplir ce nœud, le mappage tente d'obtenir des données de source provenant de la FDU, qui, en revanche, déclenche le filtre. Le rôle du filtre dans ce mappage est de 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.

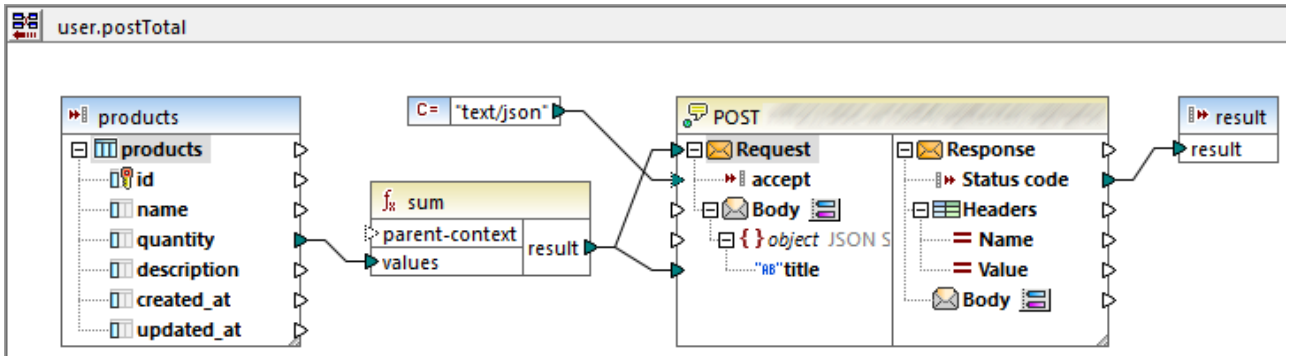
En appliquant la même logique dans des FDU plus complexes qui comprennent une base de données ou des appels de service Web, il peut être possible d'optimiser l'exécution et d'éviter des appels inutiles à la base de données ou au service Web. Néanmoins, sachez que la case à cocher **Entrée est séquence** 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 :



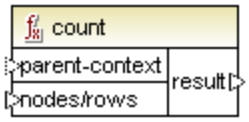

Normalement, les fonctions aggregate comme **sum**, **count**, etc produisent une seule valeur. Néanmoins, s'il existe une connexion parent qui le permet, elles peuvent également produire une séquence de valeurs, comme décrit plus loin dans l'Exemple : Changer le contexte Parent 810.

## 7.4.2.3 Exemple : Changer le contexte de parent

Certains composants de mappage ont un item optionnel **parent-context**.

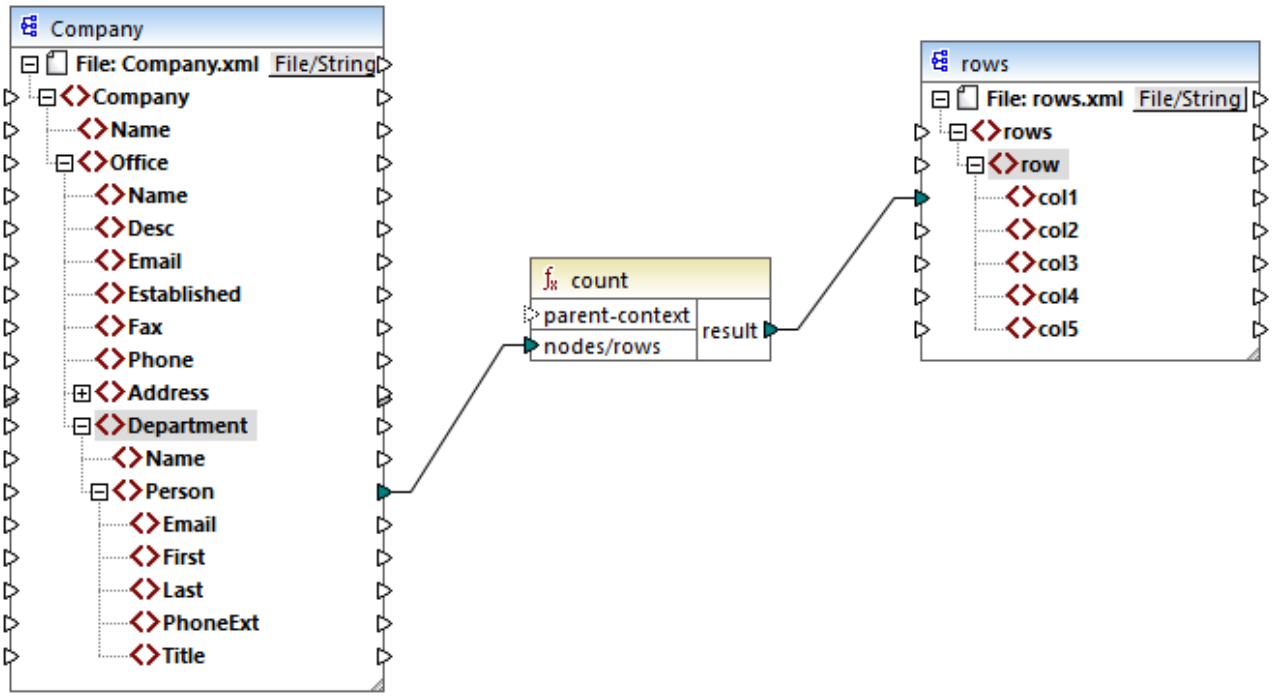
`parent-context` est un argument optionnel dans certaines fonctions d'agrégation core MapForce (comme dans **min**, **max**, **avg**, **count**). Dans un composant de source qui possède plusieurs séquences hiérarchiques, le contexte parent détermine l'ensemble de nœuds dans lequel la fonction doit fonctionner.

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 :

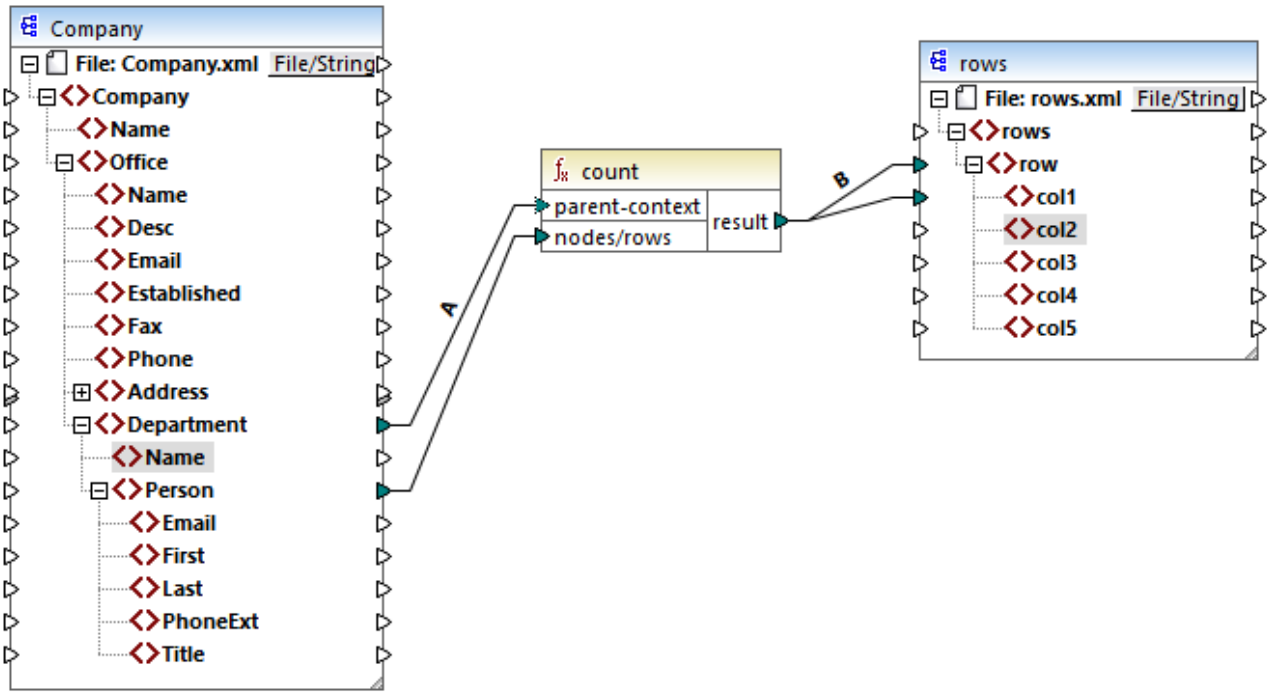
| Bureau | Département | Nombre d'employés |
|---|---|---|
| Nanonull, Inc. | Administration | 3 |
| | Marketing | 2 |
| | Engineering | 6 |
| | IT & Technical Support | 4 |
| Nanonull Partners, Inc. | Administration | 2 |
| | Marketing | 1 |
| | IT & Technical Support | 3 |

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 :

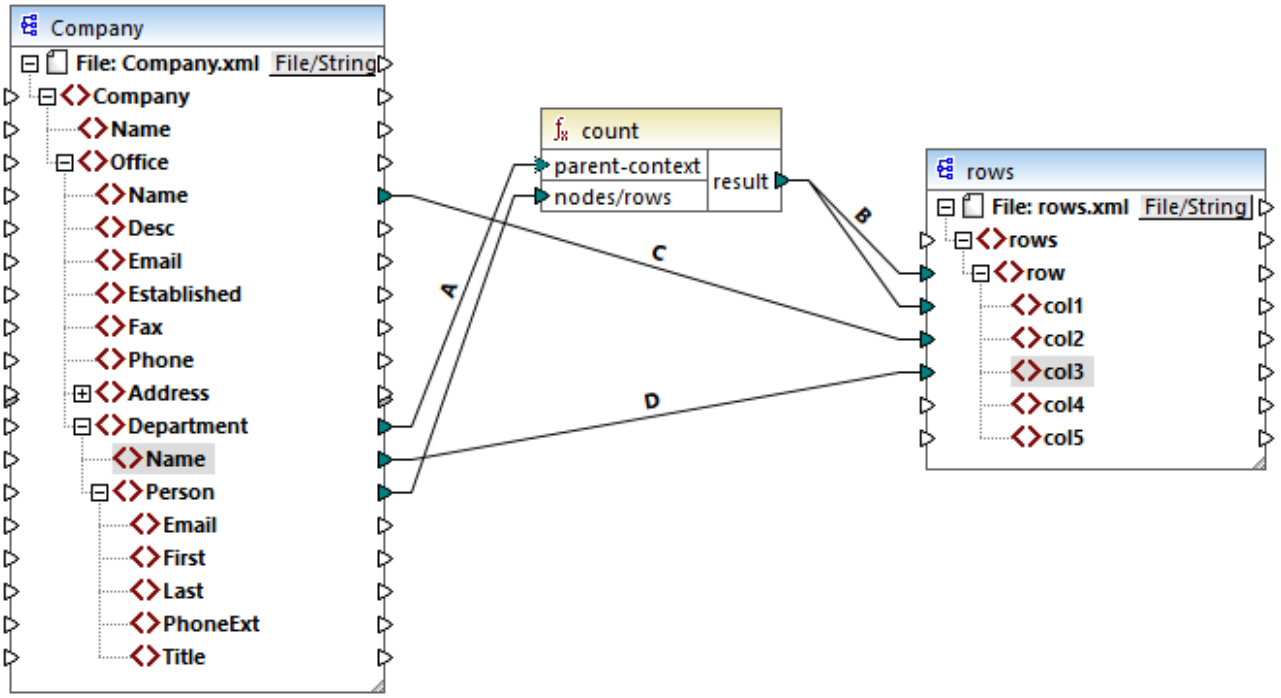



Dans le mappage ci-dessus, la connexion A change le contexte de parent de la fonction **count** dans le **Department**. En résultat, la fonction comptera le nombre de personnes se trouvant dans chaque département. Chose importante, la fonction retournera maintenant une *séquence* de résultats au lieu 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>
```

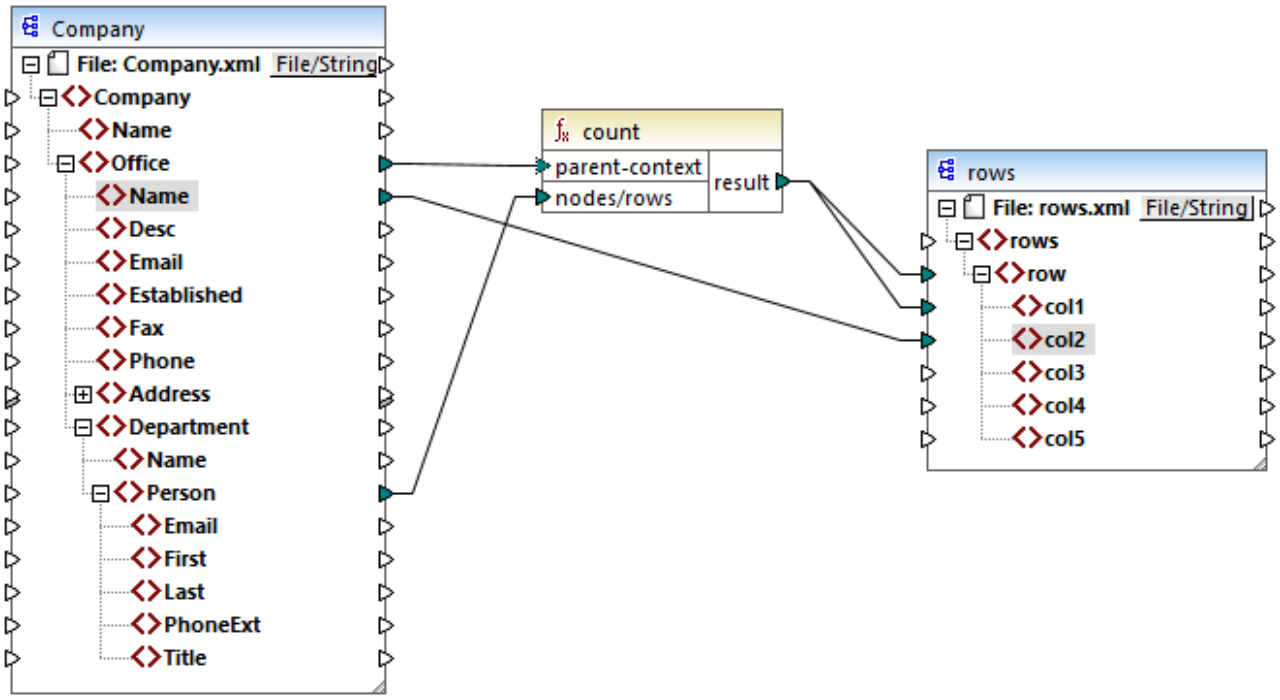
```
    </row>
    <row>
        <col1>6</col1>
    </row>
    <row>
        <col1>4</col1>
    </row>
    <row>
        <col1>2</col1>
    </row>
    <row>
        <col1>1</col1>
    </row>
    <row>
        <col1>3</col1>
    </row>
</rows>
```

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>
```

## 7.4.3 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

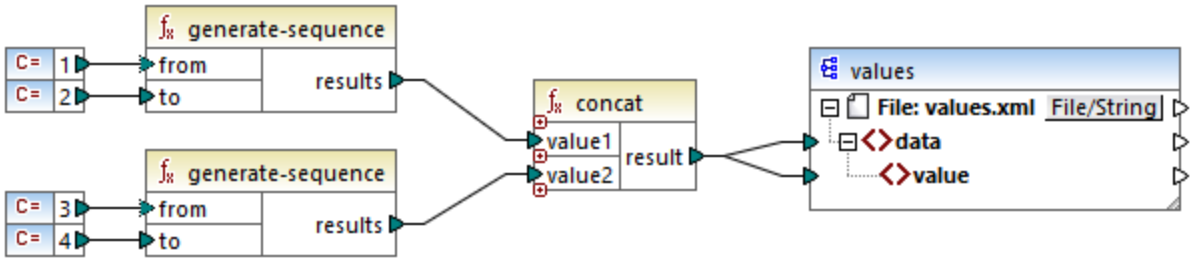
paramètres : *a* et *b*. si *a* et *b* obtiennent tous deux une séquence de valeurs, alors MapForce traite comme suit :

- 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".



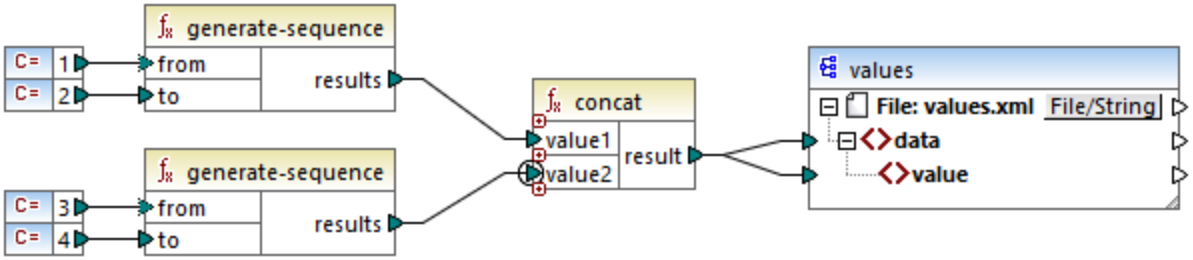

Tout d'abord, examinons le mappage sans déterminer de contexte de priorité. La fonction **concat** commence à évaluer la première séquence en premier, donc elle combine des valeurs dans l'ordre suivant :

- 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>
```

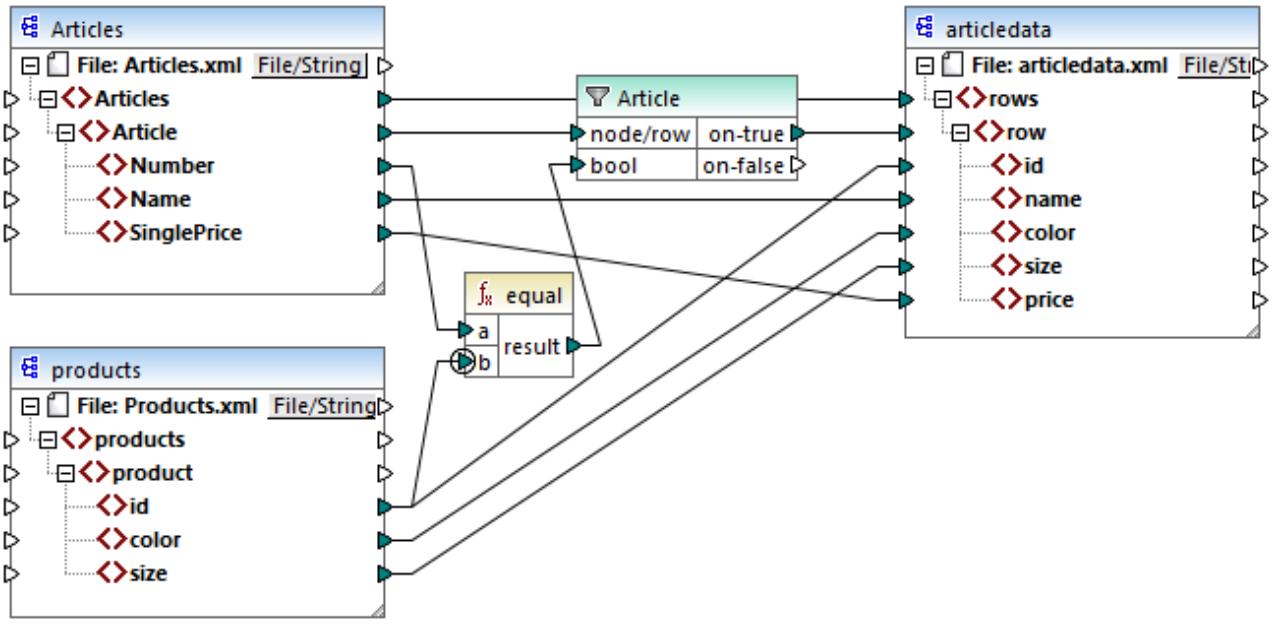
Jusqu'à présent, vous n'avez vu que la partie théorique derrière le contexte de priorité. Pour consulter un scénario plus pratique, voir Exemple: Filtrer avec un contexte de priorité 816.

## 7.4.3.1 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. Pour les bases de données, il est conseillé d'effectuer des filtres en utilisant des composants SQL WHERE 365 au lieu des filtres standard.




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.

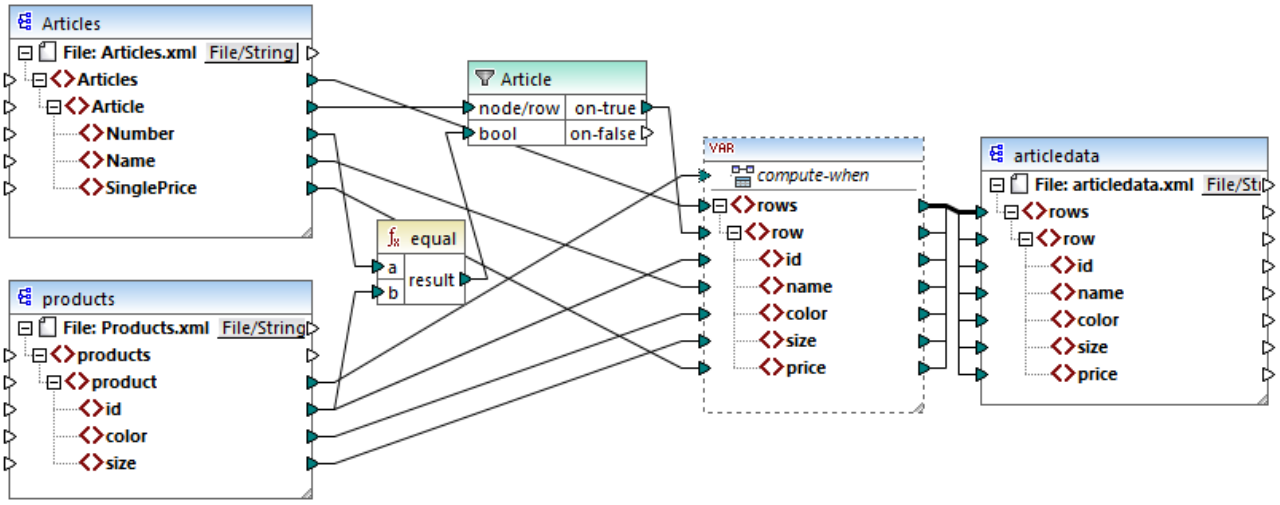
La logique de mappage ci-dessus produit le résultat correct, par exemple :

```
<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.

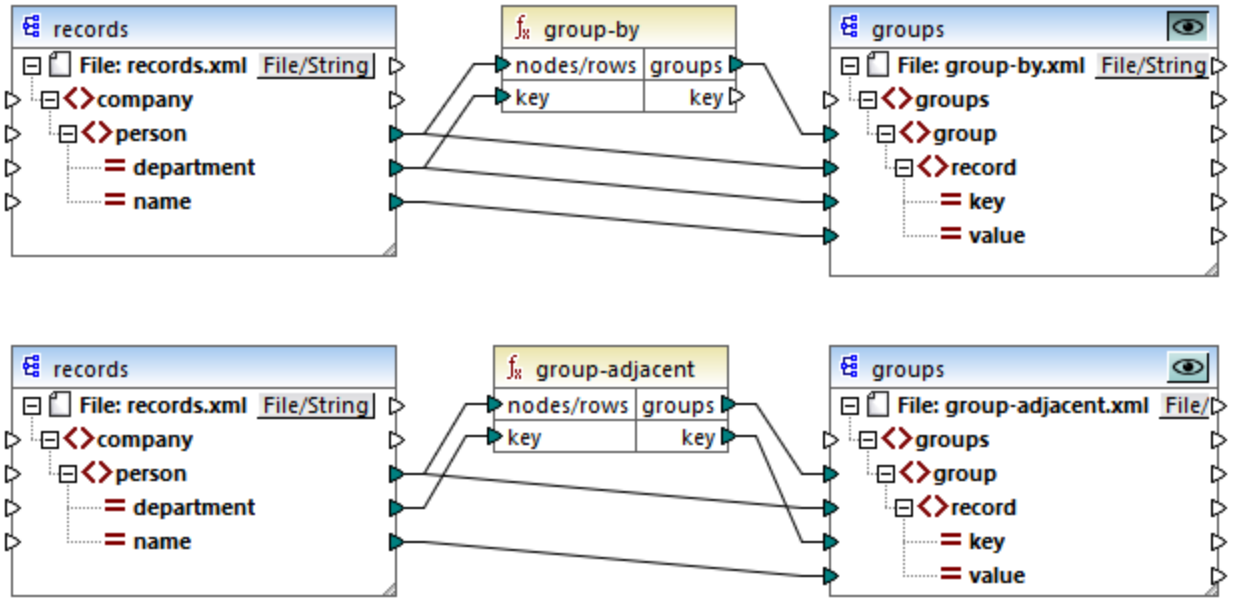
## 7.4.4 Composants de cible multiple

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.

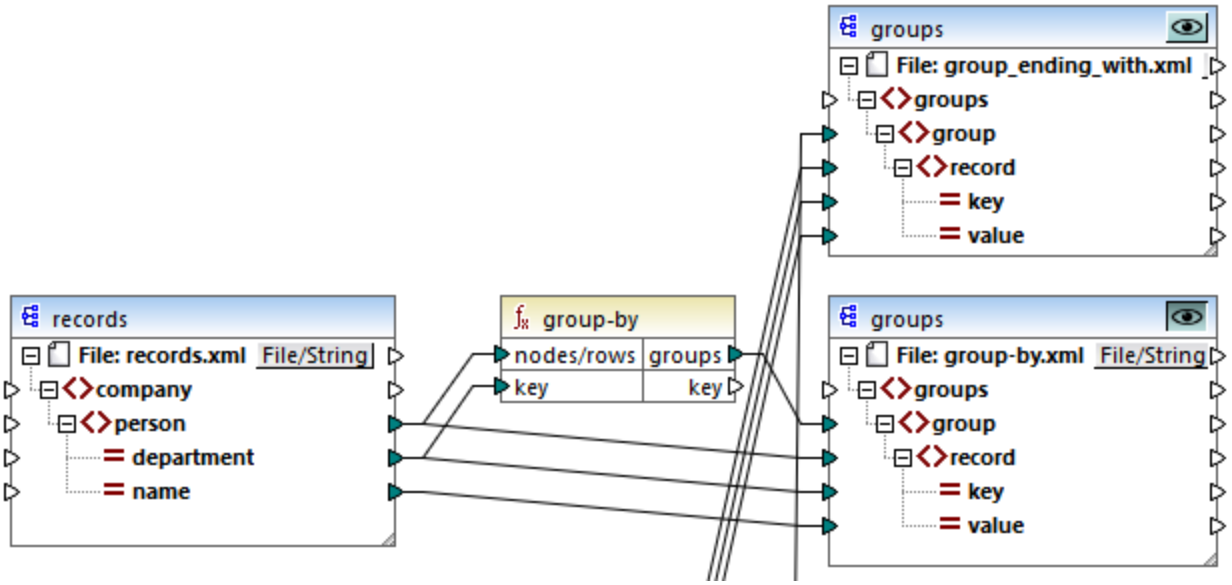
1. Dans le menu **Fichier**, cliquer sur **Générer du code dans | XSLT 2.0**.
2. Lorsque vous êtes invité à le faire, sélectionner un répertoire de cible pour le code généré.

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 Mapper la documentation

Vous pouvez générer une documentation détaillée concernant des mappages en format HTML, Microsoft Word (.doc), ou en format RTF. Si StyleVision doit être installé, vous pouvez générer une documentation en format PDF.

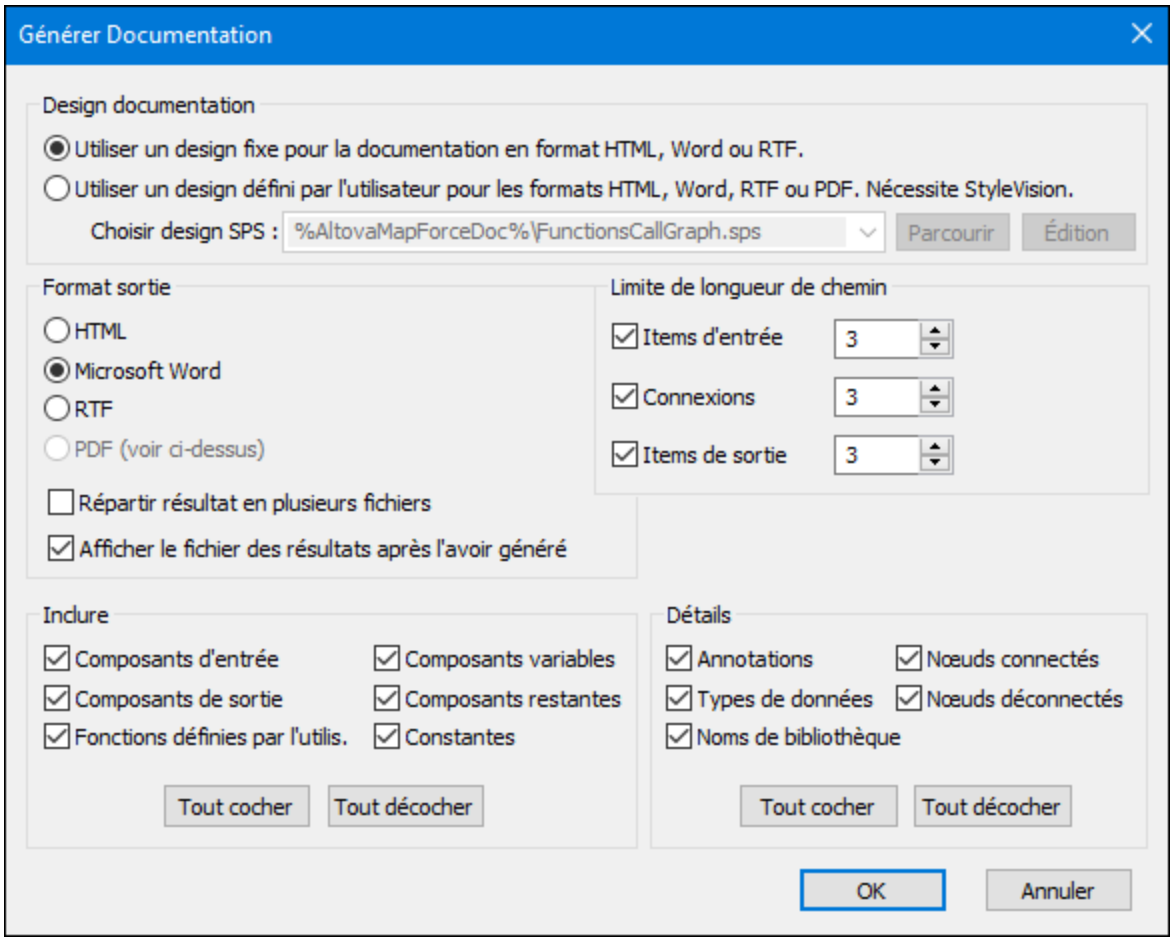
**Prérequis**

- Microsoft Word 2000 ou plus récent doit être installé si vous voulez générer une documentation sous le format Microsoft Word.
- StyleVision doit être installé si vous voulez générer une documentation en format PDF ou personnaliser le design de la documentation générée.

Par défaut, la documentation est générée avec un design fixé, Là où vous pouvez configurer les options de base telles que la profondeur des chemins affichés, et d'autres paramètres. Si StyleVision est installé, vous pouvez également profiter de plusieurs fichiers StyleVision Power Stylesheets (SPS) inclus ou même créer votre propre design dans StyleVision.

### Pour générer une documentation de mappage :

1. Dans le menu **Fichier**, cliquer sur **Générer documentation**. Cela ouvre le dialogue "Générer documentation".




2. Choisir les paramètres requis et cliquer sur **OK**.

Les paramètres que vous pouvez configurer sont décrits ci-dessous.

## Design de documentation

- Choisir « Utiliser design fixe... » pour utiliser le modèle de documentation intégré.
- Choisir « Utiliser design défini par l'utilisateur... » pour utiliser un StyleVision Power Stylesheet prédéfinie créée dans StyleVision. Les fichiers SPS sont disponibles dans le dossier **...\Documents\Altova\MapForce2024\Documentation\MapForce**. Pour plus de détails, voir Feuilles de style StyleVision Power 826 prédéfinies.
- Cliquer sur **Parcourir** pour chercher un fichier SPS prédéfini.
- Cliquer sur **Éditer** pour lancer StyleVision et ouvrir la SPS sélectionnée dans une fenêtre StyleVision.

## Format de sortie

- Sélectionnez un des formats de sortie suivants : HTML, Microsoft Word, RTF ou PDF. Les documents Microsoft Word sont créés avec l'extension de fichier **.doc** lorsqu'ils sont générés à l'aide d'un design fixe, et avec une extension de fichier **.docx** lorsqu'ils sont générés avec une SPS de StyleVision. Le format de sortie PDF exige l'utilisation de StyleVision et est uniquement disponible si vous avez choisi une SPS StyleVision.

- Choisir **Répartir la sortie sur plusieurs fichiers** si vous souhaitez générer plusieurs fichiers de documentation, un fichier pour chaque composant individuel comme le composant d'entrée ou de sortie. Dans les designs fixes, les liens entre plusieurs documents sont créés automatiquement.
- Si l'option **Afficher le fichier de résultat après génération** est activée, MapForce ouvrira les fichiers générés dans le navigateur ou l'application par défaut, le cas échéant

### Limite de longueur de chemin

Utilisez ces options pour définir la longueur de chemin maximum à être affichée pour des items d'entrée ou de sortie ou des connexions. Par exemple, avec une longueur par défaut de **3**, un chemin d'item serait affiché comme **.../ShortPO/LineItems/LineItem**.

### Inclure

Choisir ici les composants spécifiques qui doivent être inclus dans la documentation générée.

### Détails

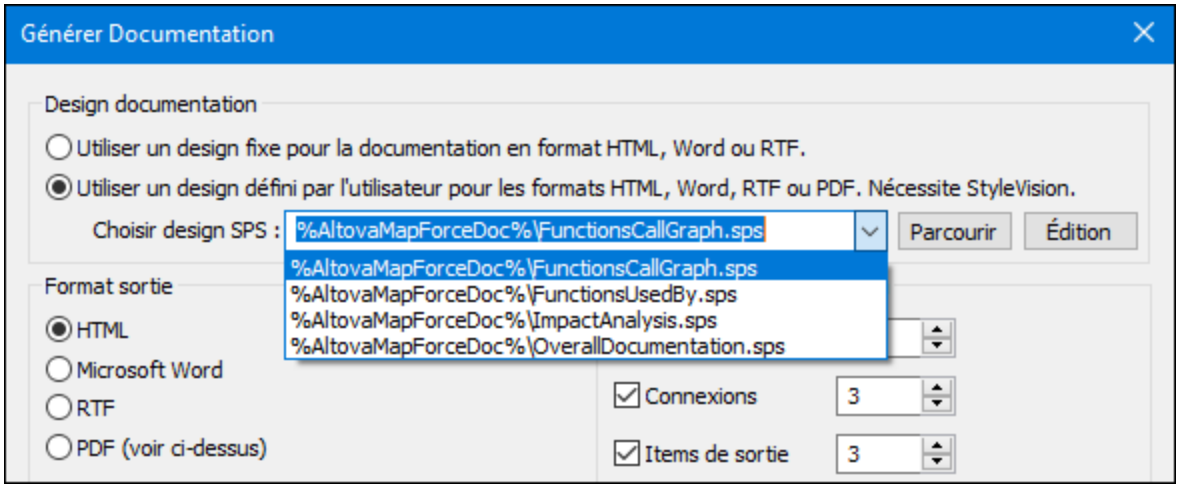
Utiliser ces options pour personnaliser le niveau de détail dans la documentation générée. L'option **Noms de bibliothèque** insère le préfixe "core" pour les fonctions.

# 8.1 Feuilles de style Power StyleVision prédéfinies

Si StyleVision est installé sur votre ordinateur, vous pouvez générer une documentation de mappage en choisissant un des fichiers StyleVision Power Stylesheet (SPS) prédéfinis en tant que modèle, au lieu du design fixe intégré. Les feuilles de style SPS prédéfinies sont disponibles :

- **FunctionCallGraph.sps** - montre le graphique d'appel du mappage principal et toute fonction définie par l'utilisateur.
- **FunctionsUsedBy.sps** - montre quelles fonctions sont utilisées directement ou indirectement dans le mappage.
- **ImpactAnalysis.sps** - recense tout nœud de source et de cible, et la route prise via les différentes fonctions vers le nœud de cible.
- **OverallDocumentation.sps** - montre toutes les connexions, fonctions et nœuds de cible. Ce modèle produit le maximum de détails et est identique à la sortie "fixed design" intégrée.

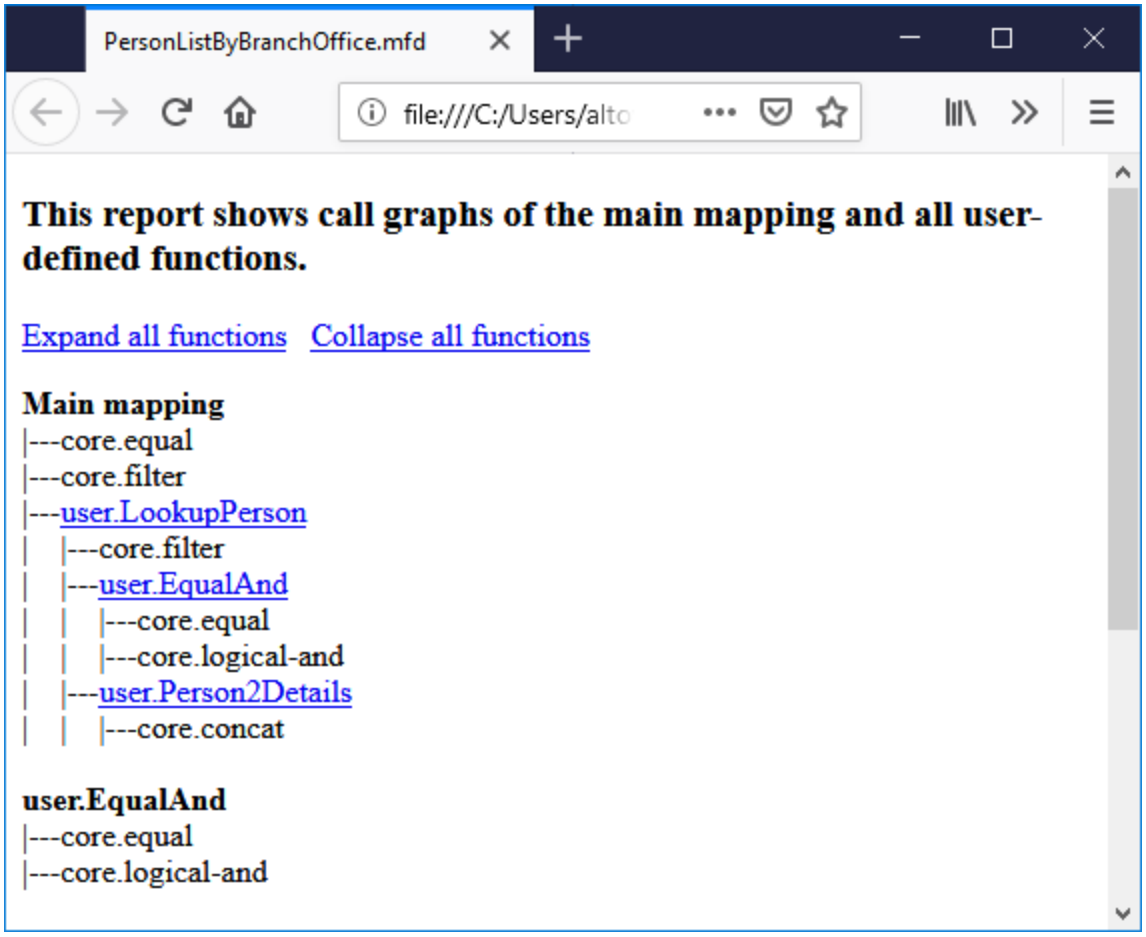
Vous pouvez sélectionner la feuille de style requise à chaque fois avant de générer une documentation, comme indiqué ci-dessous. Les fichiers se trouvent dans le dossier **...\MapForce2024\Documentation\MapForce**.



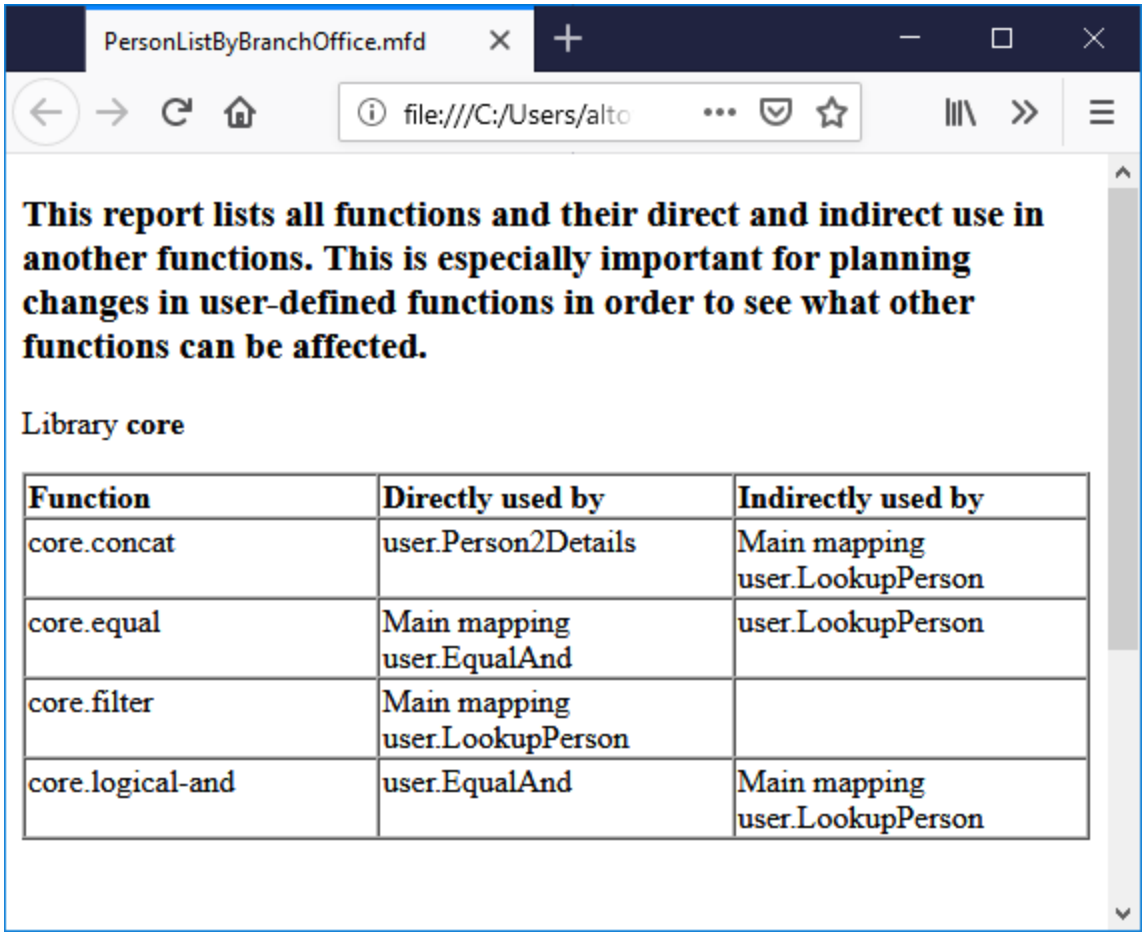

Les exemples ci-dessous illustrent la sortie produite par chacune de ces feuilles de style. Les exemples ont été générés depuis un des mappages de démonstration installés avec MapForce, **PersonListByBranchOffice.mfd**. Bien que ces exemples illustrent spécifiquement la sortie HTML, la mise en page est semblable à d'autres formats. Pour plus d'informations concernant la création ou la personnalisation de fichiers SPS, voir Feuilles de style personnalisées 831.

## Feuille de style "FunctionCallGraph.sps"



## Feuille de style "FunctionsUsedBy.sps"

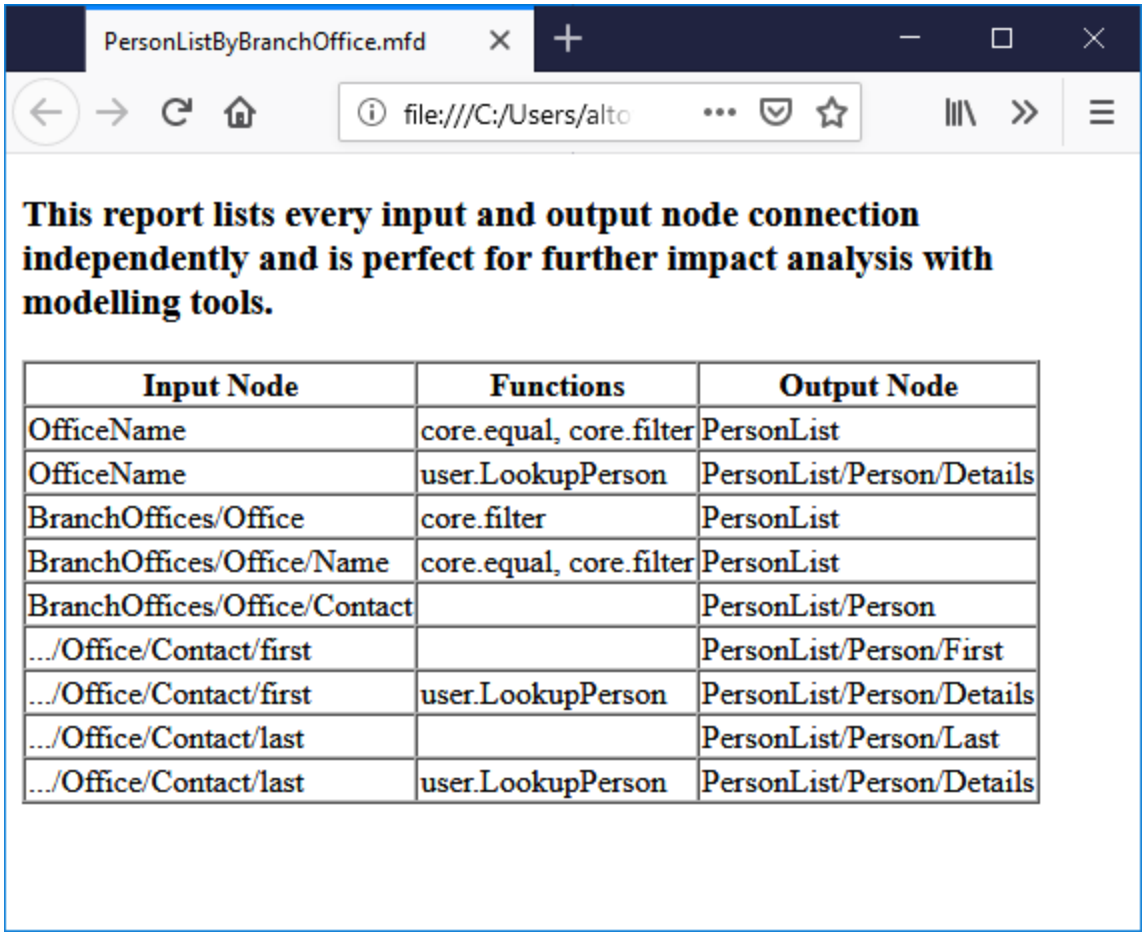



**This report lists all functions and their direct and indirect use in another functions. This is especially important for planning changes in user-defined functions in order to see what other functions can be affected.**

Library **core**

| Function | Directly used by | Indirectly used by |
|---|---|---|
| core.concat | user.Person2Details | Main mapping<br>user.LookupPerson |
| core.equal | Main mapping<br>user.EqualAnd | user.LookupPerson |
| core.filter | Main mapping<br>user.LookupPerson | |
| core.logical-and | user.EqualAnd | Main mapping<br>user.LookupPerson |

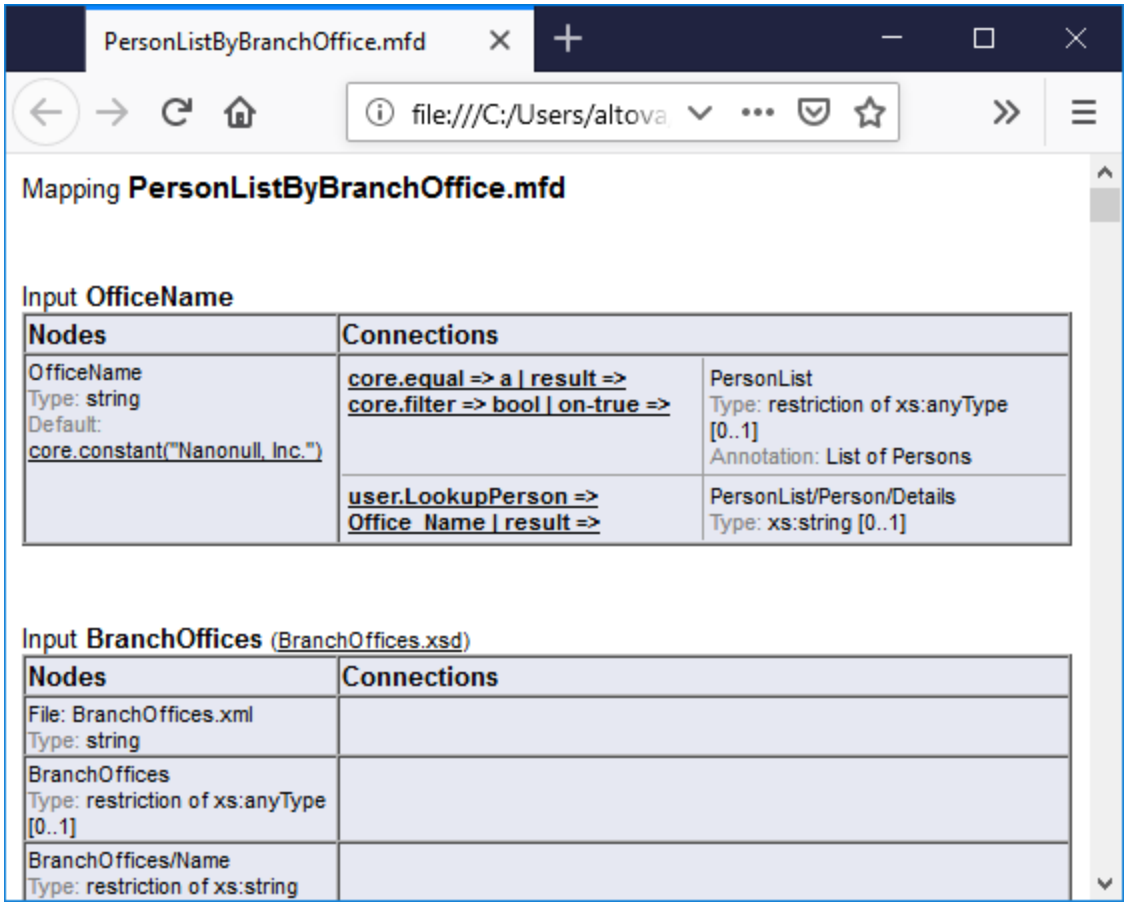
## Feuille de style "ImpactAnalysis.sps"




**This report lists every input and output node connection independently and is perfect for further impact analysis with modelling tools.**

| Input Node | Functions | Output Node |
|---|---|---|
| OfficeName | core.equal, core.filter | PersonList |
| OfficeName | user.LookupPerson | PersonList/Person/Details |
| BranchOffices/Office | core.filter | PersonList |
| BranchOffices/Office/Name | core.equal, core.filter | PersonList |
| BranchOffices/Office/Contact | | PersonList/Person |
| .../Office/Contact/first | | PersonList/Person/First |
| .../Office/Contact/first | user.LookupPerson | PersonList/Person/Details |
| .../Office/Contact/last | | PersonList/Person/Last |
| .../Office/Contact/last | user.LookupPerson | PersonList/Person/Details |

## Feuille de style "OverallDocumentation.sps"



## 8.2 Feuilles de style personnalisées

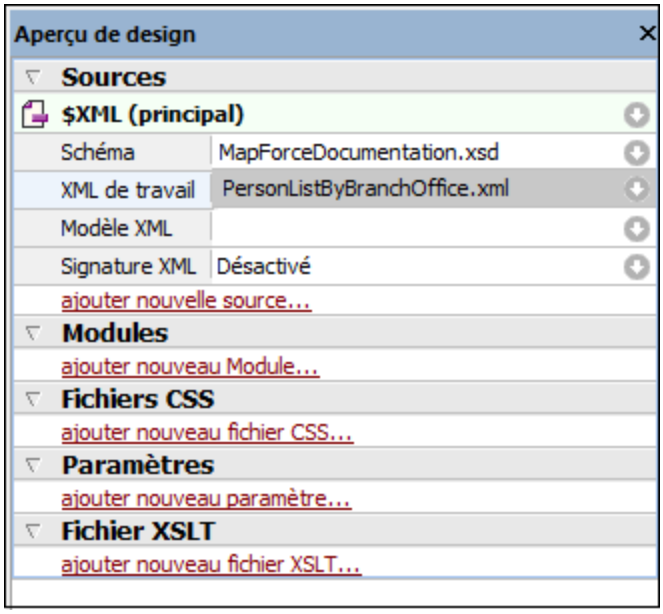
Outre le design fixe intégré, vous pouvez créer des feuilles de style personnalisées pour la documentation de mappages générés avec StyleVision (https://www.altova.com/stylevision). Vous pouvez aussi changer une des feuilles de style prédéfinies 826, par exemple, en ajustant les polices d'écriture et les autres styles.

Les StyleVision Power StyleSheet (SPS) vous permettent de réaliser un design personnalisé. L'avantage d'utiliser une SPS pour générer une documentation de mappage est que vous disposez du contrôle sur le design de la documentation.

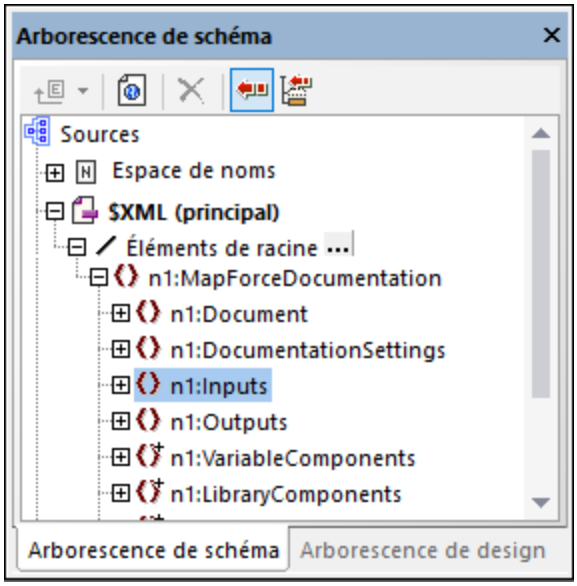
Pour créer un fichier SPS personnalisé, les conditions suivantes doivent être remplies :

1. Le Schéma XML qui apporte la structure de la documentation MapForce générée. Ce schéma est appelé **MapForceDocumentation.xsd** et est fourni avec votre pack d'installation MapForce. Il est stocké dans le dossier **...\Documents\Altova\MapForce2024\Documentation\MapForce**. Veuillez noter que **MapForceDocumentation.xsd** inclut le fichier **Documentation.xsd** situé dans le dossier ci-dessus.
2. Certaines données d'échantillon pour tester et apercevoir le design personnalisé. Vous pouvez utiliser le fichier XML suivant en tant que données d'échantillon : **...\Documents\Altova\MapForce2024\Documentation\MapForce\SampleData\PersonListByBranchOffice.xml**.

Les fichiers mentionnés ci-dessus doivent être référencés dans la fenêtre Aperçu de design dans StyleVision, par exemple :




Dans StyleVision, vous créez un design en glissant des nœuds depuis la fenêtre Arborescence de Schéma dans la zone de design puis y attribuez des styles et des propriétés.




Vous pouvez aussi ajouter des composants supplémentaires comme des liens et des images dans le design SPS. Pour consulter le design dans un format spécifique, cliquez sur un des onglets suivants : **HTML**, **RTF**, **PDF**, ou **Word 2007+**. Pour plus de détails, veuillez vous référer à la StyleVision documentation (https://www.altova.com/fr/documentation).

# 9 Débogueur

MapForce comprend un débogueur de mappage disponible pour le langage de transformation BUILT-IN de MapForce. Le débogueur de mappage vous aide à obtenir les objectifs suivants :

- Consulter et analyser les valeurs produites par le mappage à tout niveau de connecteur individuel.
- Marquer dans le mappage le contexte (jeu de nœuds) responsables pour la production d'une valeur particulière.
- Exécuter un mappage étape par étape, afin de voir comment MapForce traite ou calcule chaque valeur en temps réel et consulter la sortie de mappage telle qu'elle est générée.
- Définit des points d'étape (points d'arrêt) auxquels l'exécution de mappage doit s'arrêter et afficher les valeurs actuellement en cours de traitement.
- Afficher l'historique des valeurs traitées par un connecteur depuis le début de l'exécution de mappage jusqu'à la position d'exécution actuelle.

Le débogage d'un mappage est disponible lorsque le langage de transformation est en BUILT-IN. Si vous commencez à déboguer un mappage conçu pour un langage différent, vous serez invité à changer le langage de mappage à BUILT-IN. Vous pouvez aussi convertir un mappage en BUILT-IN en sélectionnant la commande de menu **Sortie | Moteur d'exécution Built-in**. Dans les deux cas, la conversion en BUILT-IN sera réussie si le mappage ne doit pas inclure les composants qui ne sont pas disponibles dans le langage BUILT-IN (par exemple, les fonctions XSLT).

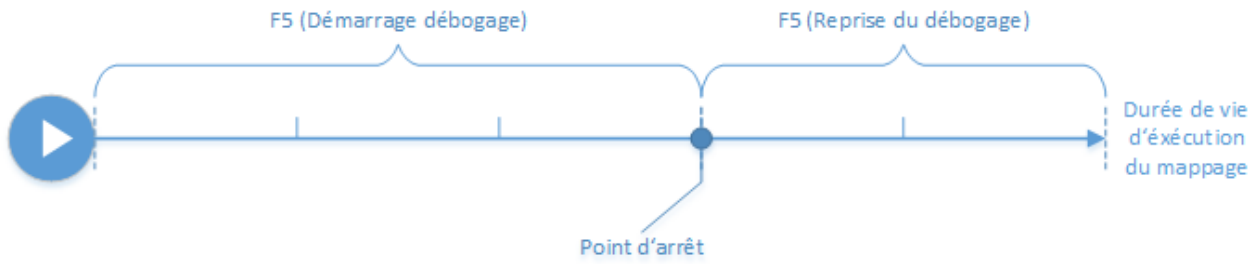
Le débogueur de MapForce, contrairement à un débogueur traditionnel, est un débogueur qui ne traverse pas la ligne de code du programme ligne par ligne (puisque vous n'écrivez aucun code avec MapForce). Au lieu de cela, le débogueur expose les résultats du code généré par MapForce produits depuis les mappages que vous concevez. Plus spécifiquement, le débogueur archive les valeurs qui sont transmises de et vers les composants de mappage par le biais de leurs connecteurs entrée et de sortie. Les valeurs archivées sont ensuite disponibles pour votre analyse directement dans le mappage ou par le biais de fenêtres dédiées.

Les sections suivantes marquent les différents moyens vous permettant d'utiliser le débogueur de mappage.

Les paramètres de débogueur sont disponibles dans la boîte de dialogue Options 1088. La liste de commandes de débogage disponibles est disponible dans Déboger 1078.

## Déboguer avec des points d'arrêt

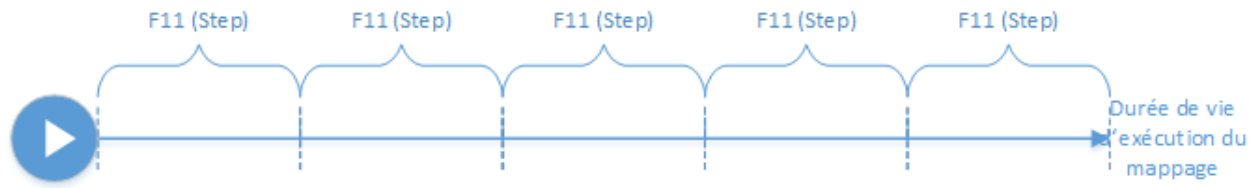
Lorsque vous devez stopper l'exécution de débogage à un endroit particulier du mappage, vous pouvez définir des points d'arrêt, comme vous le feriez dans un environnement de développement traditionnel. La différence est que les points d'arrêt sont ajoutés non pas à une ligne de code mais à un connecteur d'entrée ou de sortie d'un composant de mappage. Vous pouvez aussi ajouter des conditions aux points d'arrêt (ceci peut être utile si vous voulez interrompre l'exécution uniquement si la condition définie est satisfaite).




Vous pouvez définir des points d'arrêt sur les connecteurs désirés et exécutez le mappage jusqu'au premier point d'arrêt rencontré, ensuite passez au prochain, etc. De cette manière, vous pouvez analyser le contexte de mappage et les valeurs associées avec les connecteurs choisis. Vous pouvez aussi accélérer ou ralentir l'exécution avec les commandes **Pas à pas détaillé**, **Pas à pas sortant**, **Pas à pas principal**, et **Étape Minimale** fournies par le débogueur. Ces commandes vous permettent de sauter des portions du mappage, ou, au contraire, d'exécuter des portions du mappage d'une manière plus granulaire, si nécessaire.

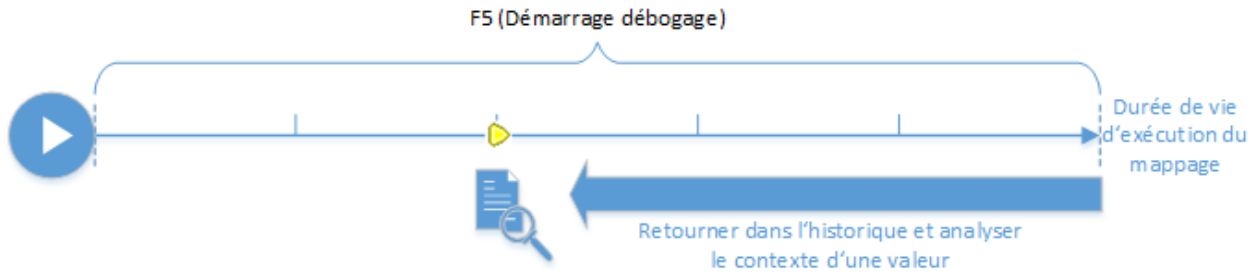
## Déboguer étape par étape

Vous pouvez débogueur un mappage étape par étape et analyser le contexte et les valeurs de mappage associés avec chaque étape. Ce scénario est semblable au scénario précédent, car vous pouvez accélérer ou ralentir l'exécution avec les commandes **Step Into**, **Step Out**, **Step Over** et **Minimal Step** commands.



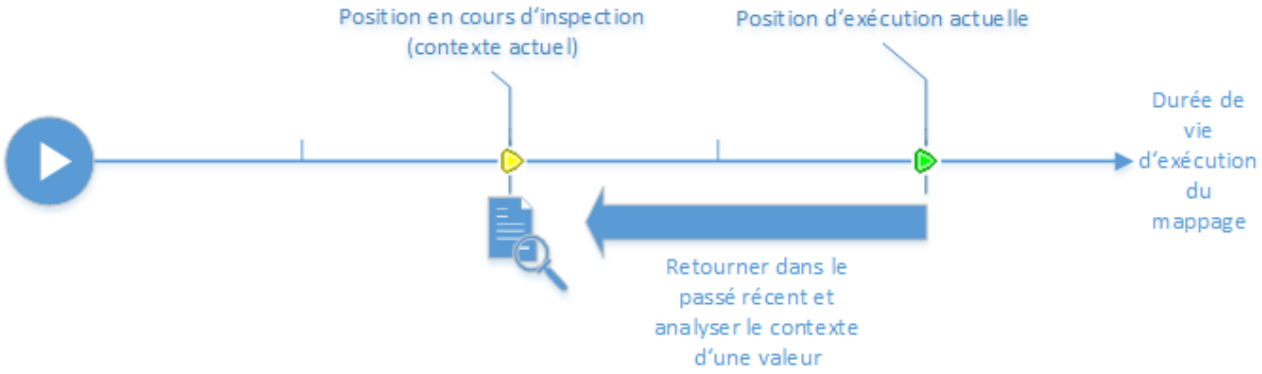

## Analyser le journal des valeurs

Vous pouvez configurer MapForce pour se souvenir du journal de toutes les valeurs (historique de traçage) qui ont été traitées par tous les connecteurs alors que vous déboguez un mappage. Il n'est pas forcément pertinent de garder l'historique complet de traçage pour les mappages qui demandent beaucoup de données, cette option peut donc être désactivée, le cas échéant. Lorsque l'option est activée, vous pouvez analyser le journal complet des valeurs traitées par chaque connecteur jusqu'à la position d'exécution actuelle. Vous pouvez aussi demander à MapForce de recréer le contexte de mappage associé avec une valeur particulière, ce qui peut vous aider à comprendre pourquoi cette valeur a été produite.



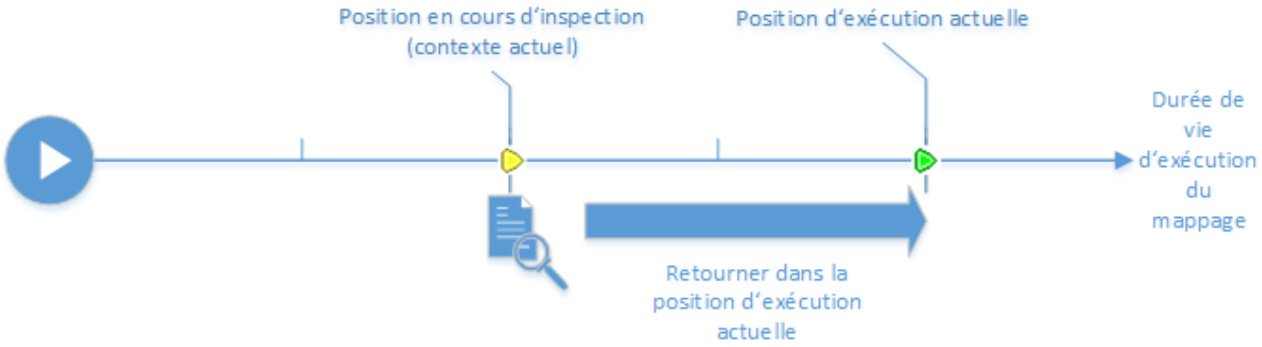

## Définir le contexte à une valeur liée à la position d'exécution actuelle

Lorsque le débogueur se trouve dans une position d'exécution particulière dans le mappage, il est possible d'analyser le contexte d'une valeur passée relativement à la position d'exécution actuelle (cela est comparable à un petit saut en arrière dans le temps) :




Un contexte a pour but d'expliquer pourquoi une valeur est calculée ; en d'autres termes, il décrit comment une valeur particulière dans le mappage a pu être générée. Le contexte est généralement la position d'exécution actuelle, bien que MapForce peut aussi vous permettre de définir un contexte dans le passé récent. Lorsque le contexte est défini sur une valeur particulière, MapForce marque directement sur le mappage les nœuds pertinents, propose des astuces à côté des connecteurs de mappage, et expose des informations supplémentaires dans les fenêtres ayant trait au débogueur (les fenêtres **Valeurs**, **Contexte**, et **Points d'arrêt**).

Une fois avoir inspecté un contexte de mappage qui n'est pas le même que celui de la position d'exécution actuelle, vous pouvez réinitialiser le contexte à la position d'exécution actuelle :




## Limitations

- Lorsque MapForce exécute un mappage, il peut optimiser du code en interne (par exemple, en cachant des données ou en calculant des résultats intermédiaires au niveau de points arbitraires). Cela peut provoquer l'inaccessibilité de certains connecteurs (et donc de points d'arrêt) pour le débogage, dans quel cas, MapForce affiche une notification. Veuillez noter que les optimisations de code MapForce (et, par conséquent, le comportement exposé par le débogueur) peut différer d'une release MapForce à une autre, même si l'outil de mappage est le même pour un mappage donné.
- Le débogueur peut déboguer la génération de sortie pour un composant cible à la fois. S'il existe plusieurs composants cible dans le mappage, vous devrez choisir lequel doit être exécuté par le débogueur.
- Actuellement, le débogage n'est pas pris en charge pour les actions de table de la base de données (telles que "Insert All", "Update If", etc.) des composants de base de données.
- Les points d'arrêt ne peuvent pas être ajoutés sur une des entités suivantes : constantes, la fonction **`core | position`**, les items descendants des connexions "Copier-tout", les paramètres des fonctions définies par l'utilisateur "inline".

# 9.1 Préparation de débogage

La préparation de débogage est nécessaire principalement pour les mappages de données volumineuses qui nécessitent généralement beaucoup de mémoire de système pour pouvoir être exécutés. C'est le cas des mappages qui traitent soit des fichiers d'entrée ou de sortie très volumineux, soit itèrent à répétition à travers une collection importante de données.

Pour accélérer le débogage et réduire les exigences de mémoire, il est recommandé de procéder aux étapes suivantes avant de commencer le débogage :

- Si le mappage est complexe, supprimer ou déconnecter des parties du mappage qui ne nécessitent pas de débogage.
- Si le mappage utilise des fichiers d'entrée volumineux, les remplacer avec des fichiers de taille plus réduite.
- Veuillez vous assurer que l'option **Garder historique du traçage complet** est désactivée (voir Paramètres de débogage 1088 )

De même, pour vous assurer que vous déboguez la sortie correcte, contrôlez les points suivants, le cas échéant :

- Si le mappage a plusieurs composants cible, choisir le composant cible à déboguer en cliquant sur la touche **Aperçu** (  ).
- Si le mappage est un mappage enchaîné (voir Mappages en chaîne 105), relâchez la touche **Passage** (  ) dans le composant intermédiaire. Le débogage de composants de Passage n'est pas prise en charge actuellement.

En option, si vous souhaitez que le débogueur s'arrête au niveau de connecteurs importants dont vous souhaitez analyser la valeur, ajoutez les points d'arrêt à ces connecteurs (voir Ajouter et supprimer des points d'arrêt 842).

## 9.2 À propos du mode Déboguer

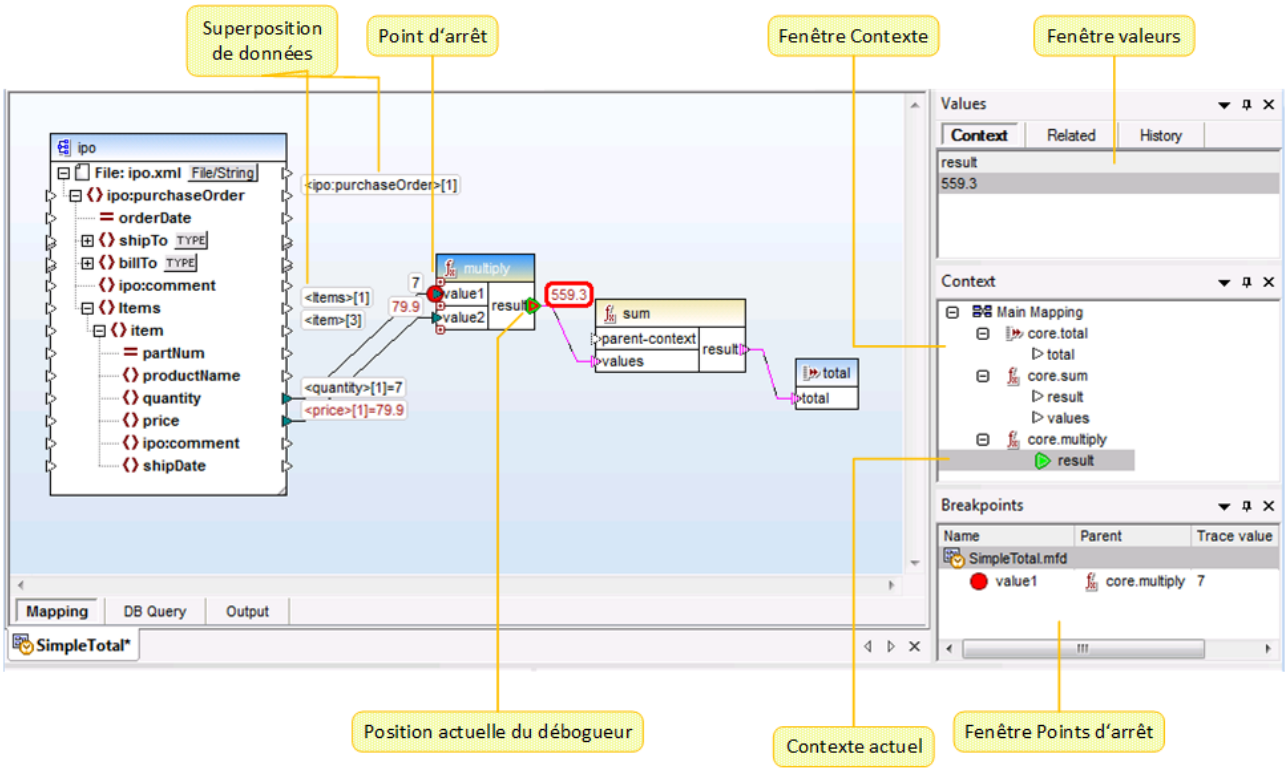
Lorsque vous lancez le débogage (en appuyant sur **F5**, ou **F11**, ou **Ctrl + F11**), MapForce exécute le mappage dans le mode de débogage.

> Tant que MapForce se trouve dans le mode débogage, le mappage est en lecture seule. Bien que vous puissiez déplacer des composants dans la zone de mappage, la plupart des commandes n'est pas disponible. Cela inclut des commandes comme la validation et le déploiement de mappage, la génération de code, les mappages de documentation, l'ajout de nouveaux composants dans la zone de mappage ou le rechargement de mappages existants, etc.

Le mode de débogage vous permet d'analyser le contexte responsable pour produire une valeur particulière. Cette information est disponible directement sur le mappage, et dans les fenêtres Valeurs, Contexte, et Points d'arrêt. Par défaut, ces fenêtres sont affichées lorsque vous commencez le débogage et sont dissimulées lorsque vous arrêtez le débogage.

MapForce se trouve en mode débogage (et le mappage est en lecture seule) jusqu'à ce que vous stoppiez le débogage, en appuyant sur **Shift + F5** (ou en cliquant sur la touche de la barre d'outils **Arrêter le débogage** ).

L'image suivante illustre un échantillon de mappage (**SimpleTotal.mfd**, depuis le répertoire **<Documents>\Altova\MapForce2024\MapForceExamples**) qui est débogué en plusieurs étapes (en appuyant sur **F11** pour avancer d'une étape).




*L'environnement de développement MapForce dans le mode de débogage*

Les astuces visuelles et les autres informations fournies par MapForce lorsque vous vous trouvez en mode de débogage sont décrites ci-dessous.

## Le volet de mappage

Pendant le débogage, le volet de mappage affiche des informations supplémentaires :

- Des superpositions de données (voir ci-dessous) montrent la valeur actuelle et les valeurs liées près de leurs connecteurs.
- Le contexte actuel (montré en tant que structure dans la fenêtre Contexte) est marqué comme suit :
  - Les connecteurs dans le contexte sont rayés en magenta ( ).
  - Les connecteurs dans le contexte ambigu sont pointillés en magenta ( ).
  - Les connexions dans le contexte sont rayées en magenta.
  - Les connexions dans le contexte ambigu sont rayées en magenta plus clair.
- L'emplacement d'exécution actuel est affiché avec une icône de connecteur verte ( ).

## Superpositions de données

Les valeurs traitées par chaque connecteur sont affichées en tant que superpositions de données (petits rectangles) près de leur connecteur correspondant. Une superposition de données actuellement sélectionnée est marquée par un gros cadre rouge. Les valeurs modifiées depuis la dernière étape sont affichées en rouge foncé. Pour des nœuds avec contenu simple, la superposition de données combine les deux valeurs - le nom de nœud et la valeur. Si le nom de nœud a été itéré plusieurs fois avant la

position d'exécution actuelle, l'index de l'itération actuelle est indiqué par le nombre contenu dans des crochets.

Les superpositions de données présentent le comportement suivant :

- Si vous pointez la souris sur une superposition de données, elle est amenée temporairement vers l'avant, si vous cliquez dessus, vous l'amenez définitivement vers l'avant. Un clic permet aussi de sélectionner le connecteur correspondant.
- Les superpositions de données peuvent être déplacées par glissement.
- Les superpositions de données se déplacent lorsqu'un composant est déplacé. Si les superpositions de données apparaissent superposées c'est parce que les composants sont trop proches l'un de l'autre. Faire glisser les composants autour de la zone de mappage pour avoir plus d'espace, et les superpositions de données se déplaceront en bloc avec le composant.
- Cliquer sur une superposition de données pour montrer sa valeur dans la fenêtre Valeurs.
- Cliquer sur un connecteur permet aussi de sélectionner sa superposition de données.

## Points d'arrêt

Les points d'arrêt sont des jalons, lorsque le mappage rencontre un point d'arrêt, il doit stopper au cours de l'exécution dans le mode de débogage. Ce terme vous est peut-être familier par analogie avec d'autres environnements de développement intégrés. Contrairement à d'autres environnements de développement dans lesquels vous ajoutez des points d'arrêt dans une ligne de code, un point d'arrêt dans MapForce peut être ajouté à un connecteur d'entrée ou de sortie (petit triangle à la gauche ou à la droite de la connexion). Dans le volet de mappage, les points d'arrêt sont représentés en tant que cercles rouges. Tout point d'arrêt défini est aussi affiché dans la fenêtre Points d'arrêt. Voir aussi Ajouter et supprimer des points d'arrêt 842.

## Position de débogueur actuelle

Le triangle vert (  ) indique la position du débogueur. Cette position est soit un connecteur d'entrée ou de sortie de tout composant donné.

La valeur actuellement traitée est aussi affichée dans la fenêtre Valeur, dans l'onglet **Contexte**.

L'ensemble des connexions et/ou des connecteurs colorés en magenta hachuré indiquent le contexte de mappage actuel. La même information est aussi affichée en tant que structure hiérarchique dans la fenêtre Contexte (voir Utiliser la fenêtre de Contexte 846).

Lorsque vous définissez manuellement le contexte d'une valeur, la position actuelle du débogueur se trouve dans une position dans le passé relative à la position d'exécution la plus récente. Pour vous aider à distinguer entre la position d'exécution la plus actuelle et celle se trouvant dans le passé, le connecteur "position actuelle" peut apparaître avec les couleurs suivantes dans l'interface du débogueur.

| | |
|---|---|
|  | Vert représente "le présent"; il indique la position d'exécution actuelle (voir Consulter la valeur actuelle d'un connecteur 851). |
|  | Jaune représente "le passé"; il indique que vous êtes en train de consulter un connecteur dans le passé, relatif à la position d'exécution actuelle. Cela peut se produire après avoir défini un contexte manuellement (voir Définir le contexte dans une valeur 854). |

## Fenêtre Valeurs

La fenêtre Valeurs fournit des informations concernant les valeurs traitées par le mappage. Elle vous permet de voir ce que le mappage traite à l'emplacement actuel de l'exécution, ou dans un contexte particulier que vous pouvez définir vous-même. Voir aussi Utiliser la fenêtre Valeurs 844.

## Fenêtre Contexte

La fenêtre Contexte offre un aperçu hiérarchique de l'ensemble de nœuds et fonctions qui sont pertinentes pour la position de débogueur actuelle. Voir aussi Utiliser la fenêtre Contexte 846.

## Fenêtre Points d'arrêt

La fenêtre Points d'arrêt affiche la liste des points d'arrêt de débogage créés depuis que MapForce a été lancé. Si vous avez défini des points d'arrêt sur plusieurs mappages, tous apparaîtront dans la fenêtre Points d'arrêt. Voir aussi Utiliser la fenêtre Points d'arrêt 848.

## 9.3 Ajouter et supprimer des points d'arrêt

Les points d'arrêt sont des jalons désignés au niveau desquels le mappage doit s'interrompre pendant l'exécution dans le mode de débogage. Tous les points d'arrêt que vous créez sont stockés globalement pour tous les mappages et sont affichés dans la fenêtre Points d'arrêt. Les points d'arrêt sont valides jusqu'à ce que vous les supprimiez explicitement ou fermez MapForce.

**Note :** Les points d'arrêt ne peuvent pas être ajoutés aux entités suivantes : constantes, fonction `core | position`, items descendants des connexions "Copy-all", paramètres des fonctions définies par l'utilisateur de "inline".

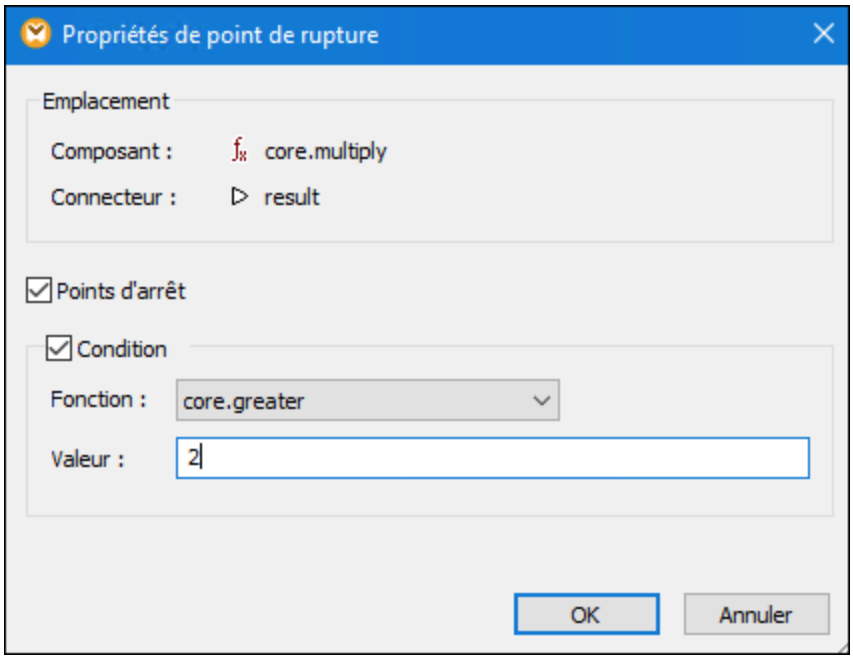
Les points d'arrêt peuvent être simples ou conditionnels. Les points d'arrêt simples arrêtent l'exécution de mappage de manière inconditionnelle. Les points d'arrêt conditionnels arrêtent l'exécution de mappage uniquement lorsque la condition qui leur est attribuée est satisfaite. Les conditions prennent la forme des fonctions de bibliothèques intégrées MapForce auxquelles vous fournissez des valeurs personnalisées. Autrement dit, si la condition retourne true, le point d'arrêt arrêtera l'exécution de mappage.

### Pour créer un point d'arrêt simple, suivre une des étapes suivantes :

- Cliquer avec la touche de droite sur un connecteur d'entrée ou de sortie (les petits triangles à gauche ou à droite d'un composant), et sélectionner **Point d'arrêt du débogueur**.
- Cliquer sur un connecteur d'entrée ou de sortie, et appuyer sur **F9**.

### Pour créer un point d'arrêt conditionnel :

1. Cliquer avec la touche de droite sur un connecteur, et choisir **Propriétés de point d'arrêt**.




2. Cliquer pour sélectionner les deux cases à cocher **Point d'arrêt** et **Condition**.

3. Choisir les fonctions requises depuis la liste et saisir la valeur de fonction (le cas échéant). Par exemple, dans l'exemple ci-dessus, le point d'arrêt arrêtera l'exécution de mappage si la valeur passant à travers est supérieure à 2.

> Si le type de données du connecteur dans lequel vous ajoutez le point d'arrêt conditionnel ne correspond pas au(x) type(s) attendu(s) par la fonction, MapForce tentera de convertir automatiquement le type de données. Si la conversion automatique n'est pas possible, l'exécution de mappage échouera. Pour éviter cela, s'assurer d'utiliser des types de données. Par exemple, la fonction `core.starts-with` attend une valeur de string pour que le connecteur du point d'arrêt ait le même type.

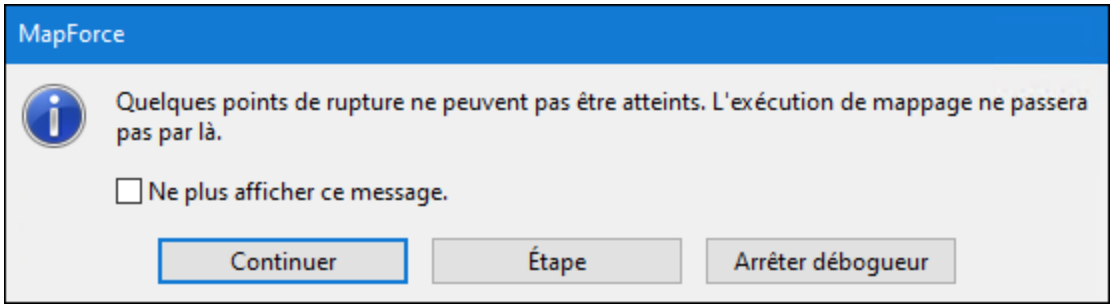
## Supprimer les points d'arrêt

Pour supprimer un point d'arrêt, cliquer avec la touche de droite sur le connecteur sur lequel le point d'arrêt existe, et sélectionner **Point d'arrêt de débogueur**. En alternative, cliquer sur le connecteur d'entrée ou de sortie sur lequel le point d'arrêt existe, puis appuyer sur **F9**.

Vous pouvez aussi supprimer des points d'arrêt depuis la fenêtre Points d'arrêt (voir Utiliser la fenêtre Points d'arrêt 848).

## Points d'arrêt non atteignables

Il peut arriver que MapForce affiche un message "Les points d'arrêt ne peuvent pas être atteints" :




Cela indique que les points d'arrêt ne peuvent pas être atteints par le débogueur, pour une des raisons suivantes :

- Un point d'arrêt a été défini sur un connecteur qui ne participe pas au mappage.
- Le point d'arrêt ne peut pas être atteint par MapForce à cause des optimisations d'exécution (voir Limites 836).

Cliquer sur **Continuer** pour avancer au prochain point d'arrêt défini (ou se rendre à la fin de l'exécution de débogage). Cliquer sur **Étape** pour lancer le débogage en étapes.

Vous pouvez désactiver les notifications à propos du point d'arrêt non atteignable rencontré par le débogueur, soit en cliquant sur **Ne pas afficher ce message à nouveau**, soit comme suit :

1. Dans le menu **Outils**, cliquer sur **Options**.
2. Cliquer sur **Messages**.
3. Cliquer pour supprimer la case à cocher **Informations concernant les points d'arrêt non atteignables**.

# 9.4 Utiliser la fenêtre Valeurs

La fenêtre Valeurs affiche des informations concernant les valeurs traitées par le mappage lorsqu'il se trouve en mode débogage. L'information affichée dans la fenêtre Valeurs dépend de la position actuelle du débogueur et des éléments d'interface utilisateur que vous avez cliqué. La fenêtre Valeurs contient les onglets suivants :

## L'onglet "Contexte"

L'onglet **Contexte** affiche la valeur actuellement traitée (la même valeur dont le contexte est affiché dans la fenêtre Contexte). Il s'agit soit de la valeur au niveau de la position d'exécution actuelle du débogueur, soit de la valeur d'un connecteur traité dans le passé. MapForce vous aide à distinguer entre les deux couleurs utilisées :

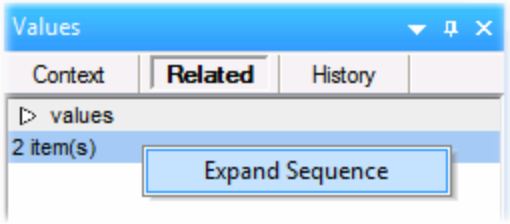
| | |
|---|---|
|  | Vert représente "le présent"; il indique la position d'exécution actuelle (voir Consulter la valeur actuelle d'un connecteur 851). |
|  | Jaune représente "le passé"; il indique que vous êtes en train de consulter un connecteur dans le passé, relatif à la position d'exécution actuelle. Cela peut se produire après avoir défini un contexte manuellement (voir Définir le contexte dans une valeur 854). |

## L'onglet "Lié"

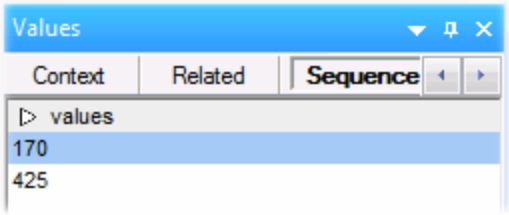
L'onglet **Lié** affiche des valeurs qui sont liées à (ou représentent le "passé proche" de) la valeur traitée actuellement. Normalement, vous n'avez pas besoin de cliquer expressément sur cet onglet ; MapForce y passe automatiquement lorsque vous cliquez sur la superposition de données d'un connecteur qui est lié à la position d'exécution actuelle du débogueur. Voir Retourner dans le passé récent 852.

## L'onglet "Séquence"

Si présent, l'onglet **Séquence** vous permet d'obtenir l'accès aux valeurs d'un connecteur qui traite une séquence. Cet onglet est uniquement visible si un connecteur a traité une séquence d'items (par exemple, avec une fonction d'agrégat comme `sum` ou `count`). lorsque vous cliquez sur la superposition de données d'un connecteur qui a traité une séquence d'items, la fenêtre Valeurs affiche une entrée dans le format "**n items**", où **n** est le nombre d'items traités par le connecteur. Pour obtenir l'accès à chaque valeur, double-cliquer sur cette entrée (ou cliquer avec la touche de droite et choisir **Agrandir la séquence** depuis le menu contextuel).




Les valeurs sont ensuite affichées dans l'onglet **Séquence**.




## L'onglet "Historique"

L'onglet **Historique** affiche des valeurs qui ont été traitées par un nœud particulier depuis le démarrage du débogage et jusqu'à la position d'exécution actuelle. Voir Consulter l'historique des valeurs traitées par un connecteur 853.

## 9.5 Utiliser la fenêtre Contexte

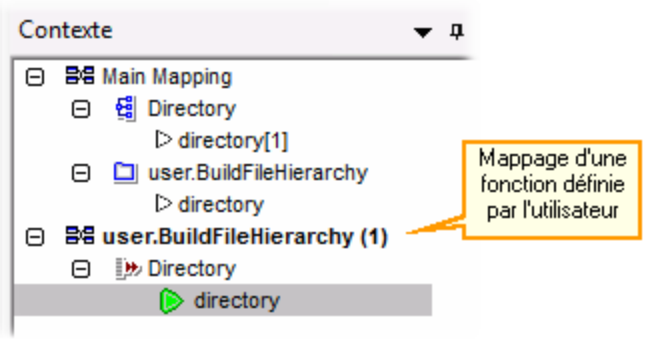
Pendant que MapForce se trouve en mode débogage, la fenêtre Contexte affiche une structure de connecteurs qui sont pertinents pour la position actuelle du débogueur. En d'autres termes, elle fournit le contexte de mappage responsable pour produire la valeur de mappage actuelle.

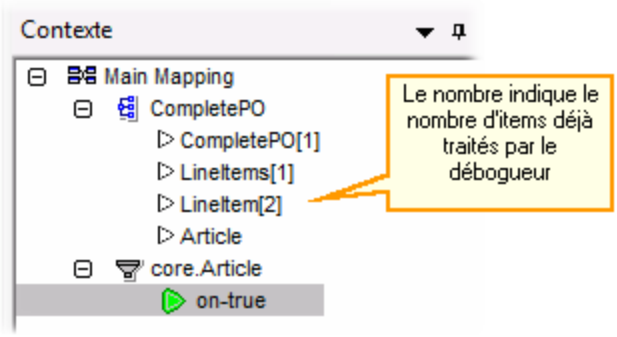
MapForce établit le contexte actuel comme suit :

1. Commencer avec le nœud racine de la structure de cible.
2. Descendre sur le nœud cible actuel.
3. À partir du nœud de cible actuel, se déplacer à gauche dans le mappage à travers tout composant qui mène à la position actuelle. Ces composants peuvent être des composants filtre ou tri, des fonctions intégrée ou définies par l'utilisateur, les variables, etc.

La fenêtre Contexte sert aussi bien en tant qu'aide informationnelle et de navigation. Pour choisir un nœud particulier dans le mappage directement depuis le contexte actuel, cliquer avec la touche de droite sur le nœud dans la fenêtre Contexte, et cliquer sur **Sélectionner dans mappage**. Cela peut être particulièrement utile lorsque le mappage est volumineux, afin d'éviter les défilements intensifs.

La fenêtre Contexte peut afficher les icônes spéciales suivantes et les notations :

| Icône | Description |
|---|---|
| | Représente le mappage auquel le contexte appartient. Il peut s'agir soit du mappage principal soit du mappage d'une fonction définie par l'utilisateur.   |
| ▷ | Représente un connecteur. Les nœuds cibles traitées jusqu'à présent présentent leur position qui s'affiche dans des crochets. |

| Icône | Description |
|---|---|
| |  |
|  | Représente le connecteur actuel (la position d'exécution la plus récente). Il s'agit de la source de la valeur actuelle dans la fenêtre Valeurs.<br><br>Dans quelques rares situations, il est possible qu'une valeur calculée soit utilisée pour plusieurs connecteurs. Dans ce cas, plusieurs icônes vertes apparaissent. |
|  | Représente le connecteur actuel lorsque le débogueur se trouve dans une position dans le passé relative au poste d'exécution le plus récent. Cela peut se produire après que vous définissiez le contexte dans une valeur (voir Définir le contexte dans une valeur 854). |

Outre les icônes ci-dessus, la fenêtre Contexte contient les icônes standard de tout type de composant qui est présent dans le mappage.

## Fenêtre Contexte et fonctions définies par l'utilisateur

Si le contexte actuel contient des fonctions définies par l'utilisateur, elles sont également affichées dans la fenêtre Contexte. Veuillez noter que si le contexte actuel sert à calculer une valeur d'entrée d'une fonction définie par l'utilisateur, le contexte est déterminé comme suit :

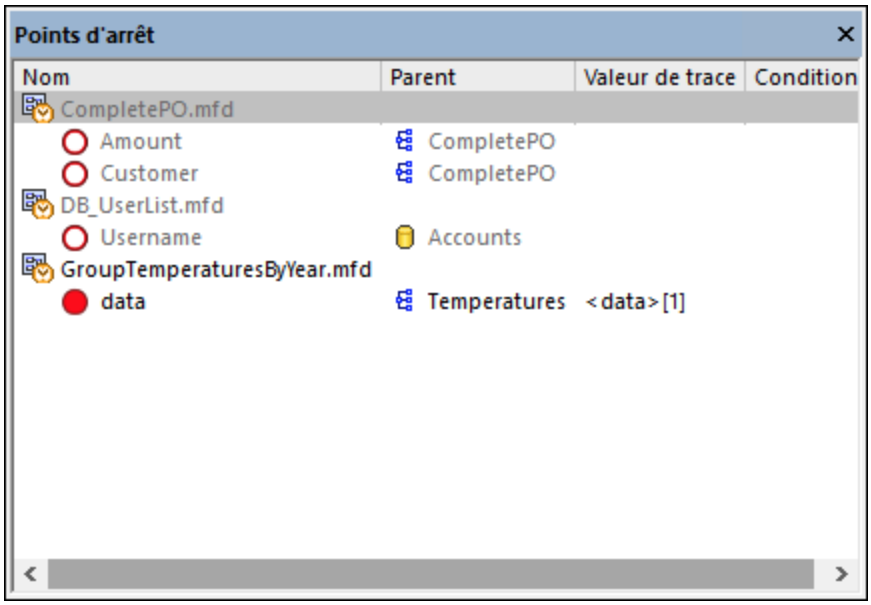
1. Depuis la cible vers le connecteur de sortie de la fonction définie par l'utilisateur vers le connecteur d'entrée de la fonction définie par l'utilisateur.
2. Depuis ici vers la gauche.

**Note :** Une fonction définie par l'utilisateur peut se produire plusieurs fois dans le contexte. Cela a lieu soit parce que plusieurs appels de fonction sont enchaînés soit parce que la fonction définie par l'utilisateur est définie comme récursive.

## 9.6 Utiliser la fenêtre Points d'arrêt

La fenêtre Points d'arrêt vous permet de consulter et de gérer les points d'arrêt globalement. Par défaut, la fenêtre Points d'arrêt est affichée lorsque MapForce se trouve en mode de débogage. Si vous souhaitez que la fenêtre Points d'arrêt soit visible en permanence, choisissez la commande de menu **Affichage | Fenêtre Déboguer | Points d'arrêt** lorsque MapForce n'est pas en mode de débogage.

La fenêtre Points d'arrêt affiche tous les points d'arrêt créés depuis que vous avez lancé MapForce, regroupés par le fichier de mappage auquel ils appartiennent. Alors que MapForce est ouvert, tout point d'arrêt associé avec un mappage est "mémorisé" par MapForce et est affiché dans la fenêtre Points d'arrêt, même si vous avez fermé le fichier de mappage entre-temps. Le mappage en cours de débogage actuellement est représenté avec une couleur de texte standard dans la fenêtre Points d'arrêt, alors que les autres mappages (ceux qui sont fermés ou actifs) sont grisés.




Vous pouvez rapidement ouvrir tout mappage en le double-cliquant (ou un de ses points d'arrêt) dans la fenêtre Points d'arrêt.

**Note :** Une fois que vous fermez ou redémarrez MapForce, tous les points d'arrêt seront supprimés.

Les informations concernant les points d'arrêt sont affichées sous forme de grille avec les colonnes suivantes :

| Colonne | Description |
|---|---|
| *Nom* | Le nom du nœud dont fait partie le point d'arrêt. |
| *Parent* | Le nom du composant de mappage dont fait partie le point d'arrêt. |
| *Valeur de trace* | La valeur qui passe à travers le connecteur sur lequel se trouve le point d'arrêt. La valeur de trace est affichée pendant l'exécution du débogage. |
| *Condition* | Si le point d'arrêt est conditionnel, cette colonne affiche la condition du point d'arrêt. |

Les points d'arrêt peuvent être associés à une des icônes suivantes.

| Icon | Description |
|---|---|
|  | Point d'arrêt actif. Dénote un point d'arrêt depuis un mappage actuellement en cours de débogage. |
|  | Point d'arrêt inactif. Dénote un point d'arrêt depuis un mappage ouvert, mais qui n'est pas actuellement en cours de débogage. |
|  | Point d'arrêt inaccessible. Dénote un point d'arrêt qui ne peut pas être atteint par le débogueur. |
|  | Point d'arrêt conditionnel. Dénote un point d'arrêt avec une condition qui lui est attachée. |

### Pour consulter ou modifier les propriétés d'un point d'arrêt :

- Cliquer dessus avec la touche de droite, et sélectionner **Propriétés de point d'arrêt** depuis le menu contextuel.

### Pour supprimer un point d'arrêt :

- Cliquer avec la touche de droite sur le point d'arrêt que vous souhaitez supprimer, puis sélectionner **Supprimer point d'arrêt** depuis le menu contextuel.
- Cliquer sur un point d'arrêt, puis appuyer sur **Supprimer**.

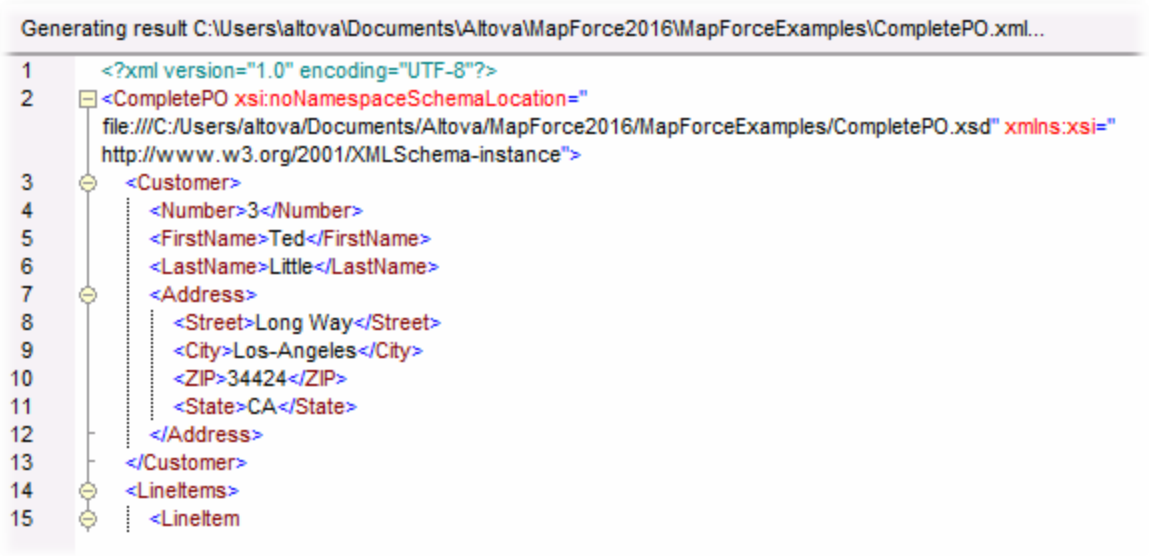
La commande de contexte **Supprimer tous les points d'arrêt** supprime tous les points d'arrêt affichés dans la fenêtre des points d'arrêt, quelque soit le mappage auquel ils appartiennent.

Voir aussi : Ajouter et supprimer des points d'arrêt 842

# 9.7 Visualiser une sortie partiellement générée

Lorsque vous déboguez en étapes ou en utilisant des points d'arrêt, vous pouvez consulter la sortie de mappage générée jusqu'à la position actuelle du débogueur. La visualisation de la sortie partiellement générée est prise en charge par XML, texte plat, et les composants de cible EDI.

Par défaut, lorsque vous appuyez sur **F5** (sans avoir défini de points d'arrêt), MapForce exécute l'ensemble du mappage dans le mode de débogage, puis passe à l'onglet **Sortie**, en affichant la sortie finale générée. Néanmoins, si vous avez défini des points d'arrêt ou si vous déboguez par étapes (**F11**, ou **Ctrl + F11**), l'exécution du débogueur stoppe alors que la sortie de mappage est en cours de génération. Même si la sortie de mappage est partiellement rédigée à ce niveau, vous pouvez toujours cliquer sur l'onglet **Sortie**, et la visualiser.



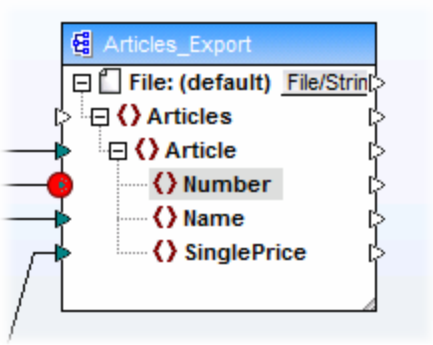

## Limites

- Le nœud de cible actuellement calculé n'est pas toujours affiché dans l'onglet **Sortie**. Par exemple, les attributs XML sont rassemblés en interne et rédigés immédiatement.
- Si la sortie produit plusieurs fichiers, seul le fichier rédigé actuellement peut être affiché ; le passage à un autre fichier de sortie est désactivé.

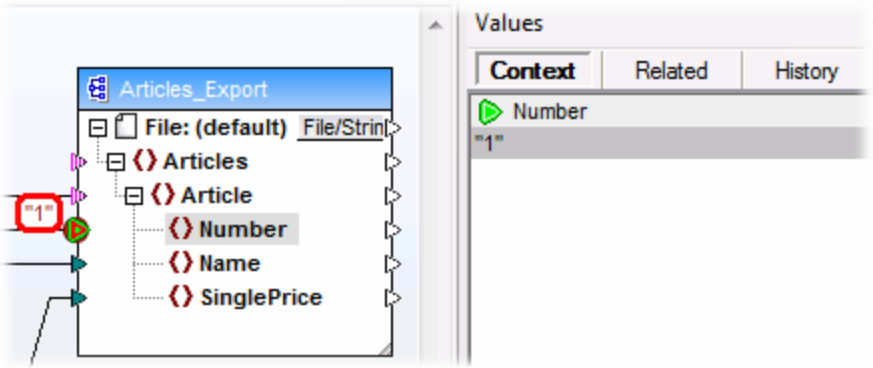
# 9.8 Consulter la valeur actuelle d'un connecteur

Lorsque la position d'exécution actuelle du débogueur ( ) se trouve sur un connecteur particulier (soit parce que vous effectuez un débogage en étapes ou parce qu'il y a un point d'arrêt défini dans le connecteur), la valeur actuelle traitée par le connecteur est affichée dans l'onglet **Contexte** de la fenêtre Valeurs. Il s'agit de la valeur qui va être rédigée dans la sortie, c'est à dire, "le présent". Il s'agit aussi de la valeur dont le contexte est affiché dans la fenêtre Contexte (voir Utiliser la fenêtre Contexte 846).

Pour comprendre cet exemple, ouvrir l'échantillon **PreserveFormatting.mfd** dans le répertoire **<Documents>\Altova\MapForce2024\MapForceExamples**. Cliquer dans le connecteur d'entrée du nœud `Number` dans le composant de cible, et appuyer sur **F9** pour y ajouter un point d'arrêt.




Ensuite, appuyer sur **F5** pour commencer le débogage et observer les résultats.



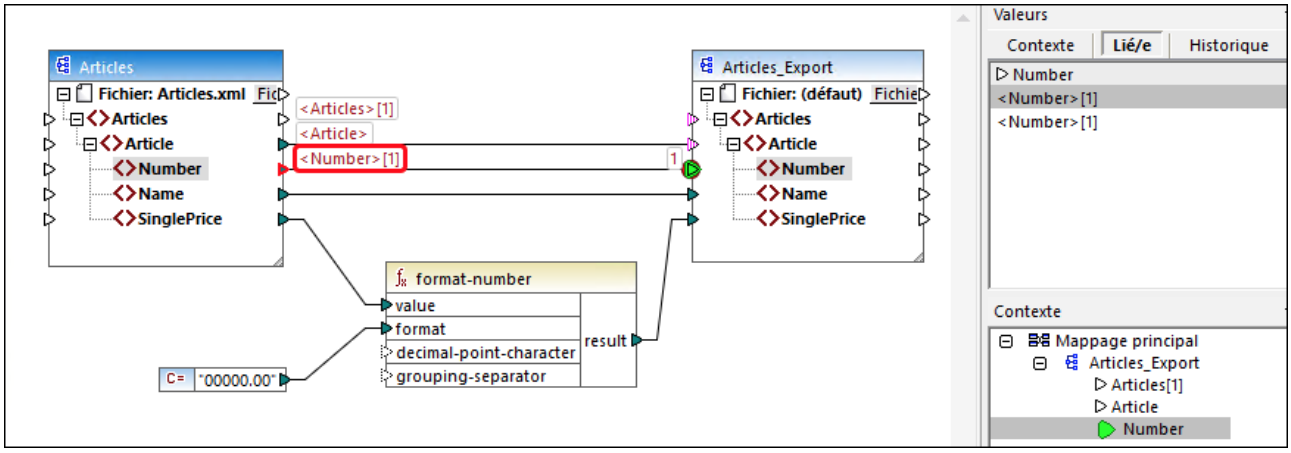

Comme indiqué dans l'image, la position de débogueur actuelle (et le point d'arrêt ) se trouve sur le nœud `Number` du composant de cible. La fenêtre Valeurs indique que ce nœud traite la valeur "1" (cette valeur est aussi marquée avec un bord rouge épais dans le mappage).

# 9.9 Retourner dans le passé récent

Lorsque vous cliquez sur une superposition de données (petite case rectangulaire) à côté d'un connecteur de mappage, la fenêtre **Valeurs** affiche le nom et, en option, la valeur associée avec le connecteur sélectionné. Le centre d'attention n'est plus sur la position actuelle du débogueur, mais sur la superposition de données sélectionnée. Vous pouvez considérer cette vue comme un petit saut en arrière dans le temps dans l'historique de débogage. Il s'agit du passé "récent", puisque le mappage affiche uniquement des superpositions de données pour les derniers connecteurs liés à la position du débogueur actuelle. Lorsque vous cliquez sur une telle superposition de données "liées", la fenêtre Valeurs passe automatiquement à l'onglet **Lié**.

Pour une illustration de ce scénario, ouvrir le mappage **PreserveFormatting.mfd** du répertoire **<Documents>\Altova\MapForce2024\MapForceExamples**.

Une fois avoir ouvert le mappage, cliquer sur le connecteur situé à côté du nœud `Number` dans le composant de cible, et appuyer sur **F9** pour y ajouter un point d'arrêt. Appuyer sur **F5** pour commencer le débogage, puis cliquer sur la superposition de données (petite case rectangulaire) à côté du nœud`Number` du composant de source.




Étant donné qu'un connecteur est généralement itéré plusieurs fois pendant la durée de vie d'un mappage, l'index actuel de l'itération est affiché entre des crochets : **<Number>[1]**. De même, étant donné que le connecteur porte une valeur, celle-ci est aussi représentée après le signe égal : **<Number>[1]=1**. La même valeur est affichée sur une nouvelle ligne dans la fenêtre Valeurs, comme indiqué ci-dessous.

Si vous nécessitez des informations supplémentaires concernant une valeur particulière, n'oubliez pas que vous pouvez recréer le contexte qui l'a produit (voir Configurer le Contexte en une Valeur 854 ).

# 9.10 Consulter l'historique des valeurs traitées par un connecteur

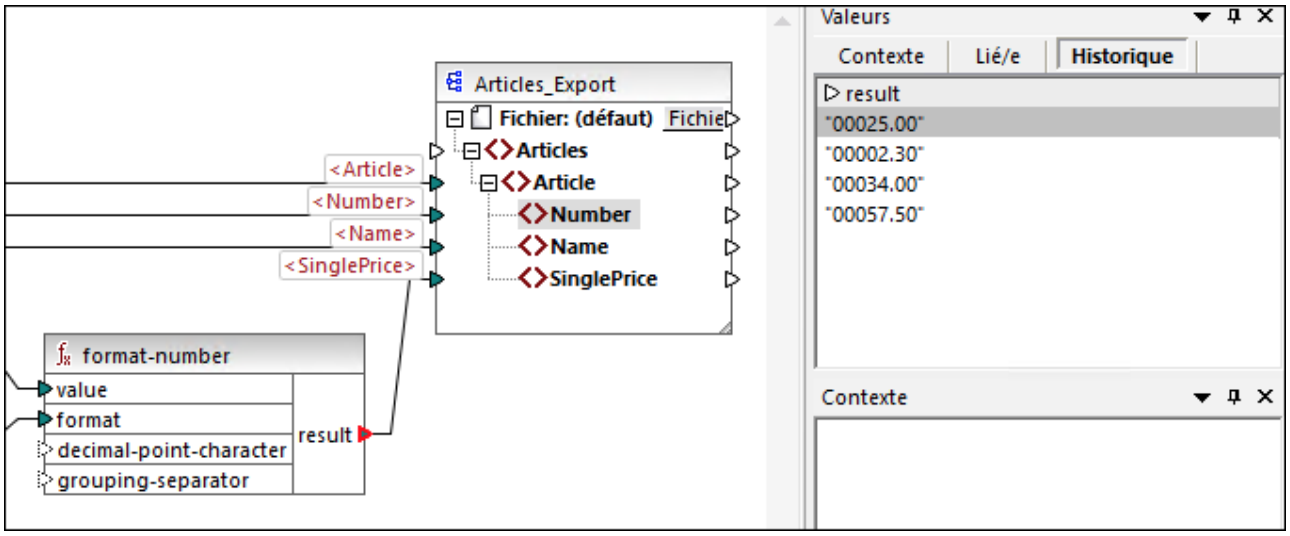
Si l'option **Garder historique du traçage complet** est activée (voir Paramètres du débogueur[1088]), vous pouvez consulter l'historique de toutes les valeurs qui ont été traitées par ce connecteur (jusqu'à la position d'exécution actuelle).

L'historique est affiché lorsque vous cliquez sur un connecteur, puis cliquez sur l'onglet **Historique** de la fenêtre Valeurs. Veuillez noter que cette opération est uniquement utile pour les connecteurs qui ont des valeurs traitées depuis le début de l'exécution du mappage jusqu'à la position du débogueur actuel.

Pour illustrer cet exemple, procédons à un débogage d'un mappage du début à la fin sans utiliser de points d'arrêt, puis observons l'historique de toutes les valeurs qui ont été traitées par un connecteur particulier. En premier lieu, ouvrez le mappage **PreserveFormatting.mfd** depuis le répertoire **<Documents>\Altova\MapForce2024\MapForceExamples** . S'il est déjà ouvert, assurez-vous de procéder comme suit :

- Supprimer tous les points d'arrêt, le cas échéant, (voir Ajouter et supprimer des points d'arrêt[842] )
- Arrêter le débogage s'il est actuellement en cours d'exécution, en appuyant sur **Shift + F5**.

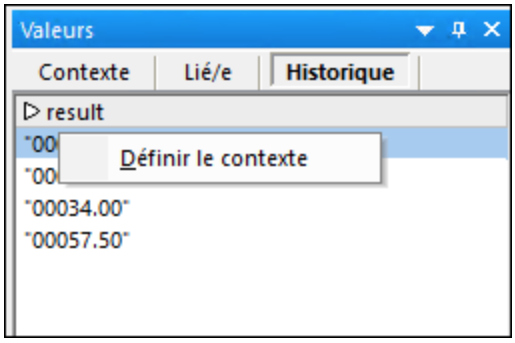
Lorsque vous êtes prêt, appuyez sur **F5** pour lancer une nouvelle opération de débogage. Lorsque vous appuyez sur **F5**, MapForce exécute le mappage en mode débogage et passe à l'onglet **Sortie**. Cliquez sur l'onglet **Mappage** pour retourner à la fenêtre de mappage principal, puis cliquez sur le nœud `result` de la fonction `format-number` (marquée en rouge dans l'image ci-dessous). Enfin, cliquez sur l'onglet **Historique** de la fenêtre Valeurs, et notez les valeurs affichées.




Comme indiqué dans l'image ci-dessus, ce nœud particulier (`result`) a traité quatre valeurs au total. Si vous nécessitez plus d'informations concernant une valeur particulière, n'oubliez pas que vous pouvez recréer le contexte qui l'a produit (voir Configurer le Contexte en une Valeur[854]).

# 9.11 Configurer le Contexte en une Valeur

Configurer le contexte en une valeur est une action qui peut être comparée à un saut dans le passé, afin de consulter plus de détails à propos du contexte de mappage qui a produit cette valeur. Vous pouvez définir le contexte sur toute valeur affichée dans la fenêtre Valeurs (dans l'onglet **Lié**, l'onglet **Séquence** ou l'onglet **Historique**). Si vous avez activé l'option **Garder historique du traçage complet** (voir Paramètres du débogueur 1088), l'onglet **Historique** affiche toutes les valeurs traitées par le connecteur sélectionnées actuellement ; c'est pourquoi, dans ce cas, vous pouvez en plus définir le contexte sur toute valeur dans le passé pour ce connecteur.

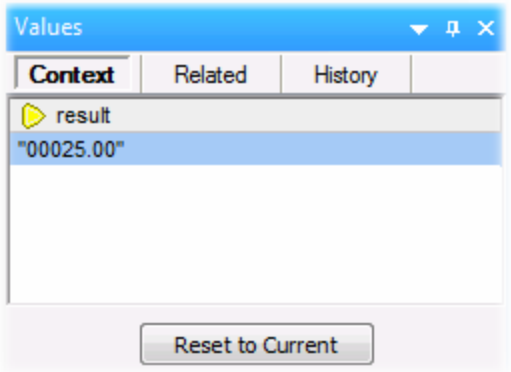



Pour définir le contexte en une valeur, suivre une des étapes suivantes :

- Cliquez avec la touche de droite sur la valeur, et choisissez **Définir Contexte** depuis le menu contextuel.
- Double-cliquer sur la valeur.

Lorsque vous définissez le contexte á une valeur, MapForce met la zone de mappage en surbrillance afin de recréer la situation qui a produit la valeur, et peuple la fenêtre **Valeurs** et la fenêtre **Contexte** conformément au contexte sélectionné. Pour consulter une légende des indications visuelles utilisées dans la zone de mappage tout en se trouvant dans un contexte, voir À propos du mode Déboguer 838. Pour plus d'informations concernant le contexte lui-même, voir Utiliser la fenêtre Contexte 846.

Le connecteur d'un contexte défini manuellement est jaune ( ), qui indique que vous ne vous trouvez plus dans la position d'exécution la plus récente. Pour retourner à la position d'exécution la plus récente (le cas échéant), cliquer sur la touche **Réinitialiser sur actuel** de l'onglet **Contexte** dans la fenêtre Valeurs.



# 10 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 856.
- *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. Voir Automatisation avec MapForce Server 857.

** Le langage de transformation BUILT-IN nécessite l'édition MapForce Professional ou Enterprise.*

De plus, MapForce permet d'automatiser la génération de code XSLT, XQuery, C#, C++ et Java depuis l'interface de ligne de commande. Cela inclut la capacité à compiler les fichiers d'exécution de serveur (.mfx) prévus pour l'exécution MapForce Server. Pour plus d'informations, voir Interface de ligne de commande MapForce 876.

# 10.1 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, ou dans XQuery, 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 (ou XQuery) sur le serveur.

Si vous avez l'intention d'exécuter ou d'automatiser les mappages MapForce pour d'autres sorties sur un serveur, voir Automatisation avec MapForce Server 857.

**Note :** Vous pouvez aussi consulter le code XSLT 70 et XQuery à l'aide du moteur intégré.

## 10.2 Automatisation avec MapForce Server

MapForce Server est une solution logicielle de serveur pour entreprise conçue pour fonctionner avec les systèmes d'exploitation Windows, Linux et macOS. Le rôle de MapForce Server est d'exécuter des mappages dans un environnement de serveur (y compris sur des plateformes non-Windows) et avec une performance de niveau serveur. Tout mappage MapForce dont le langage d'exécution cible est BUILT-IN se qualifie pour une exécution de serveur (voir aussi Choisir un langage de transformation 22). MapForce Server peut fonctionner soit en autonomie (invoqué depuis la ligne de commande ou l'API), soit sous la gestion de FlowForce Server.

Si MapForce Server est utilisé en tant que produit individuel, le mappage MapForce doit être compilé et copié sur l'appareil sur lequel MapForce Server est exécuté. Le mappage est ensuite exécuté à l'aide de la commande de ligne de commande `run`. Vous pouvez aussi exécuter le mappage en invoquant la méthode `run` de l'API de MapForce Server. Pour plus d'informations, voir Compiler des mappages sur des fichiers d'exécution MapForce Server 864.

Si MapForce Server est exécuté sous la gestion de FlowForce Server, le mappage peut être déployé sur un appareil cible par le biais d'une connexion HTTP (ou SSL/HTTPS) directement depuis MapForce. Sur le serveur, le mappage peut ensuite être exécuté en tant que tâche déclenchée ou programmée, ou bien, par le biais d'un appel de service Web défini depuis l'interface d'administration de FlowForce Server. Pour plus d'informations, voir Déployer des mappages sur FlowForce Server 867.

Il existe deux éditions de MapForce Server:

- MapForce Server
- MapForce Server Advanced Edition

MapForce Server Advanced Edition fournit les mêmes fonctions que MapForce Server, et inclut en plus des fonctions d'optimisation pour les mappages qui se qualifient pour l'optimisation. C'est le cas avec les mappages qui joignent ou filtrent des quantités importantes de données et lorsqu'il est possible d'appliquer une optimisation join de manière à augmenter la vitesse d'exécution. Contrairement à MapForce Server, MapForce Server Advanced Edition peut exécuter des mappages lorsque des fonctions de nœud sont présentes, voir Défauts et fonctions de nœud 476.

Limites :

- Les signatures numériques XML ne sont pas prises en charge
- Les connexions de base de données ADO, ADO.NET et ODBC sont uniquement prises en charge sur Windows. (Pour les autres systèmes d'exploitation, voir Préparer des mappages pour l'exécution de serveur. 858).

Pour plus d'informations concernant MapForce Server, veuillez vous référer à sa documentation (https://www.altova.com/fr/documentation).

# 10.3 Préparer des mappages pour l'exécution de serveur

Un mappage conçu et prévisionné avec MapForce peut référer à des ressources se trouvant en dehors de l'appareil et du système d'exploitation actuels (comme des bases de données). Outre ce fait, dans MapForce, tous les chemins de mappage suivent des conventions de style Windows par défaut. De plus, l'appareil sur lequel MapForce Server est exécuté pourrait ne pas prendre en charge les mêmes connexions de base de données que l'appareil sur lequel de mappage a été conçu. C'est pour cette raison que les mappage exécutés dans un environnement de serveur nécessitent généralement une certaine préparation, en particulier si l'appareil cible n'est pas le même que l'appareil de source.

**Note :** Le terme "appareil source" renvoie à l'ordinateur sur lequel MapForce est installé et le terme "appareil cible" réfère à l'ordinateur sur lequel MapForce Server ou FlowForce Server est installé. Dans le scénario le plus simple, il s'agit du même ordinateur. Dans un scénario plus complexe, MapForce est exécuté sur un appareil Windows alors que MapForce Server ou FlowForce Server est exécuté sur un appareil Linux ou macOS.

En règle générale, il est recommandé de s'assurer que le mappage soit validé avec succès dans MapForce avant de le déployer dans FlowForce Server ou de le compiler dans un fichier d'exécution MapForce Server (voir Valider les mappages 68).

Si MapForce Server est exécuté seul (sans FlowForce Server), les licences requises sont les suivantes :

- Sur l'appareil source, l'édition MapForce Enterprise ou Professional est exigée pour concevoir le mappage et le compiler dans un fichier d'exécution de serveur (.mfx), voir Compiler des mappages sur des fichiers MapForce Server Execution 864.
- Sur l'appareil cible, MapForce Server ou MapForce Server Advanced Edition est exigé pour exécuter le mappage.

Si MapForce Server est exécuté sous la gestion FlowForce Server, les exigences suivantes s'appliquent :

- Sur l'appareil source, l'édition MapForce Enterprise ou Professional est exigée pour concevoir le mappage et le déployer sur un appareil cible, voir Déployer les mappages sur FlowForce Server 899.
- MapForce Server et FlowForce Server doivent tous deux être sous licence sur l'appareil cible. Le rôle de MapForce Server est d'exécuter le mappage ; le rôle de FlowForce est de rendre le mappage disponible en tant que tâche qui profite des fonctions telles que exécutions programmées ou exécution sur demande, exécution en tant que service Web, gestion d'erreur, traitement conditionnel, notifications d'e-mail, etc.
- FlowForce Server doit être activé et exécuté sous l'adresse et le port de réseau. Le service "FlowForce Web Server" doit être lancé et configuré pour accepter les connexions provenant des clients HTTP (ou HTTPS si configuré) et ne doit pas être bloqué par le pare-feu. Le service "FlowForce Server" doit aussi être démarré et exécuté à l'adresse et le port désigné.
- Vous devez disposer d'un compte utilisateur FlowForce Server avec les permissions d'un des conteneurs (par défaut, le conteneur **/public** est accessible à n'importe quel utilisateur authentifié).

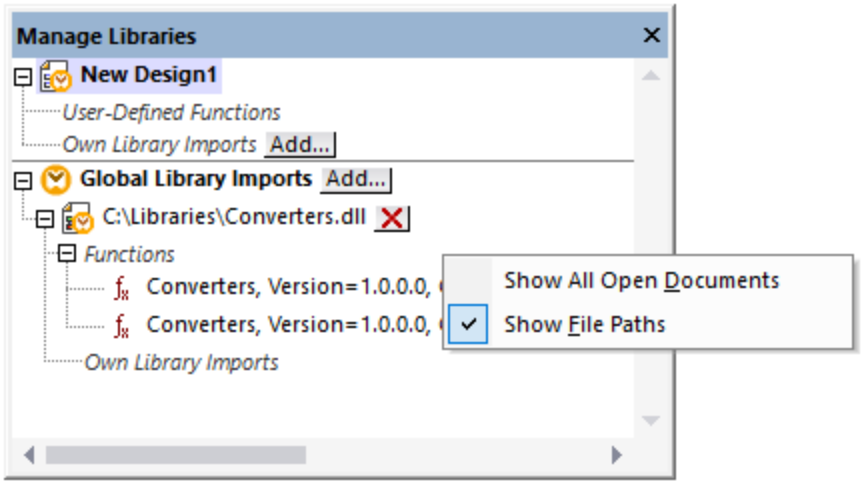
## Considérations d'ordre général

- Si vous comptez exécuter le mappage sur un appareil cible avec un MapForce Server autonome, tous les fichiers d'entrée référencés par le mappage doivent également être copiés dans l'appareil cible. Si MapForce Server est exécuté sous la gestion FlowForce Server, aucun besoin de copier les fichiers manuellement. Dans ce cas, les fichiers d'instance et de schéma sont inclus dans le pack déployé sur

l'appareil cible, voir Déployer des mappages sur FlowForce Server[899].

- Si le mappage comprend des composants de base de données qui nécessitent des pilotes de base de données spécifiques, ces pilotes doivent aussi être installés sur l'appareil cible. Par exemple, si votre mappage lit des données provenant d'une base de données Microsoft Access, Microsoft Access ou Microsoft Access Runtime (https://www.microsoft.com/en-us/download/details.aspx?id=50040) doit aussi être installés sur l'appareil cible.
- Lorsque vous déployez un mappage sur des plateformes non-Windows, les connexions de base de données ADO, ADO.NET et ODBC sont modifiées automatiquement en JDBC. Les connexions SQLite natives et PostgreSQL natives sont préservées et ne nécessitent aucune autre configuration. Voir "Connexions aux bases de données" ci-dessous.
- Si le mappage contient des appels de fonction personnalisés (par exemple vers des fichiers .dll ou .class), Ces dépendances ne sont pas déployées avec le mappage, puisqu'ils ne sont pas connus avant l'exécution. Dans ce cas, les copier manuellement dans l'appareil cible. Le chemin du fichier .dll ou .class sur le serveur doit être le même que dans la fenêtre "Gérer bibliothèques" de MapForce, par exemple.




- Certains mappages lisent plusieurs fichiers d'entrée en utilisant un chemin de caractère générique (voir Traiter plusieurs fichiers d'entrée et de sortie dynamiquement[787]). Dans ce cas, les noms de fichier d'entrée ne sont pas connus avant l'exécution et ils ne seront donc pas déployés. Pour que le mappage puisse être exécuté correctement, les fichiers d'entrée doivent exister sur l'appareil cible.
- Si le chemin de sortie de mappage contient des répertoires, ces répertoires doivent exister sur la machine cible. Sinon, une erreur sera générée lorsque vous exécuterez le mappage. Ce comportement est différent de celui sur MapForce, où des répertoires non existants sont générés automatiquement si l'option **Générer une sortie dans les fichiers temporaires** est activée (voir Modifier les options MapForce[1086]).
- Si le mappage appelle un service web qui nécessite une authentification HTTPS avec un certificat client, le certificat doit aussi être transféré à l'appareil cible, voir .
- Si le mappage se connecte à des bases de données basées sur des fichiers comme Microsoft Access et SQLite, le fichier de base de données doit être transféré manuellement à l'appareil cible ou enregistré sous un répertoire partagé qui est accessible aussi bien à l'appareil source que cible et référencé à partir de cet endroit, voir "Bases de données basées sur fichier" ci-dessous.

## Rendre les chemins portables

Si vous comptez exécuter le mappage sur un serveur, veuillez vous assurer que le mappage suive les conventions de chemin applicables et utilise une connexion de base de données prise en charge.

Pour rendre les chemins portables pour les systèmes d'exploitation non-Windows, utiliser des chemins relatifs et non pas absolus lors de la conception du mappage dans MapForce:

1. Ouvrir le fichier de design de mappage désiré (.mfd) avec MapForce sur Windows.
2. Dans le menu **Fichier**, choisir **Paramètre de mappage**, et décocher la case **Rendre les chemins absolus dans le code généré** si elle était cochée.
3. Pour chaque composant de mappage, ouvrir le dialogue **Propriétés** (en double-cliquant la barre de titre du composant, par exemple), et modifier tous les chemins de fichier d'absolus en relatifs. Ensuite, cocher la case **Enregistrer tous les chemins de fichier relatifs au fichier MFD**. Pour un plus grand confort, vous pouvez copier tous les fichiers d'entrée et les schémas dans le même dossier que le mappage lui-même, et les référencer uniquement par le nom de fichier.

Pour plus d'informations concernant la gestion des chemins relatifs et absolus pendant la conception des mappage, , voir Utiliser des chemins relatifs et absolus 46.

Chose importante, MapForce Server et FlowForce Server prennent en charge tous les deux un soit-disant "répertoire de travail" par rapport auquel tous les chemins relatifs seront résolus, voir aussi Chemins dans des environnements d'exécution variés 49. Le répertoire de travail est spécifié au moment de l'exécution du mappage, comme suit :

- Dans FlowForce Server, en éditant le paramètre "Répertoire de travail" de toute tâche.
- Dans MapForce Server API, par le biais de la propriété `WorkingDirectory` de l'API COM et .NET, ou par le biais de la méthode `setWorkingDirectory` de l'API.
- Dans la ligne de commande MapForce Server, le répertoire de travail est le répertoire actuel du shell de commande.

## Connexions de base de données

Sachez que les connexions ADO, ADO.NET et ODBC ne sont pas prises en charge sur les appareils Linux et macOS. Ainsi, si l'appareil cible est Linux ou macOS, ces connexions sont converties en JDBC lorsque vous déployez le mappage sur FlowForce ou lorsque vous compilez le mappage sur un fichier d'exécution MapForce Server. Dans ce cas, vous disposez des options suivantes avant de déployer le mappage ou de le compiler dans un fichier d'exécution de serveur :

- Dans MapForce, créer une connexion JDBC vers une base de données (voir Configurer une connexion JDBC 185).
- Dans MapForce, remplir les détails de connexion de la base de données JDBC dans la section "Paramètres spécifiques à JDBC" du composant de base de données (voir Paramètres de composant de base de données 256).

Si le mappage utilise une connexion native à une base de données PostgreSQL ou SQLite, la connexion native est préservée et aucune conversion JDBC n'aura lieu, voir Mappages de base de données dans plusieurs environnements d'exécution 159. Si le mappage se connecte à une base de données basée sur fichier, comme Microsoft Access et SQLite, une configuration supplémentaire est nécessaire, voir "Bases de données basée sur fichier" ci-dessous.

L'exécution de mappage avec des connexions JDBC exige que le Java Runtime Environment ou le Java Development Kit soit installé sur la machine du serveur. Il peut s'agir soit d'un Oracle JDK soit d'un build open source comme Oracle OpenJDK.

- La variable d'environnement `JAVA_HOME` doit pointer vers le répertoire d'installation JDK.

- Sur Windows, un chemin Java Virtual Machine trouvé dans le registre de Windows prendra la priorité sur la variable `JAVA_HOME`.
- La plateforme JDK (64-bit, 32-bit) doit être la même que celle de MapForce Server. Sinon, vous risquez d'obtenir une erreur : "JVM est inaccessible".

**Pour configurer une connexion JDBC sur Linux ou macOS:**

1. Télécharger le pilote JDBC fournit par le revendeur de base de données et l'installer sur le système d'exploitation. Veillez à choisir la version 32-bit si votre système d'exploitation fonctionne sur 32-bit, et la version 64-bit si votre système d'exploitation fonctionne sur 64-bit.
2. Définir les variables d'environnement à l'emplacement sur lequel le pilote JDBC est installé. Généralement, vous devrez définir la variable CLASSPATH, et éventuellement quelques autres variables. Afin de déterminer les variables d'environnement spécifiques que vous devez configurer, consulter la documentation fournie avec le pilote JDBC.

**Note** : Sur macOS, le système prévoit que toute bibliothèque JDBC installée se trouve dans le répertoire /Library/Java/Extensions. C'est pourquoi nous recommandons de déballer le pilote JDBC à cet emplacement ; sinon, vous devrez configurer le système pour chercher la bibliothèque JDBC dans le chemin dans lequel vous avez installé le pilote JDBC.

## Connexions Oracle Instant Client sur macOS

Ces instructions sont applicables si vous vous connectez à une base de données Oracle par le biais de Oracle Database Instant Client, sur macOS. Conditions préalables :

- Java 8.0 ou plus est installé. Si l'appareil Mac est exécuté avec une version Java précédente à Java 8, vous pouvez aussi vous connecter par le biais de la bibliothèque JDBC Thin for All Platforms et ignorer les instructions ci-dessous.
- Oracle Instant Client doit être installé. Vous pouvez télécharger l'Oracle Instant Client depuis la page de téléchargement officielle. Veuillez noter qu'il y a plusieurs packs Instant Client packages disponibles sur la page de téléchargement Oracle. Veuillez vous assurer de sélectionner un pack avec un prise en charge Oracle Call Interface (OCI), (par exemple, Instant Client Basic). De même, veillez à choisir la version 32-bit si votre système d'exploitation fonctionne sur 32-bit, et la version 64-bit si votre système d'exploitation fonctionne en 64-bit.

Une fois que vous avez téléchargé et déballé l'Oracle Instant Client, éditer le fichier de liste de propriété (.plist) envoyé avec le programme d'installation de manière à ce que les variables d'environnement suivantes pointent vers l'emplacement des chemins de pilote correspondants, par exemple :

| Variable | Valeur échantillon |
|---|---|
| `CLASSPATH` | `/opt/oracle/instantclient_11_2/ojdbc6.jar:/opt/oracle/instantclient_11_2/ojdbc5.jar` |
| `TNS_ADMIN` | `/opt/oracle/NETWORK_ADMIN` |
| `ORACLE_HOME` | `/opt/oracle/instantclient_11_2` |
| `DYLD_LIBRARY_PATH` | `/opt/oracle/instantclient_11_2` |

| | |
|---|---|
| PATH | $PATH:/opt/oracle/instantclient_11_2 |

**Note** : Éditer les valeurs d'échantillon ci-dessus pour correspondre aux chemins sur lesquels les fichiers Oracle Instant Client sont installés sur votre système d'exploitation.

## Bases de données basées sur fichier

Les bases de données basées sur fichier comme Microsoft Access et SQLite ne sont pas incluses dans le paquet déployé sur FlowForce Server ou dans le fichier d'exécution MapForce Server compilé. Ainsi, si l'appareil source et cible ne sont pas identiques, suivez les étapes suivantes :

1. Dans MapForce, cliquer avec la touche de droite sur le mappage et décocher la case **Rendre chemins absolus dans le code généré** (voir Changer les paramètres de mappage 79).
2. Cliquer avec la touche de droite sur le composant de base de données sur le mappage et ajouter une connexion au fichier de base de données en utilisant un chemin relatif , voir Configurer le chemin vers les Bases de données sur base de fichier 46. Un moyen simple d'éviter les problème liés au chemin est d'enregistrer le design de mappage (fichier .mfd) dans le même répertoire que le fichier de base de données et de référer dans ce dernier depuis le mappage uniquement par le nom de fichier (en utilisant donc un chemin relatif).
3. Copier le fichier de base de données dans un répertoire sur l'appareil cible (appelons-le "répertoire de travail"). Nous vous conseillons de garder ce répertoire en tête, il sera nécessaire pour exécuter le mappage sur le serveur, tel que montré ci-dessous.

Pour exécuter ce type de mappage sur le serveur, suivre une des étapes suivantes :

- Si le mappage est exécuté par MapForce Server sous le contrôle de FlowForce Server, configurer la tâche FlowForce Server pour pointer vers le répertoire de travail créé précédemment. Le fichier de base de données doit résider dans le répertoire de travail. .
- Si le mappage est exécuté par le MapForce Server autonome dans la ligne de commande, changer le répertoire actuel en répertoire de travail (par exemple, `cd path\to\working\directory`) avant d'appeler la commande `run` de MapForce Server.
- Si le mappage est exécuté par l'API MapForce Server, définir le répertoire de travail par programme avant d'exécuter le mappage. Pour faciliter la tâche, la propriété `WorkingDirectory` est disponible pour l'objet MapForce Server dans l'API COM et .NET. Dans l'API Java, la méthode `setWorkingDirectory` est disponible.

Si les appareils source et cible sont des appareils Windows exécutés sur le réseau local, une autre approche est de configurer le mappage pour lire le fichier de la base de données depuis un répertoire partagé commun, comme suit :

1. Stocker le fichier de base de données dans un répertoire qui est accessible aussi bien par l'appareil source que cible.
2. Cliquer avec la touche de droite sur le composant de base de données sur le mappage et ajouter une connexion au fichier de base de données en utilisant un chemin absolu (voir Configurer le chemin vers les Bases de données sur base de fichier 46).

## Ressources globales

Si un mappage inclut les références aux Ressources globales au lieu des chemins directs ou des connexions

de base de données, vous pourrez également utiliser des Ressources globales du côté serveur. Lorsque vous compilez un mappage dans un fichier d'exécution MapForce Server (.mfx), les références aux Ressources globales resteront intactes afin que vous puissiez les fournir du côté serveur, lors de la marche du mappage. Lorsque vous déployez un mappage vers FlowForce Server, vous pouvez choisir en option s'il doit utiliser des ressources sur le serveur.

Pour que les mappages (ou des fonctions de mappage, dans le cas de FlowForce Server) puissent être exécutés avec succès, le fichier lui-même, le dossier ou les détails de connexion de base de données que vous fournissez en tant que Ressources globales doivent être compatibles avec le nouvel environnement de serveur. Par exemple, les chemins de fichier et de dossier doivent utiliser la convention Linux pour des chemins si le mappage sera exécuté sur un serveur Linux. De même, les Ressources globales définies en tant que connexions de base de données doivent être possibles sur l'appareil du serveur.

Pour plus d'informations, voir Ressources globales dans MapForce Server 899 et Ressources globales dans FlowForce Server 899..

# 10.4 Compiler des mappages sur des fichiers d'exécution MapForce Server

Lorsque la langue cible d'un mappage créé dans MapForce est défini sur BUILT-IN, celui-ci peut être exécuté non seulement par MapForce, mais aussi par MapForce Server (voir À propos de MapForce Server 857). Il existe deux moyens d'exécuter un mappage avec MapForce Server:

- Si MapForce Server est exécuté en mode autonome (donc si aucun FlowForce Server n'est installé), le mappage doit être compilé dans un fichier d'exécution de serveur (.mfx), comme indiqué ci-dessous. Vous pouvez ensuite exécuter le fichier .mfx au niveau de la ligne de commande, en utilisant la commande `run`. Vous pouvez aussi exécuter le mappage en invoquant la méthode `run` de l'API MapForce Server. Pour plus d'informations, voir la documentation MapForce Server (https://www.altova.com/fr/documentation).
- Si MapForce Server est exécuté sous la gestion de FlowForce Server, le mappage peut être déployé sur un appareil sur lequel sont exécutés MapForce Server et FlowForce Server (voir Déployer des mappages sur FlowForce Server 867).

## Conditions préalables

Voir Préparer des mappages pour l'exécution de serveur 858.

### Pour compiler un mappage dans un fichier MapForce Server Execution (.mfx) :

1. Ouvrir un mappage dans MapForce (par exemple, **myMapping.mfd**).
2. Dans le menu **Fichier**, cliquer sur **Compiler dans fichier MapForce Server Execution**.
3. Choisir le dossier dans lequel vous souhaitez placer le fichier .mfx et changer le nom du fichier le cas échéant.
4. Cliquer sur **Enregistrer**. Le fichier MapForce Server Execution **myMapping.mfx** est généré dans le dossier choisi.

### Pour compiler un mappage dans un fichier MapForce Server Execution (.mfx), en utilisant la ligne de commande :

- Exécuter MapForce au niveau de la ligne de commande et spécifier le fichier de mappage et l'option de ligne de commande `/COMPILE`.

Par exemple, la commande suivante compile le mappage **C:\Users\altova\Documents\Altova\MapForce2024\MapForceExamples\SimpleTotal.mfd** dans un fichier d'exécution MapForce Server qui sera créé dans le répertoire de sortie cible **C:\Users\altova\Desktop**.

```
"C:\Program Files (x86)\Altova\MapForce2024\MapForce.exe" "C:
\Users\altova\Documents\Altova\MapForce2024\MapForceExamples\SimpleTotal.mfd" /COMPILE
"C:\Users\altova\Desktop"
```

Voir aussi l'Interface de ligne de commande MapForce 876.

## Qu'est-ce qui est inclus dans le fichier .mfx

Le fichier .mfx comprend les données suivantes :

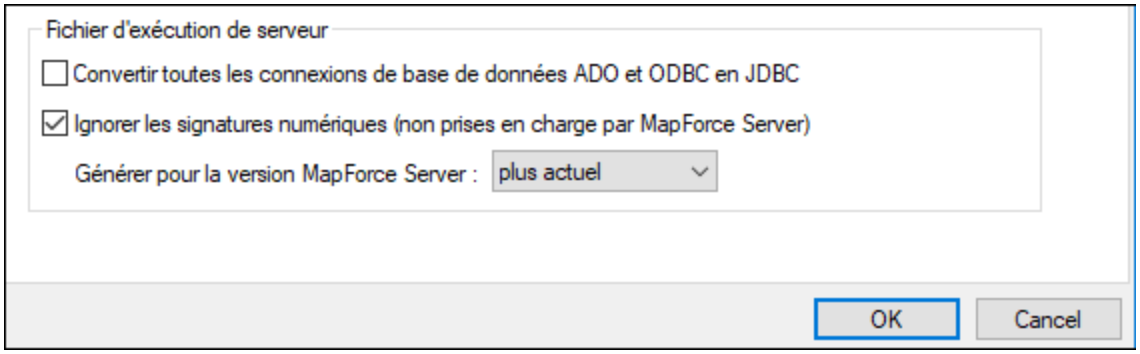
- L'algorithme de mappage, qui comprend toutes les fonctions définies par l'utilisateur (FDUs) importées depuis les autres mappages.
- Les noms de fichier d'entrée et de sortie référencés depuis des composants. Les chemins sont absolus ou relatifs, selon les paramètres du mappage, voir Chemins dans des environnement d'exécution différents 49.
- Si le mappage contient des composants XML, les informations nécessaires concernant le schéma XML pour pouvoir exécuter le mappage sont encodées dans l'algorithme de mappage.
- Les détails de connexion de la base de données, si le mappage contient des connexions à la base de données. Les mots de passe sont chiffrés.

Les fichiers d'instance d'entrée (XML, CSV, Text) qui sont utilisés par le mappage ne sont pas inclus dans le fichier .mfx compilé. La même chose vaut pour les bases de données à base fichier comme Access ou SQLite. Pour plus de détails, voir Préparer des mappages pour l'exécution de serveur 858.

## Compiler des mappages pour une version MapForce Server spécifique

Si votre MapForce Server est une version plus ancienne que celle de MapForce, elle risque de ne pas pouvoir exécuter les fichiers .mfx créés avec une version plus récente de MapForce, étant donné que des nouvelles fonctions auront probablement été ajoutées entre-temps. Dans ces cas, vous pouvez compiler le fichier .mfx pour une version spécifique de MapForce Server, comme suit :

1. Dans le menu **Outils**, cliquer sur **Options**, puis cliquer sur **Génération**.
2. Sous **Fichier d'exécution de serveur**, à côté de **Générer pour la version MapForce**, choisir la version MapForce Server requise depuis la liste déroulante.




Si vous possédez une version de MapForce Server plus récente, n'oubliez pas de modifier cette option. Si vous n'avez pas de raison particulière pour compiler sur une version spécifique de MapForce Server, sélectionner l'option la "plus actuelle" (il s'agit de l'option par défaut). Lorsque cette option est choisie, le fichier .mfx est compilé pour la version la plus récente de MapForce Server et peut profiter des toutes dernières fonctions et améliorations qui pourraient ne pas être disponibles dans les versions plus anciennes.

Pour spécifier une version MapForce Server cible dans la ligne de commande, exécuter la commande `/COMPILE` avec `/MFXVERSION`, par exemple :

```
"C:\Program Files (x86)\Altova\MapForce2024\MapForce.exe" /COMPILE /MFXVERSION:2024
```

Voir aussi l'Interface de ligne de commande MapForce[876].

## Autres options

La compilation des Fichiers d'exécution de MapForce est aussi concernée par les options suivantes :

| | |
|---|---|
| *Convertir toutes les connexions de base de données ADO et ODBC sur JDBC* | Si cette option est activée, les connexions de base de données ADO, ADO.NET et ODBC sont transformées en JDBC en utilisant le pilote JDBC et l'URL de base de données définie dans le dialogue Paramètres de composant de base de données (voir Paramètres de composant de base de données[256]).<br><br>La connexion JDBC sera utilisée implicitement si l'appareil cible est un serveur Linux ou macOS. |
| *Ignorer les signatures numériques (non prises en charge par MapForce Server)* | Cette option est uniquement applicable pour MapForce Enterprise. Elle est activée par défaut. Si le mappage utilise des signatures numériques XML, il saute l'information de signature numérique, puisque MapForce Server ne prend pas en charge des signatures numériques XML. |

Pour consulter ou modifier ces options :

- Dans le menu **Outils**, cliquer sur **Options**, puis cliquer sur **Génération**.

Ces options sont aussi disponibles depuis l'interface de ligne de commande. Voir aussi l'Interface de ligne de commande MapForce[876].

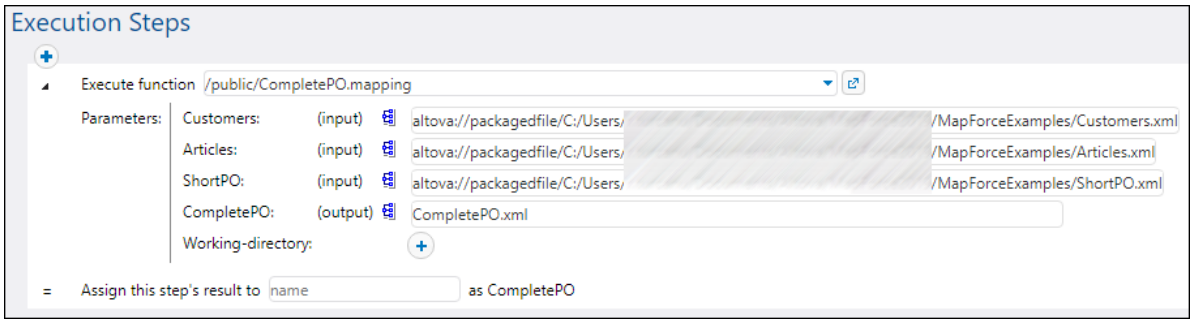
# 10.5 Déployer des mappages sur FlowForce Server

Le fait de déployer un mappage sur FlowForce Servers signifie que MapForce organise les ressources utilisées par le mappage spécifique dans un objet et les passe par HTTP (ou HTTPS si configuré) sur l'appareil sur lequel FlowForce Server est exécuté. Les mappages MapForce sont généralement déployés sur FlowForce Server pour automatiser leur exécution par le biais des tâches FlowForce Server. Une fois qu'un mappage est déployé, vous pouvez créer une tâche FlowForce Server contenant toutes les fonctions que vous souhaitez, et profiter de toutes les fonctions spécifiques à la tâche (par exemple, définir des conditions de déclenchement personnalisées pour la tâche, l'exposer en tant que service Web, etc.).

**Note :** Le terme "appareil source" se réfère à l'ordinateur sur lequel MapForce est installé et le terme "appareil cible" se réfère à l'ordinateur sur lequel FlowForce Server est installé. Dans le scénario le plus simple, il s'agit du même ordinateur. Dans des scénarios plus avancés, MapForce est exécuté sur un appareil Windows alors que FlowForce Server est exécuté sur un appareil Linux ou macOS.

Le pack déployé sur FlowForce contient les éléments suivants :

- Le mappage lui-même. Après déploiement, le mappage devient disponible dans l'interface d'administration FlowForce Server en tant que fonction de mappage (.mapping), sous le chemin que vous spécifiez. Tout composant de source devient des arguments d'entrée, et tous les composants de cible deviennent des arguments de sortie de cette fonction.



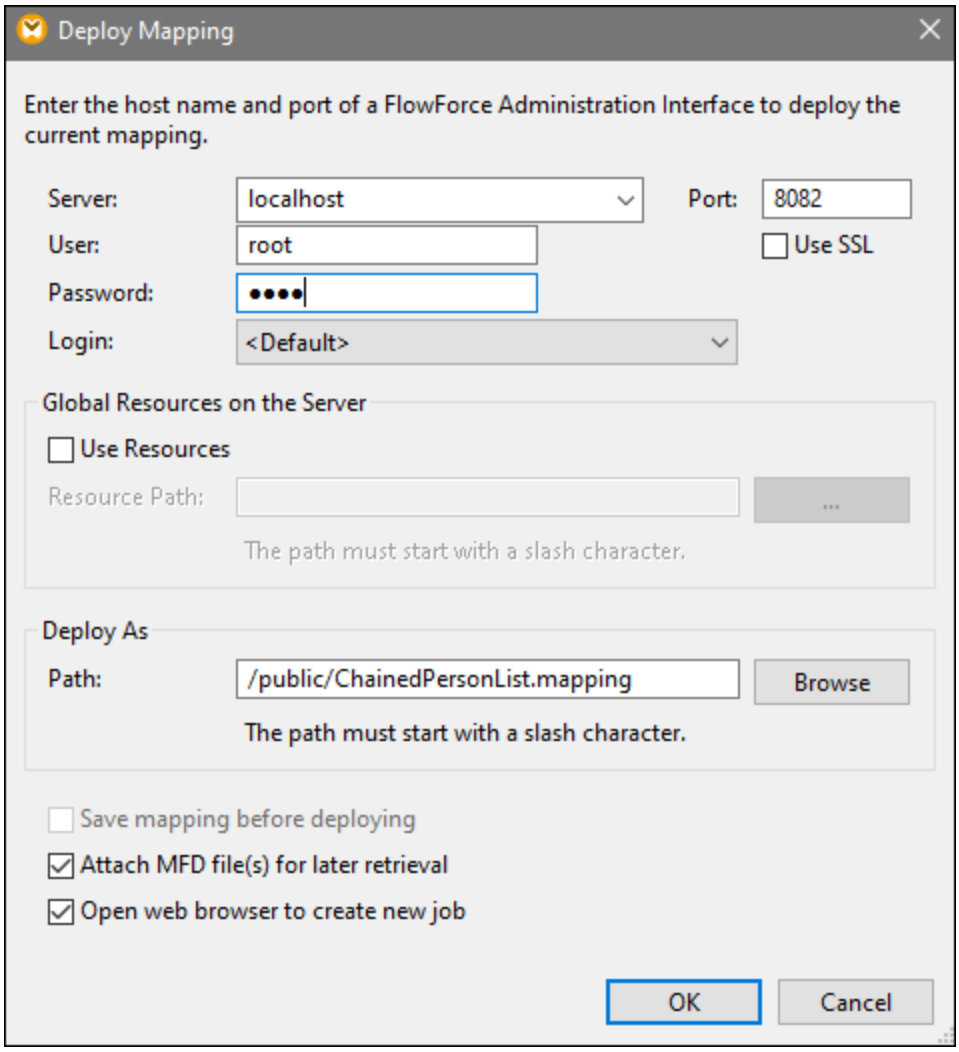

- Toute sorte de fichiers d'instance d'entrée (XML, CSV, Text) qui sont utilisés par le mappage.

## Prérequis

Voir Préparer des mappages pour l'exécution de serveur 858.

## Déployer le mappage sur FlowForce Server

1. Exécutez MapForce et assurez-vous que le langage de transformation est défini comme Built-In.
2. Dans le menu **Fichier**, cliquez sur **Déployer sur FlowForce Server**. Le dialogue **Déployer mappage** s'ouvre (*voir ci-dessous*).




3. Saisissez vos paramètres de déploiement (comme décrit ci-dessous et cliquez sur OK. Si vous avez coché la case *Ouvrir navigateur pour créer une nouvelle tâche*, l'interface d'administration FlowForce Server s'ouvre dans le navigateur, et vous pouvez commencer à créer une tâche FlowForce Server immédiatement.

La table ci-dessous liste les paramètres de - déploiement de mappage disponibles dans le dialogue **Déployer mappage**.

| Paramètre | Description |
|---|---|
| Serveur, Port et Utiliser SSL | Saisir le nom d'hôte du serveur (ou adresse IP) et le port de FlowForce Server. Ceux-ci pourraient être **localhost** et **8082** si FlowForce Server est exécuté sur le même appareil au port par défaut. En cas de doute, se connecter sur l'interface d'administration FlowForce Server Web et contrôler l'adresse IP et le port affiché dans la barre d'adresse du navigateur Web.<br><br>Si vous êtes confronté à des erreurs de connectivité, s'assurer que |

| Paramètre | Description |
|---|---|
| | l'appareil sur lequel FlowForce Server est exécuté est configuré pour permettre des connexions entrantes sur l'adresse et le port désigné.<br><br>Pour déployer le mappage par le biais d'une connexion chiffrée par SSL, cocher la case **Utiliser SSL**. Cela part du principe que FlowForce Server est déjà configuré pour accepter les connexions SSL. Pour plus d'informations, veuillez vous référer à la documentation FlowForce Server (https://www.altova.com/fr/documentation). |
| Utilisateur et mot de passe | Le nom d'utilisateur et le mot de passe à saisir dépendent de la valeur de la liste déroulante de Login (voir option suivante). Si la liste déroulante de Login est définie sur **<Default>** ou sur **Directement**, saisir votre nom d'utilisateur et mot de passe FlowForce Server. Sinon, saisir votre nom d'utilisateur et votre mot de passe de domaine et choisir le nom de domaine depuis la liste déroulante Login. |
| Connexion | Si « Directory Service » est activé dans FlowForce Server, choisir le nom de domaine depuis cette liste déroulante, et saisir vos identifiants de domaine dans les champs Utilisateur et Mot de passe (voir des options précédentes). |
| Utiliser des Ressources, Chemin de Ressource | Sélectionnez la case à cocher **Utiliser Ressources** si la fonction du mappage doit utiliser Ressources 881 après l'avoir déployé sur le serveur. Si vous cochez la case, vous devez aussi saisir le chemin de la ressource respective dans le champ de saisie **Chemin de ressource**. Pour sélectionner la ressource, cliquer sur la touche **Ellipse**.<br><br>S'il n'y a pas encore de ressources à sélectionner sur le serveur, cliquer sur **Déployer Ressources Globales**, et déployez la Ressource Globale requise sur le serveur. Pour plus d'informations, voir Déployer des Ressources Globales sur FlowForce Server 900.<br><br>Si vous ne cochez pas la case **Utiliser des ressources**, toute Ressource Globale sera résolue, sur la base de la configuration actuellement choisie. Sur le serveur, la fonction de mappage ne nécessitera plus de Ressources Globales, mais utilisera la valeur résolue à la place. |
| Chemin | Cliquez sur **Navigateur**, et sélectionnez le chemin dans lequel la fonction de mappage doit être enregistrée dans la hiérarchie du conteneur de FlowForce Server. Par défaut, le chemin est défini sur le conteneur **/public** de FlowForce Server.<br><br>Depuis le dialogue, vous pouvez aussi créer de nouveaux conteneurs ou supprimer les conteneurs et les mappages existants, pourvu que vous déteniez les permissions et les privilèges de FlowForce Server requis. |

| Paramètre | Description |
|---|---|
| Enregistrer le mappage avant le déploiement | Cette option est disponible si vous déployez un mappage non enregistré. Cocher cette case pour enregistrer le mappage avant le déploiement. |
| Joindre des fichiers MFD pour retrait ultérieur | Cette option vous permet de déployer le fichier MFD ensemble avec les fichiers d'entrée dépendants (par ex., fichier(s) XML source), à l'exception des fichiers définissant la structure (par ex., schémas XSD). Lorsque vous ouvrez le mappage déployé dans FlowForce Server, la section *Fichiers déployés* recensera tous les fichiers que vous pouvez télécharger. |
| Ouvrir le navigateur pour chercher une nouvelle tâche | Si vous cochez cette case, l'interface d'administration FlowForce Server Web s'ouvre dans le navigateur après le déploiement et vous pouvez commencer à créer une tâche FlowForce Server immédiatement. |

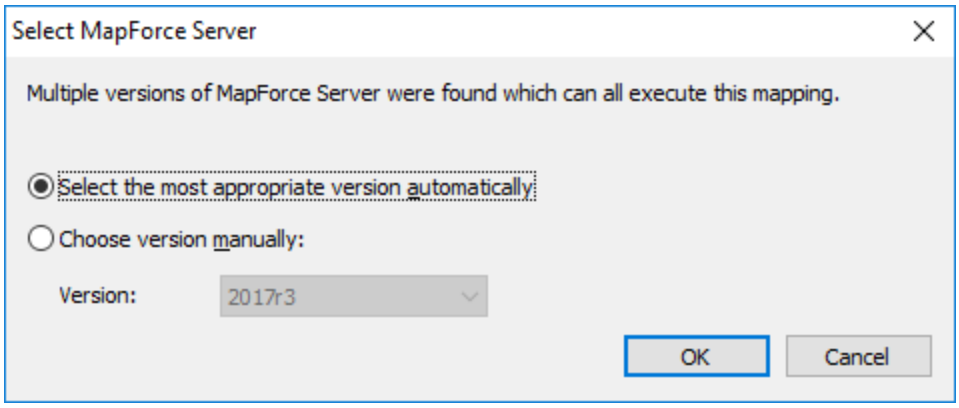
## Dépannage

Le tableau suivant regroupe les problèmes qui risque de survenir lors du déploiement d'un mappage, suivi des solutions.

| Problème | Solution |
|---|---|
| Le déploiement du mappage retourne l'erreur suivante :<br><br>`I/O operation on file ... failed.`<br>`I/O Error 28: Failed to connect to <server> port 8082. Timed out`<br>`System error 10060: Un essai de connexion a échoué car la partie connectée n'a pas répondu après une période de temps ou à établi une connexion échouée car l'hôte connecté n'a pas répondu.` | Assurez-vous que sur l'appareil cible, le service *FlowForce Web Server* est en cours d'exécution et est configuré pour écouter les connexions sur le port spécifié (**8082**, par défaut). De même, veuillez vous assurer que le pare-feu ne bloque pas des connexions entrantes par ce port.<br><br>Le service *FlowForce Server* doit être exécuté pour que le déploiement soit possible. |
| Le déploiement du mappage retourne l'erreur suivante :<br><br>`I/O operation on file ... failed.`<br>`I/O Error 413: Payload Too Large` | Cette erreur peut apparaître si le fichier d'entrée du mappage déployé dépasse la limite de taille maximum des requêtes HTTP autorisées par FlowForce Server (environ 100 MB). Vous pouvez augmenter la limite en définissant l'option `max_request_body_size` (en octets) dans les fichiers **flowforceweb.ini** et **flowforce.ini**. Pour plus de détails, voir la documentation FlowForce Server. |

## Sélectionner la version de serveur (uniquement Windows)

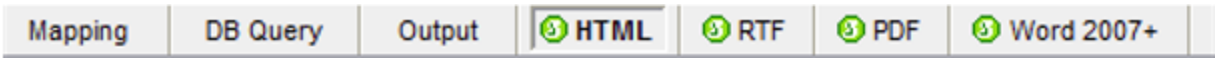
Si le serveur sur lequel vous déployez le mappage a plusieurs versions de MapForce Server exécutées sous la gestion de FlowForce Server (uniquement applicable sur des serveurs Windows), alors vous serez également invité à spécifier la version de MapForce Server avec laquelle vous souhaitez que ce mappage soit exécuté.




**Note :** Le dialogue apparaît lorsque le répertoire d'installation de FlowForce Server contient des fichiers .tool pour chaque version MapForce Server exécutée sous gestion FlowForce Server. Par défaut, un fichier MapForce Server .tool est ajouté automatiquement dans ce répertoire lorsque vous installez MapForce Server sous forme d'installation FlowForce Server. Le chemin sous lequel les fichiers .tool sont stockés dans FlowForce est : **C:\Program Files\Altova\FlowForceServer2024\tools**. Si vous utilisez des versions supplémentaires de MapForce Server que vous souhaitez faire fonctionner sous une gestion FlowForce Server, leurs fichiers .tool peuvent être copiés manuellement dans le répertoire ci-dessus. Le fichier .tool de MapForce Server peut être trouvé sous : **C:\Program Files\Altova\MapForceServer2024\etc**.

# 10.6 Volets de Sortie StyleVision

Dans les mappages où le composant cible est XML , il est possible de consulter et enregistrer la sortie de mappage en tant qu'HTML, RTF, PDF, Word 2007+, et les documents texte si Altova StyleVision est installée sur votre ordinateur. Si vous utilisez l'édition Enterprise de StyleVision, les graphiques seront aussi rendus dans ces aperçus. Lorsqu'un mappage prend en charge la consultation dans l'un de ces formats, des volets supplémentaires deviennent disponibles à côté du volet **Sortie** (*voir la capture d'écran ci-dessous*).




**Important**

- Si StyleVision Professional est installé, il est possible de voir des sorties **HTML**, **RTF** et **Texte**.
- Avec StyleVision Enterprise, il est possible d'apercevoir la sortie **HTML**, **RTF**, **PDF** et **Word 2007+**.
- La consultation de la sortie de mappage en tant que PDF exige Java, Acrobat Reader, et FOP (Formatting Objects Processor) version 0.93 ou 1.0. FOP est installé avec StyleVision, à moins que vous ayez choisi de ne pas l'installer pendant l'installation de StyleVision.
- Dans l'édition 64-bit de MapForce, les aperçus Word 2007+ et RTF sont ouverts en tant qu'applications non intégrées.
- Si votre mappage contient des composants qui agissent à la fois en tant que sources et cibles (composants pass-through), la consultation StyleVision sera possible uniquement pour les composants où le bouton **Preview** du composant a été activé. Pour plus d'informations concernant ce type de mappages, voir Mappages en chaîne 105.

Pour consulter vos données de mappage dans les volets de sortie StyleVision, l'étape suivante est requise :

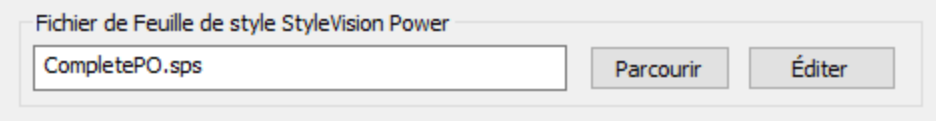
- Altova StyleVision doit être installée sur votre ordinateur soit en tant qu'installation standalone ou faisant partie de Altova MissionKit.
- Le composant cible doit avoir un fichier StyleVision Power Stylesheet (SPS) qui y est associé. Le fichier de feuille de style peut être créé ou édité avec StyleVision. Vous ne pouvez pas éditer ou modifier la feuille de style directement dans MapForce, mais vous pouvez l'ouvrir par le biais de MapForce dans StyleVision. Une fois que la feuille de style est prête, vous pouvez l'assigner à un composant MapForce cible, comme indiqué ci-dessous.

## Configuration des volets de sortie StyleVision

Les instructions ci-dessous vous aideront à définir les volets de sortie StyleVision.

*Attribuer une feuille de style StyleVision Power à un composant cible*
Pour attribuer un fichier SPS à un composant cible, suivez les étapes suivantes :

1. Dans StyleVision, créez le fichier de feuille de style requis. Assurez-vous d'utiliser le même schéma XML en tant que source que pour le composant MapForce.
2. Dans MapForce, cliquez avec la touche de droite sur le composant XML et sélectionnez **Propriétés**.
3. Dans le dialogue **Paramètres de composant**, situé à côté du **fichier StyleVision Power Stylesheet**, cherchez le fichier de feuille de style créé précédemment (*voir la capture d'écran ci-dessous*).




**Note :**le chemin vers le fichier StyleVision Power Stylesheet peut être absolu ou relatif. Pour des détails, voir aussi Chemins relatifs et absolus [47].

*Enregistrer la sortie générée par StyleVision*
Vous pouvez enregistrer la sortie générée par StyleVision sur un fichier de la même manière que pour enregistrer le résultat de tout autre mappage. Cliquez sur le bouton de la barre d'outils (**Enregistrer sortie générée**) ou allez au menu de **Sortie** et cliquez sur **Enregistrer fichier de sortie**.

*Automatiser la génération de différents formats avec des produits Altova*
Si vous voulez que votre mappage génère HTML, PDF, RTF, Word 2007+ et des fichiers de texte automatiquement (soit sur le même ordinateur, soit sur un autre ordinateur ou même une plateforme), vous pouvez utiliser Serveur MapForce ou StyleVision Server. Il s'agit de produits de serveur séparément sous licence qui élargissent respectivement la fonctionnalité de MapForce et StyleVision. Dans ce scénario, chaque application joue son propre rôle spécifique :

- MapForce vous permet de concevoir un mappage (**.mfd**) qui définit les entrées et sorties de transformation de données (par exemple, base de données à XML)
- Le serveur MapForce exécute le mappage exécutable (**.mfx**) depuis la ligne de commande ou depuis un API (soit dans le même système d'exploitation, soit dans un autre).
- StyleVision vous permet de concevoir la feuille de style (**.sps**) requise pour transformer la sortie de mappage en HTML, PDF, RTF, Word 2007+ et les fichiers de texte.
- Le serveur de StyleVision Server exécute la feuille de style **.sps** qui transforme la sortie de mappage en un format désiré. Ceci est réalisé dans la ligne de commande ou depuis un API (sur le même système d'exploitation ou différent).
- Les deux serveurs StyleVision Server et MapForce peuvent optionnellement être exécutés sous la gestion du Serveur FlowForce (sous licence séparément). Dans ce scénario, les mappages MapForce et transformations StyleVision peuvent être exécutés comme prévus, déclenchés ou des tâches sur demande. Ceci signifie que les mappages MapForce et transformations StyleVision peuvent être entièrement automatisés.

## Exemples

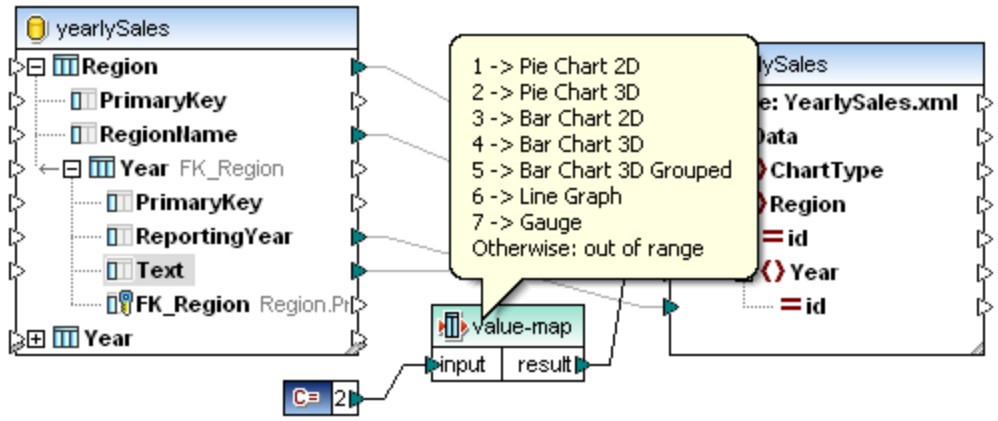
L'exemple ci-dessous (**MapForceExamples\CompletePO.mfd**) affiche la sortie dans le volet de sortie de StyleVision appelé **HTML**. Ce mappage produit une commande d'achat en format XML. Cliquez avec la touche de droite sur le composant cible, sélectionnez **Propriétés**, et constatez qu'un fichier **.sps** lui a été attribué.

Si vous cliquez sur le volet **HTML**, vous verrez la sortie suivante :


Un autre exemple est **`Tutorial\YearlySales.mfd`**. La feuille de style attribuée à ce mappage a été conçue dans StyleVision de manière à ce qu'il soit possible de contrôler le type de graphique en changeant la valeur de l'élément `ChartType`. Cela permet de changer le type de graphique directement depuis le mappage : Vous pouvez changer la valeur par défaut de la constante sur n'importe quelle valeur allant de 1 à 7. Si vous placez le

curseur de la souris sur le composant `value-map`, vous verrez les valeurs possibles (*voir la capture d'écran ci-dessous*).




La valeur par défaut de la constante est 2, ce qui génère un graphique camembert 3D dans la sortie. Pour afficher d'autres types de graphiques, changez cette valeur en une autre valeur autorisée et cliquez sur l'onglet **Sortie** pour voir les modifications.

# 10.7 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. Pour générer le code pour tout le projet, définir la cible `/GENERATE` (*voir /{target} ci-dessous*) et saisir le chemin du projet comme `<filename>` (par ex., **`MapForceExamples.mfp`**).

- `/{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`
  - La commande `/COMPILE[:compileoptions]` compile un mappage vers un fichier d'exécution de MapForce Server (**`.mfx`**). Vous pouvez aussi fournir les options suivantes séparées par une virgule :

    - L'option `JDBC` transforme toutes les connexions de bases de données en JDBC utilisant le pilote JDBC et l'URL de base de données défini dans le dialogue Paramètres de composant de base de données 256.
    - L'option `NOXMLSIGNATURES` masque la génération de signatures numériques dans le fichier d'Exécution de MapForce Server. Notez que les signatures numériques ne sont pas appuyées par MapForce Server.

- La commande `/GENERATE` génère un code de projet pour tous les mappages dans le fichier de projet en utilisant les paramètres de dossier actuel (voir Gérer les dossiers de projet[86]). Si vous choisissez cette cible, assurez-vous de fournir un projet MapForce (**`.mfp`**) en tant que `<filename>`.
- `/XQuery`
- `/JAVA`
- La commande `/CS` génère un code C#. Cette commande permet de définir les options de génération de code suivants :

  `/CS[:{VS2013|VS2015|VS2017|VS2019|VS2022|DOTNETCORE31|DOTNET50|DOTNET60}]`

  Si aucune version de Visual Studio n'est spécifiée, le code sera généré en utilisant la version Visual Studio définie dans les options de génération de code, voir options de génération de code.

- La commande `/CPP` génère un code C++ Cette commande permet de définir les options de génération de code suivants :

  `/CPP[:{VS2013|VS2015|VS2017|VS2019|VS2022|DOTNETCORE31|DOTNET50|DOTNET60},`
  `{MSXML|XERCES3},{LIB|DLL},{MFC|NoMFC}]`

  Le premier groupe d'options précise l'environnement de développement (par ex., `VS2022` signifie Visual Studio 2022).

  Le deuxième groupe d'options spécifie la bibliothèque XML ciblée par le code généré. Les valeurs suivantes sont valides :

  o `MSXML` (génère du code pour MSXML 6.0)
  o `XERCES3` (génère un code pour Xerces 3)

  Le troisième groupe des options spécifie si les bibliothèques statiques par rapport aux bibliothèques dynamiques doivent être générées. Les valeurs valides incluent ce qui suit :

  o `LIB` (génère des bibliothèques statiques LIB)
  o `DLL` (génère des bibliothèques DLL)

  Le quatrième groupe d'options spécifie si le code doit être généré avec ou sans prise en charge MFC. Les valeurs valides incluent ce qui suit :

  o `MFC` (active la prise en charge MFC)
  o `NoMFC` (désactive la prise en charge MFC)

  Si les options ci-dessous ne sont pas spécifiées, le code sera généré en utilisant la version Visual Studio définie dans les options de génération de code.

`<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.

Lorsque la cible est `/GENERATE` et le paramètre `<outputdir>` n'est pas défini, le langage de génération de code et le chemin de sortie de chaque mappage sont fournis par les paramètres définis pour chaque dossier à l'intérieur du projet (voir Gérer Dossier de projet[86]).

Lorsque la La section cible est `/GENERATE` et que le paramètre `<outputdir>` est défini, la valeur `<outputdir>` fournie au niveau de la ligne de commande prend précédence sur le directoire de sortie définie au niveau du projet root. Néanmoins elle ne prend pas précédence sur les paramètres de génération de code définis dans chaque dossier dans le projet.

### `/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[881].
  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.

- L'option `[/MFXVERSION[:<version>]` est applicable si la cible est `/COMPILE`. Cette option compile le fichier MapForce Server Execution (`.mfx`) pour une version particulière de MapForce Server. Vous pouvez fournir toute version de MapForce Server, commençant par 2013r2 et supérieure. Voir aussi Compiler des mappages pour une version MapForce Server spécifique[865].

- L'option `/LIBRARY <libname> (...)` est utilisée ensemble avec un langage cible de génération de code pour spécifier des bibliothèques de fonctions supplémentaires. Cette option peut être spécifiée plus d'une fois pour charger plusieurs bibliothèques. Voir aussi Gérer les Bibliothèques de fonction[471].

*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`**, nom de fichier **`.mfp`** 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 398).

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>
```

# 11 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. Dans MapForce, vous pouvez aisément extraire des données soit de la base de données de `développement` ou de `production`, en choisissant la configuration souhaitée à partir d'une liste déroulante avant de consulter le mappage.

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 :

# 11.1 Configuration des Ressources globales, Partie 1

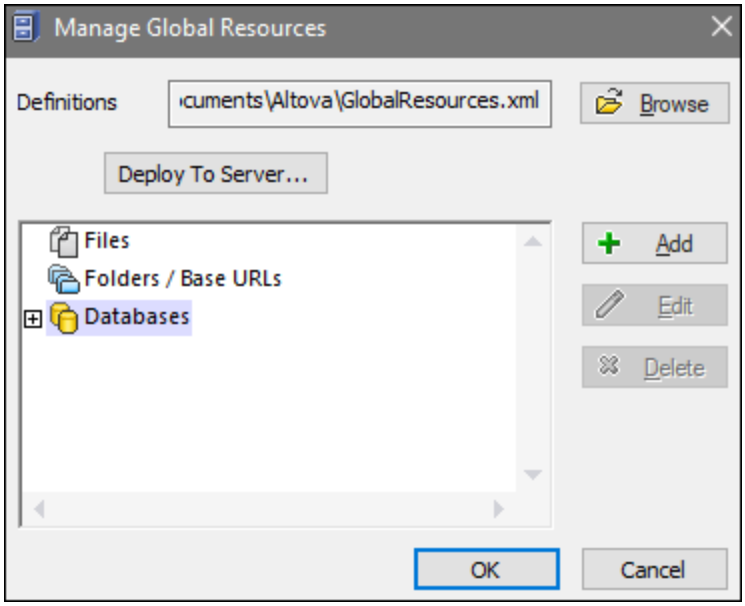
La configuration des Ressources globales a lieu en deux parties : (i) créer une ressource globale dans la boîte de dialogue **Gérer les Ressources globales** (*voir ci-dessous*) et (ii) définir les propriétés de cette ressource globale dans le dialogue **Ressource globale**. La deuxième partie est discutée dans la prochaine rubrique 884.

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.

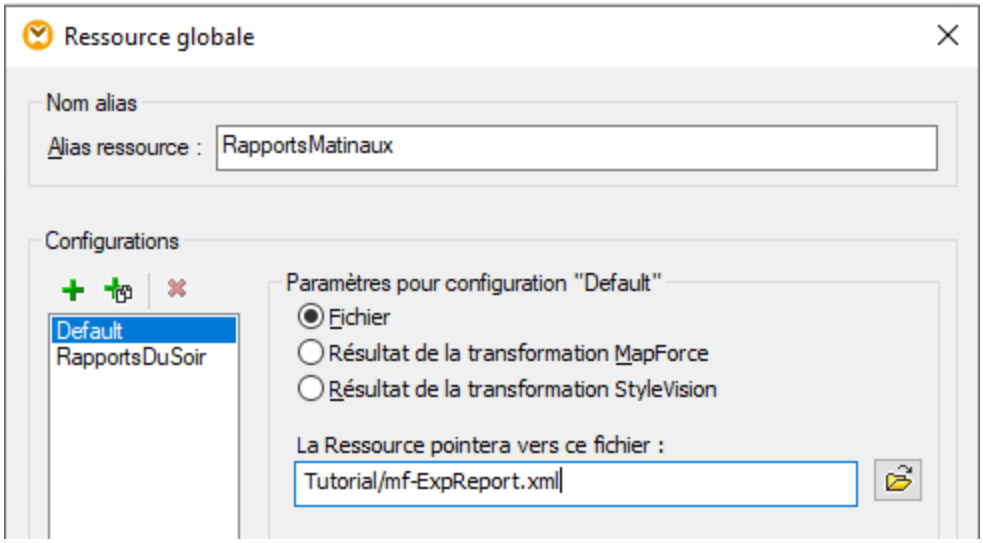
# 11.2 Configuration des Ressources globales, Partie 2

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 882.

Pour en savoir plus sur la configuration des différents types de ressources globales, voir les exemples suivants : Fichiers XML en tant que Ressources globales 888, Dossiers en tant que Ressources globales 890, Bases de données en tant que Ressources globales 892.

## Fichiers

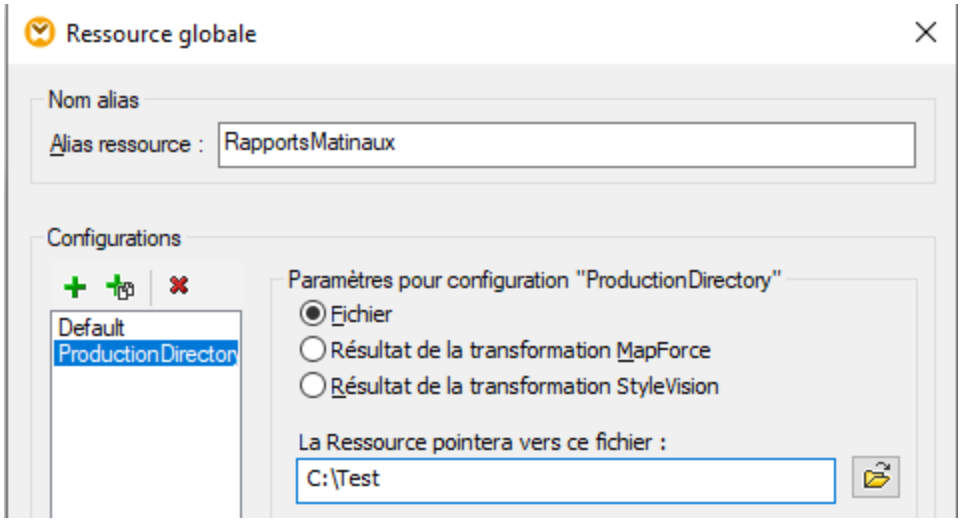
Les propriétés spécifiques au fichier sont affichées dans la boîte de dialogue **Ressource globale** ci-dessous. La configuration est divisée en trois parties majeures : (i) le nom du fichier, (ii) l'emplacement de ce fichier, et (iii) la liste de configurations définies dans l'alias du fichier.




Les paramètres *Résultat de la transformation de MapForce* et *Résultat de la transformation de StyleVision* sont discutés dans Résultats de transformation en tant que Ressources globales 894.

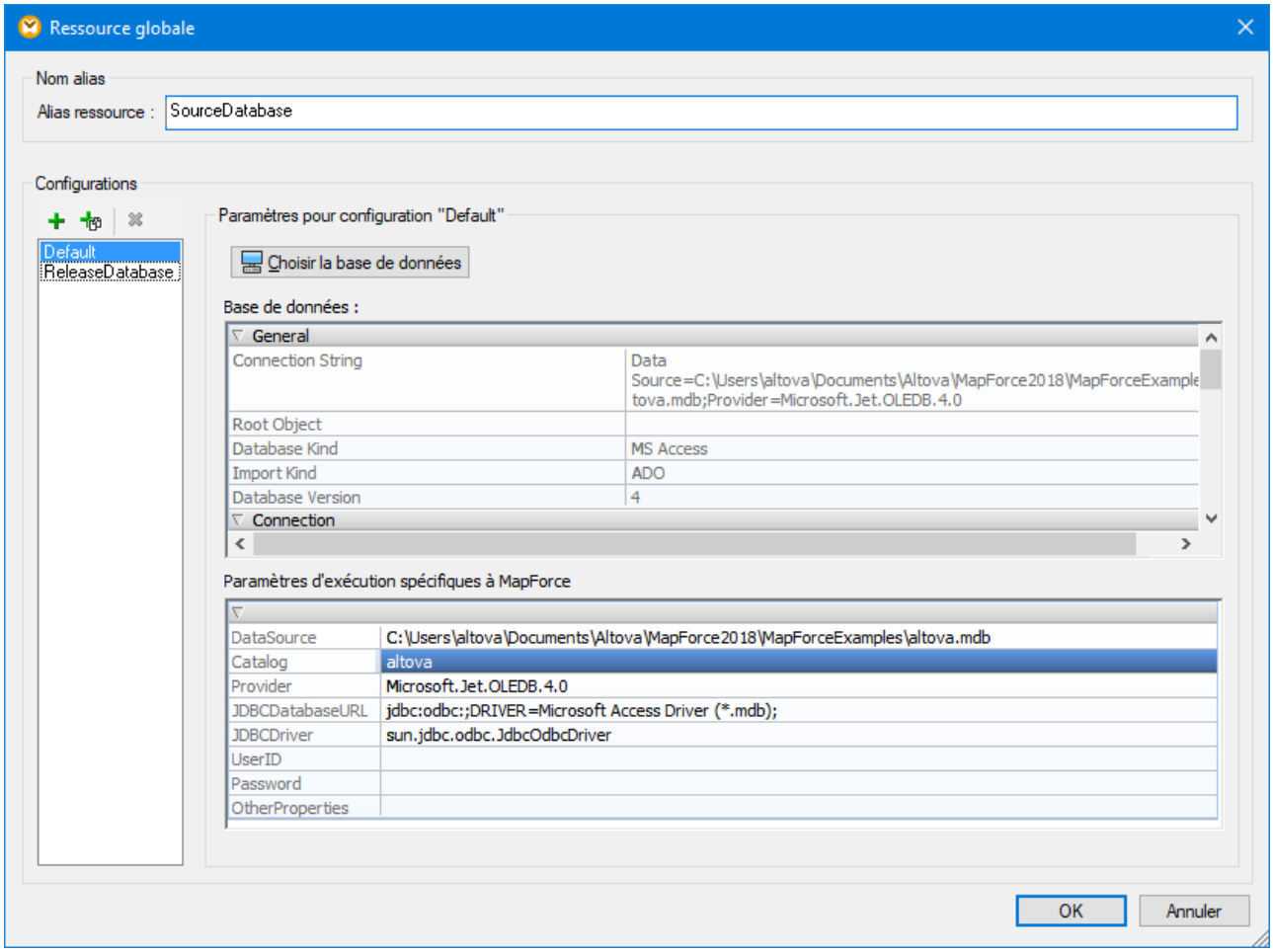
## Dossiers

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.




## Bases de données

Lorsque vous ajoutez une connexion de base de données en tant que Ressource globale, un assisant à la connexion vous guide à travers les étapes requises pour configurer la connexion. Pour plus d'informations, voir Démarrer l'assistant de connexion à la base de données 163. Une fois que l'assistant a terminé, les paramètres de connexion de la base de données sont affichés dans la boîte de dialogue **Ressource globale** (*voir la capture d'écran ci-dessous*).




Dans le dialogue **Ressource Globale**, il est possible d'éditer certains paramètres de connexions de base de données. Les paramètres sont groupés en deux catégories : les paramètres de base de données et les paramètres d'exécution spécifiques à MapForce (*voir ci-dessous*).

*Base de données*

Ces paramètres sont partagés parmi les applications Altova. Dans MapForce, ils sont utilisés au moment de la conception, c'est-à-dire, lorsque le mappage est chargé, ou lorsque vous cliquez sur le volet **Sortie** dans MapForce pour consulter le mappage.

*Paramètres d'exécution spécifiques à MapForce*

Ces paramètres sont applicables lorsque vous générez du code de programmation ou compilez un mappage sur le fichier d'exécution MapForce Server (`.mfx`). Ils sont utilisés lors de l'exécution du mappage comme suit :

- Dans le code de programme C++, C# ou Java généré.
- Si vous compilez le mappage dans un fichier d'exécution MapForce Server, la conversion JDBC automatique a lieu. Pour plus d'informations sur la conversion JBCD automatique, voir Mappages de base de données dans des environnements d'exécution divers[159]..

Si un mappage utilise une Ressource globale pour se connecter à une base de données, les détails de connexion de base de données dans le dialogue **Ressource globale** prend précédence sur ceux définis dans le mappage. La boîte de dialogue Paramètres de composant vous informe que les paramètres de connectivité

sont définis en tant que Ressource globale. Pour changer le composant de base de données pour se connecter à la base de données directement (sans utiliser de Ressources globales), cliquez sur **Changer**, et suivre les étapes de l'assistant pour se reconnecter à la base de données.

## 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 (voir Se connecter à une base de données 161). 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.

# 11.3 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**.

# 11.4 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**.

# 11.5 Bases de données en tant que Ressources globales

Cette rubrique explique comment utiliser les bases de données tant que ressources globales. Vous rencontrerez des situations dans lesquelles vous aurez besoin de mapper des données depuis des bases de données avec la même structure mais différentes données. Utiliser une ressource de base de données qui vous permet de basculer entre bases de données sans éditer votre mappage. À cette fin, nous devons créer un alias de base de données avec deux configurations :

1. La configuration `Default` pointera vers la `DevelopmentDatabase` : **`MapForceExamples\Altova.sqlite`**.
2. La configuration `ReleaseDatabase` pointera vers **`Tutorial\ Altova.sqlite`**.

Pour créer et configurer un alias de base de données, suivez les étapes ci-dessous.

## Étape 1 : Créer une ressource globale

La première étape est de créer un alias de base de données. Suivez les étapes 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 | Base de données** et saisissez un descriptif nom dans la zone de texte **alias de Ressource**. Dans cet exemple, nous appelons la configuration par défaut `DevelopmentDatabase`.
3. Cliquez sur **Choisir Base de données**, sélectionnez **Microsoft Access (ADO)** et cherchez **`MapForceExamples\altova.mdb`**.
4. Cliquez sur  et nommez la deuxième configuration `ReleaseDatabase`.
5. Cliquez sur **Choisir Base de données**, sélectionnez **Microsoft Access (ADO)** et cherchez **`Tutoriel\Altova.sqlite`**.

## Étape 2 : Utiliser la Ressource Globale dans le mappage

Dans la prochaine étape, nous allons configurer le mappage pour que vous puissiez utiliser l'alias de base de données :

1. Ouvrez le mappage **`Tutorial\PersonDB.mfd`**.
2. Double-cliquez sur le composant de base de données pour ouvrir la boîte de dialogue **Paramètres de composant**. Cliquez sur **Changer**.
3. Sélectionnez **Ressources globales** dans la boîte de dialogue **Sélectionner une base de données** et sélectionnez l'alias `DevelopmentDatabase`. Cliquez sur **Connexion**.
4. Vous êtes invité à sélectionner les objets de base de données, laissez la sélection par défaut telle quelle et cliquez sur **OK**.

Les paramètres de connectivité peuvent être modifiés en cliquant sur le bouton de la barre d'outils  .

## Étape 3 : Exécuter le mappage avec la configuration désirée

Vous pouvez maintenant facilement passer entre les bases de données avant d'exécuter le mappage comme suit :

- Pour utiliser la configuration `DevelopmentDatabase`, sélectionnez l'item de menu **Outils | Configuration active | Par défaut**.

- Pour utiliser la configuration `ReleaseDatabase`, sélectionnez l'item de menu **Outils | Configuration active |ReleaseDatabase**.

En alternative, sélectionnez la configuration requise depuis la liste déroulante **Ressources Globales** (*voir la capture d'écran ci-dessous*).


Lorsque vous passez entre les configurations, la boîte de dialogue **Configuration switch** vous informe que la ressource a été modifiée. Cliquez sur **Recharger**.

**Note :** les bases de données utilisées dans cet exemple contiennent les mêmes données. C'est la raison pour laquelle il n'y a pas de différence dans les sorties générées.

# 11.6 Résultats de transformation MapForce et StyleVision comme Ressources globales

Vous pouvez utiliser le résultat d'un mappage de MapForce ou d'une transformation StyleVision comme ressource globale. Cette rubrique vous montre comment créer une ressource globale depuis le résultat de transformation et utilise cette ressource globale dans toutes les différentes applications d'Altova.

Afin de rendre une sortie de mappage disponible comme ressource globale, le langage de transformation du mappage doit être défini comme Built-In, ou le mappage doit contenir d'autres composants qui sont pris en charge par le langage Built-In.

**Important :**

- Les flux de travail mentionnés ci-dessus sont significatifs entre les applications desktop d'Altova installées sur le même ordinateur.
- Il n'est **pas** possible d'utiliser le résultat des transformations de MapForce et StyleVision comme ressources globales dans les produits du serveur Altova et dans MapForce Basic Edition.

L'exemple ci-dessous montre comment utiliser le résultat d'une transformation de MapForce comme ressource globale.

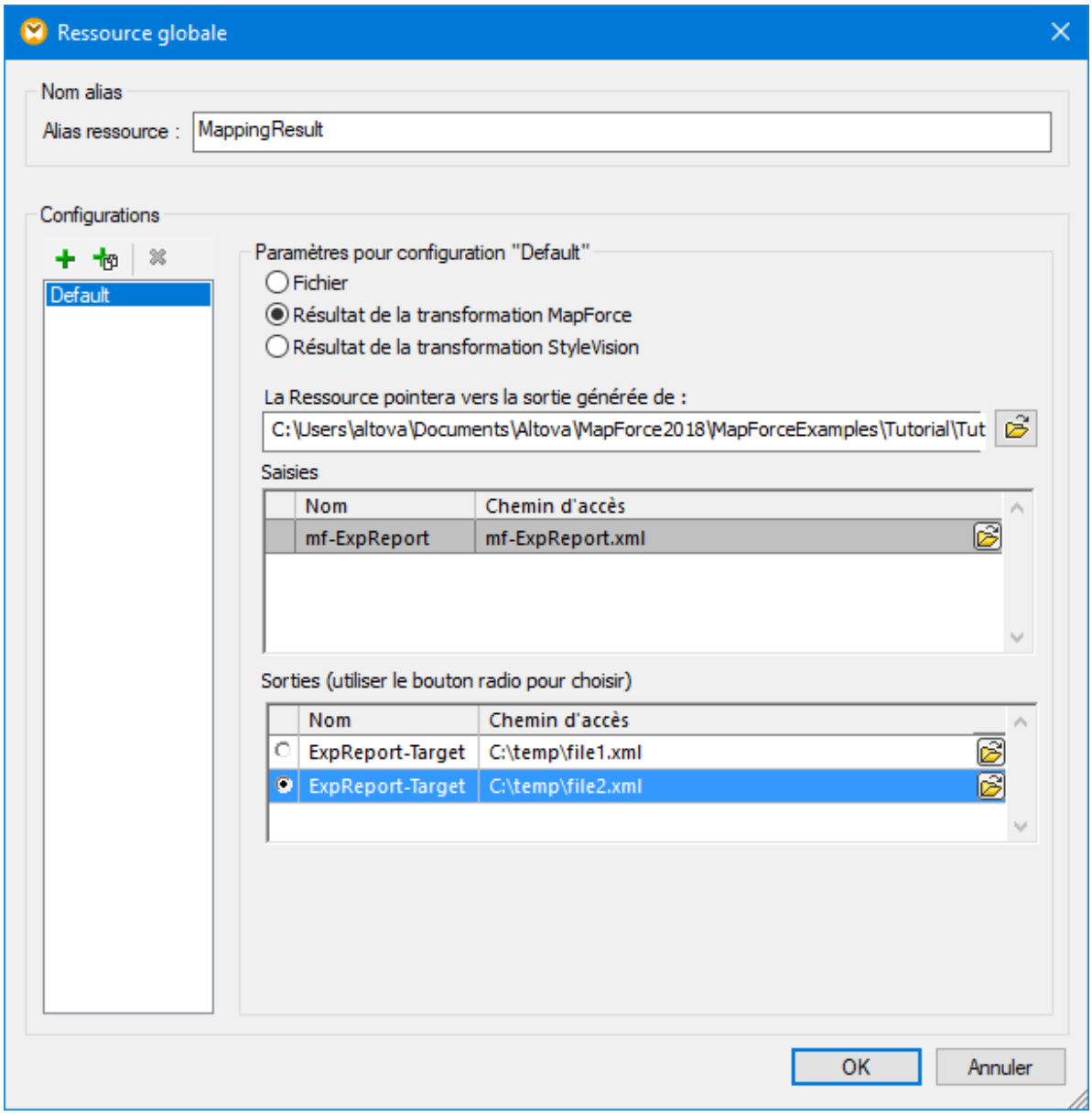
## Exemple : Résultat de la transformation de MapForce

Cet exemple illustre comment créer un flux de travail entre Altova MapForce et Altova XMLSpy, utilisant les Ressources globales. Plus spécifiquement, l'exemple montre comment créer une ressource globale depuis un mappage de MapForce, déclencher l'exécution de ce mappage dans XMLSpy, et visualiser les sorties dans XMLSpy, qui ont été générés par MapForce.

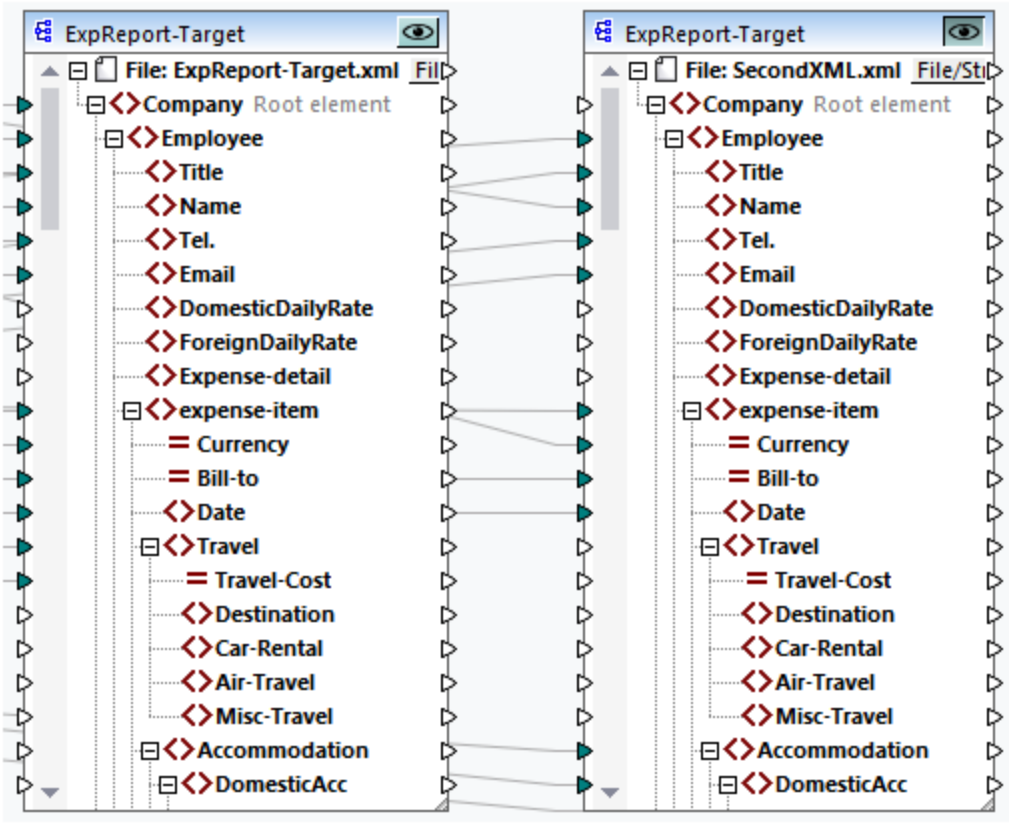
## Étape 1 : Créer une ressource globale

Vous pouvez prendre cette étape en utilisant MapForce ou XMLSpy.

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 descriptif nom dans la zone de texte **alias de Ressource**. Dans cet exemple, nous appelons notre configuration par défaut `RésultatDeMappage`.
3. Sélectionnez l'option **Résultat de la transformation de MapForce**.
4. Cliquez sur **Parcourir** et sélectionnez le mappage **`Tutorial\Tut-ExpReport-multi.mfd`**. Comme indiqué ci-dessous ce mappage à une entrée et deux sorties.




La capture d'écran ci-dessous illustre les deux sorties dans la boîte de dialogue de la **Ressource globale**. Nous générerons chaque fichier de sortie séparément dans le dossier **`C:\temp`** (*voir l'étape 2 ci-dessous*).




## Étape 2 : Générer des fichiers de sortie

À cette étape, nous aimerions générer chacun des deux fichiers de sortie dans le dossier (*voir la capture d'écran ci-dessus*) dans le dossier **`C:\temp`** et changer les noms de fichier. Pour ce faire, nous allons créer une configuration pour chaque sortie. Suivez les étapes ci-dessous :

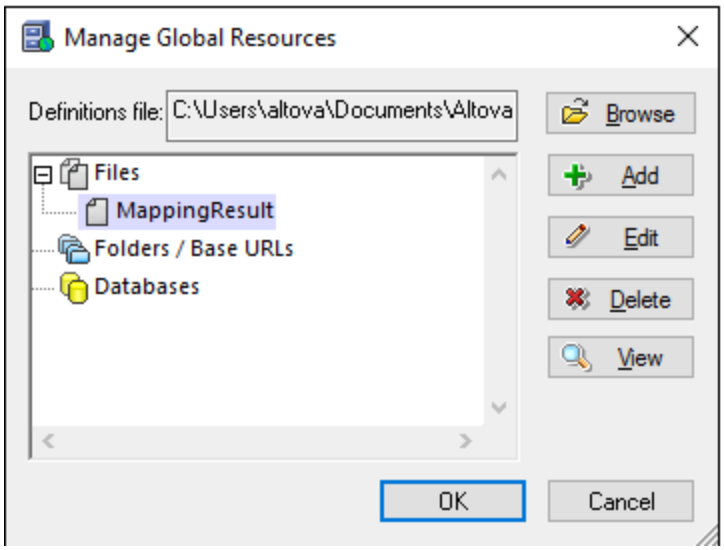
1. Dans la section *Sorties* de la boîte de dialogue **Ressource globale**, cliquez sur **Parcourir** à côté de la première sortie et saisissez **`C:\temp\file1.xml`** comme nom de fichier de destination. Il s'agit de la configuration par défaut qui produira le premier fichier de sortie.
2. Cliquez sur  sous les *Configurations* et saisissez un nom pour la nouvelle configuration (dans cet exemple, `Output2`). Dans la section *Sorties*, cliquez sur la case d'option à côté du deuxième fichier (**`SecondXML.xml`**).
3. Dans la section *Outputs*, cliquez sur **Parcourir** à côté du deuxième sortie et saisissez **`C:\temp\file2.xml`** en tant que nom de fichier de destination. Il s'agit de la deuxième configuration qui produit le deuxième fichier de sortie.
4. Cliquez sur **OK**.

## Étape 3 : Utiliser la Ressource globale

Les instructions ci-dessous vous montre comment utiliser la ressource globale que nous avons créé à l'étape précédente.

*Configuration par défaut*
Pour utiliser la configuration par défaut dans XMLSpy, suivez les étapes ci-dessous.

1. Exécuter XMLSpy.
2. Allez au menu **Outils** et cliquez sur **Ressources globales**.
3. Dans la section *Fichiers*, cliquez sur la ressource globale `MappingResult`, puis cliquez sur **Aperçu** (*voir la capture d'écran ci-dessous*). Ceci permet d'exécuter le mappage, produit la sortie par défaut (**`file1.xml`**) et le charge dans le volet principal de XMLSpy. Le fichier est enregistré comme **`C:\temp\file1.xml`**.




*Deuxième configuration*
Pour déclencher l'exécution de mappage avec la deuxième configuration, procédez comme suit :

1. Allez au menu **Outils** dans XMLSpy et cliquez sur **Configuration Active | Output2**.
2. Cliquez sur **Recharger** lorsque vous y êtes invité.

Par conséquent, le deuxième fichier de sortie est chargé dans la fenêtre principale de XMLSpy. Le fichier est enregistré comme **`C:\temp\file2.xml`**.

# 11.7 Ressources globales dans des environnements d'exécution variés

Cette sous-section explique comment travailler avec les ressources globales dans différents environnements d'exécution. La sous-section est organisée en rubriques suivantes :



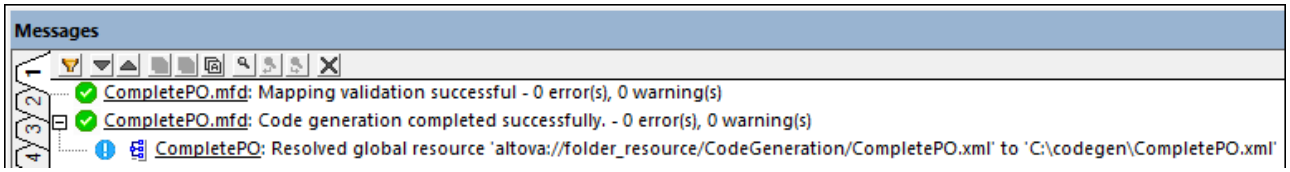
## 11.7.1 Ressources globales dans XSLT, XSLT2, XQuery

Cette rubrique explique comment es ressources globales sont utilisées dans le code généré. Pour plus d'information, veuillez voir les sous-sections ci-dessous.

*Ressources globales dans XSLT, XSLT2, XQuery*
Lorsque vous générez un code XSLT ou XSLT2 et que le mappage utilise des Ressources globales, cela n'affecte pas la feuille de style XSLT générée. Avec ou sans ressources globales, vous pouvez spécifier de manière flexible les fichiers d'entrée et de sortie quand vous exécutez la feuille de style XSLT dans votre processeur XSLT. La même chose s'applique au code XQuery généré.

Une exception est le fichier **`DoTransform.bat`** généré pour l'exécution RaptorXML. Les Ressources globales utilisées par le mappage seront résolues dans les chemins actuels dans **`DoTransform.bat`**. La configuration étant actuellement sélectionnée depuis la liste déroulante des ressources globales sera prise en compte. Pour plus d'informations concernant la livraison de Ressources globales dans RaptorXML, voir la documentation de RaptorXML.

*Ressources globales dans C++, C#, Java*
Lorsque vous générez un code de programme C#, C++, ou Java, Ressource globale utilisée par le mappage sera résolue. Par exemple, un alias de fichier ou de dossier défini en tant que Ressource globale sera converti dans le chemin de fichier ou de dossier actuel. Si une configuration de Ressource globale particulière est sélectionnée depuis la liste déroulante des Ressources globales, le code sera généré pour la configuration sélectionnée. La fenêtre **Messages** affiche des informations expliquant exactement comment une Ressource globale a été résolue (*voir capture d'écran ci-dessous*).




Afin de générer le code pour une configuration de Ressource globale particulière, choisissez-le depuis la liste déroulante de la Ressource globale avant de générer le code. En alternative, si vous générez du code depuis la ligne de commande, fournissez les paramètres **`GLOBALRESOURCEFILE`** et **`GLOBALRESOURCECONFIG`** au niveau de la ligne de commande. Pour plus d'informations, voir Interface de ligne de commande MapForce 876.

Il n'est pas possible de basculer vers les ou faire référence aux Ressources globales depuis le code généré. À la place, vous pouvez modifier le code pour changer le chemin de fichier d'entrée ou de sortie.

**Note :** dans C# ou Java, vous pouvez modifier le chemin et le type de données de l'entrée ou de la sortie.

## 11.7.2 Ressources globales dans MapForce Server

Lorsque vous compilez un mappage dans un fichier d'exécution MapForce Server (**.mfx**), toutes les références de ressources globales utilisées par le mappage sont préservées, mais pas résolues. Cela signifie que vous devrez fournir ces références du côté du serveur pour pouvoir exécuter le mappage avec succès. Dans MapForce Server, l'étape suivante est requise pour exécuter un fichier **.mfx** qui utilise les ressources globales :

1. *Le fichier de définition des ressources globales*. Sur l'appareil sur lequel MapForce est installé, le fichier est appelé **GlobalResources.xml**. Vous trouverez ce mappage dans le dossier **Documents\Altova**. Vous pouvez copier ce fichier sur l'appareil sur lequel MapForce est exécuté et créer plusieurs fichiers de ce type, le cas échéant. Voir aussi Configuration des Ressources globales, Partie 1 882.
2. *Le nom de configuration de la Ressource globale*. Chaque Ressource globale a une configuration par défaut. Vous pouvez aussi créer des configurations supplémentaires. Pour plus d'informations, voir Configuration des Ressources globales, Partie 2 884.

Dans MapForce, le fichier de définition des ressources globales et le nom de configuration de la Ressource globale sont définis ou modifiés depuis l'interface utilisateur graphique. Dans MapForce Server, elles sont spécifiées lors de l'exécution du mappage (*voir ci-dessous*).

- Si vous exécutez le mappage par l'interface de ligne de commande, définissez les options **--globalresourceconfig** et **--globalresourcefile** après la commande `run`, par exemple :

```
C:\Program Files (x86)\Altova\MapForceServer2024\bin\MapForceServer.exe run
SomeMapping.mfx --globalresourcefile="C:
\Users\me\Documents\Altova\GlobalResources.xml" --globalresourceconfig="Default"
```

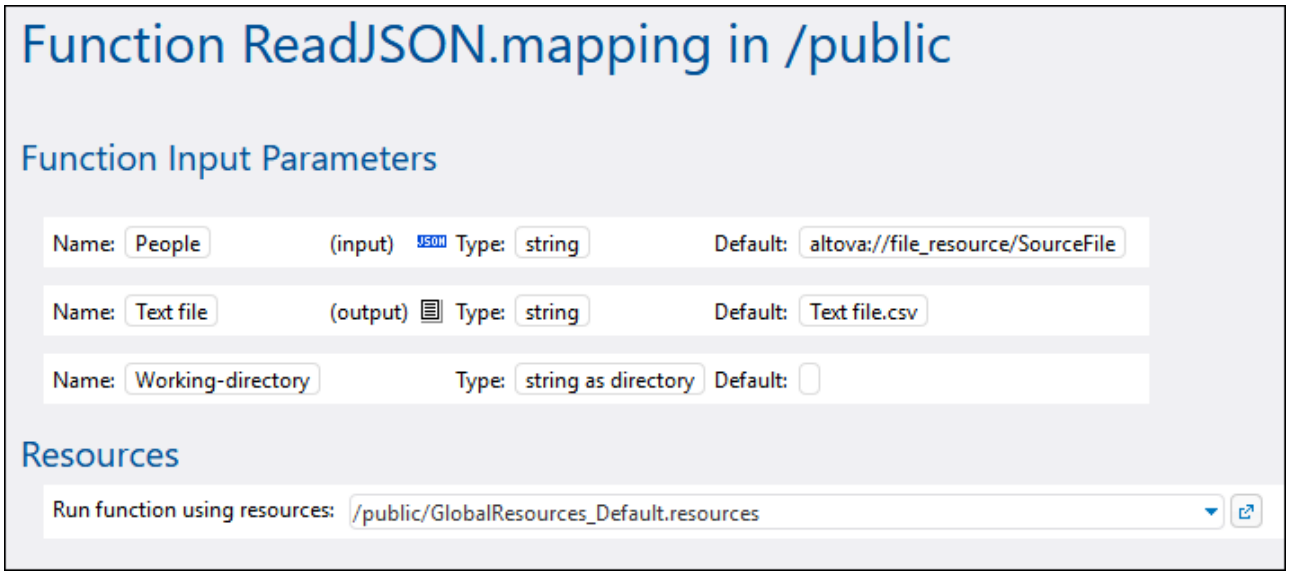
- Si vous exécutez le mappage par l'API de MapForce Server, appelez la méthode **SetOptions** deux fois avant d'appeler la méthode **Run**. Le premier appel est nécessaire pour fournir le chemin de fichier de définition de la Ressource Globale en tant qu'option, et le second appel est nécessaire pour fournir le nom de configuration de la Ressource Globale.

Pour plus d'informations, voir la documentation de MapForce Server.

## 11.7.3 Ressources globales dans FlowForce Server

Dans FlowForce Server, les ressources globales ne sont pas stockées dans un fichier XML comme dans les applications desktop. Dans FlowForce, chaque ressource est un objet réutilisable qui peut contenir des chemins de fichier ou de dossier ou des détails de connexion de base de données. Les ressources peuvent être copiées, exportées et importées, et sont sujettes au même mécanisme d'accès d'utilisateur que d'autres objets de FlowForce Server. Ceci signifie que tout utilisateur FlowForce peut utiliser toute ressource dans ses fonctions de mappage, s'il est doté des permissions exigées.

Une fois que vous avez créé un mappage avec des ressources globales dans MapForce, vous pouvez le déployer sur FlowForce Server. Au moment du déploiement, vous pouvez choisir si le mappage doit utiliser des ressources globales, cochant alors la case **Utiliser les ressources** dans le dialogue de déploiement. Si vous ne cochez pas la case, toutes les ressources globales utilisées par le mappage seront résolues, basées sur la configuration sélectionnée actuellement. Si vous avez coché la case, la fonction de mappage nécessitera aussi des ressources dans FlowForce Server. La capture d'écran ci-dessous est un exemple de fonction de mappage déployée vers FlowForce qui exige des ressources pour être exécutée. Veuillez noter que le premier paramètre obtient un chemin par défaut de la ressource.




Dans FlowForce Server, c'est la fonction de mappage qui utilise des ressources globales, pas la tâche. La fonction de mappage lit le chemin du premier fichier d'entrée depuis la ressource. Cela signifie que toutes les tâches employant cette fonction utiliseront le même chemin, à moins que vous contourniez le chemin depuis la page de configuration de la tâche.

Vous pouvez aussi déployer des Ressources globales dans FlowForce Server en tant qu'objets autonomes. Cela signifie il n'est pas nécessaire de déployer d'abord un mappage pour pouvoir déployer une ressource globale. Pour plus d'informations, voir *Déployer des Ressources sur FlowForce Server* ci-dessous.

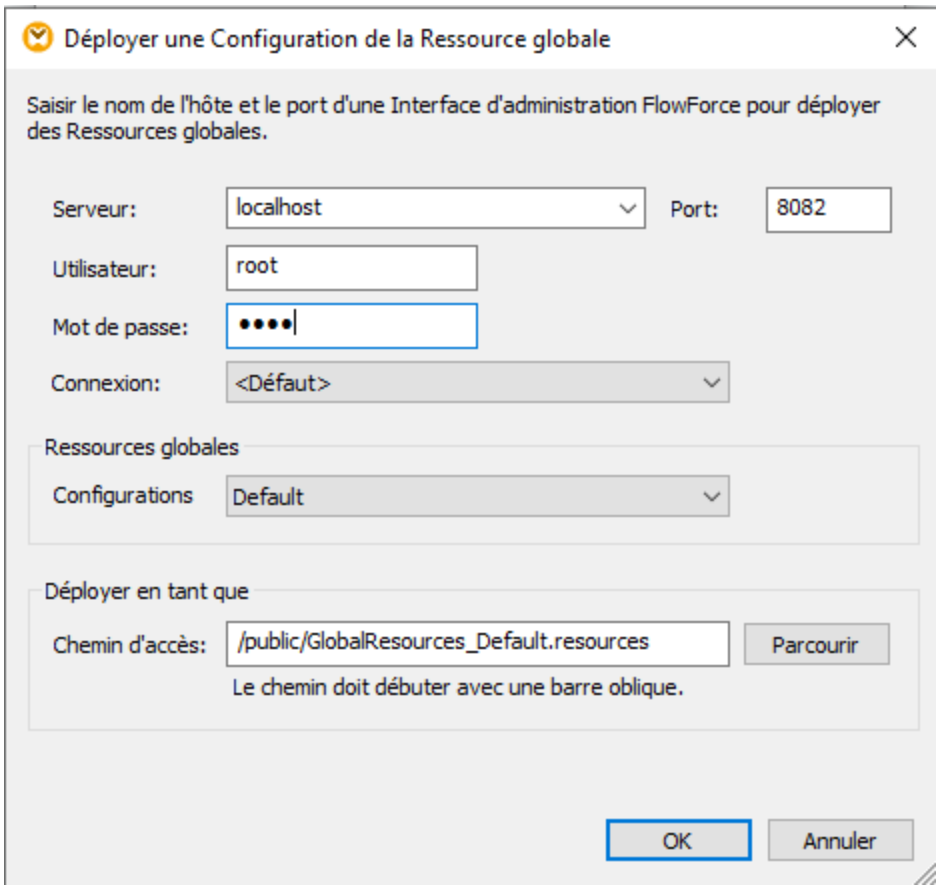
Pour plus d'informations concernant la consommation des ressources dans FlowForce Server, voir (the FlowForce Server documentation).

## Déployer des ressources sur FlowForce Server

Vous pouvez déployer des Ressources Globales créées avec MapForce sur FlowForce Server. Lors du déploiement, vous devez choisir la configuration avec laquelle la ressource doit être déployée vers le serveur. Si vous avez besoin de toutes les configurations de la même ressource globale sur le serveur, vous pouvez déployer cette ressource globale plusieurs fois et choisir la configuration souhaitée avant chaque déploiement. Vous pouvez également changer le nom de chaque ressource globale sur le serveur et choisir le conteneur de destination sur le serveur.

Vous pouvez déployer des Ressources Globales sur FlowForce Server soit au moment où vous déployez le mappage ou séparément. Pour déployer les ressources globales sur FlowForce Server, suivez les étapes suivantes :

1. Exécuter MapForce.
2. Cliquez sur la touche de la barre d'outils **Gérer les Ressources globales**. En alternative, allez au menu **Outils** and cliquez sur **Ressources globales**.
3. Cliquer sur **Déployer sur le serveur**. Ceci ouvre la boîte de dialogue **Déployer la configuration de la Ressource globale** (*voir la capture d'écran ci-dessous*).




4. Saisir les détails de connexion sur FlowForce Server (serveur, port, utilisateur, mot de passe, méthode de connexion). Ces paramètres sont les mêmes que ceux requis lors du déploiement d'un mappage, voir Déployer des mappages sur FlowForce Server 867.
5. Sélectionnez une configuration depuis la liste de **Configuration**. Cette liste inclut toutes les configurations depuis le fichier de définition des ressources globales 882 actuel. Veuillez noter que seule une configuration de ressource globale peut être déployée à la fois. Vous pouvez déployer la même ressource plusieurs fois avec un nom différent, si vous avez besoin de toutes les configurations sur le serveur.
6. Choisir un chemin cible où la ressource doit être enregistrée sur le serveur. Cliquer sur **Parcourir** pour afficher un dialogue dans lequel vous pouvez sélectionner un conteneur FlowForce cible, ou en créer un nouveau, le cas échéant.
7. Cliquez sur **OK**.

Vous pouvez voir l'information sur le déploiement des ressources globales sur FlowForce Server dans la fenêtre des **Messages**.

**Note :** les Ressources globales qui exécutent d'autres applications d'Altova ne sont *pas* prises en charge dans un environnement de serveur. Pour plus d'informations, voir Résultats de transformation en tant que Ressources globale 894.

# 12 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.

# 12.1 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.

## 12.2 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 |

| | |
|---|---|
| %<br>ApplicationWritableDataFolde<br>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.

## 12.3 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.

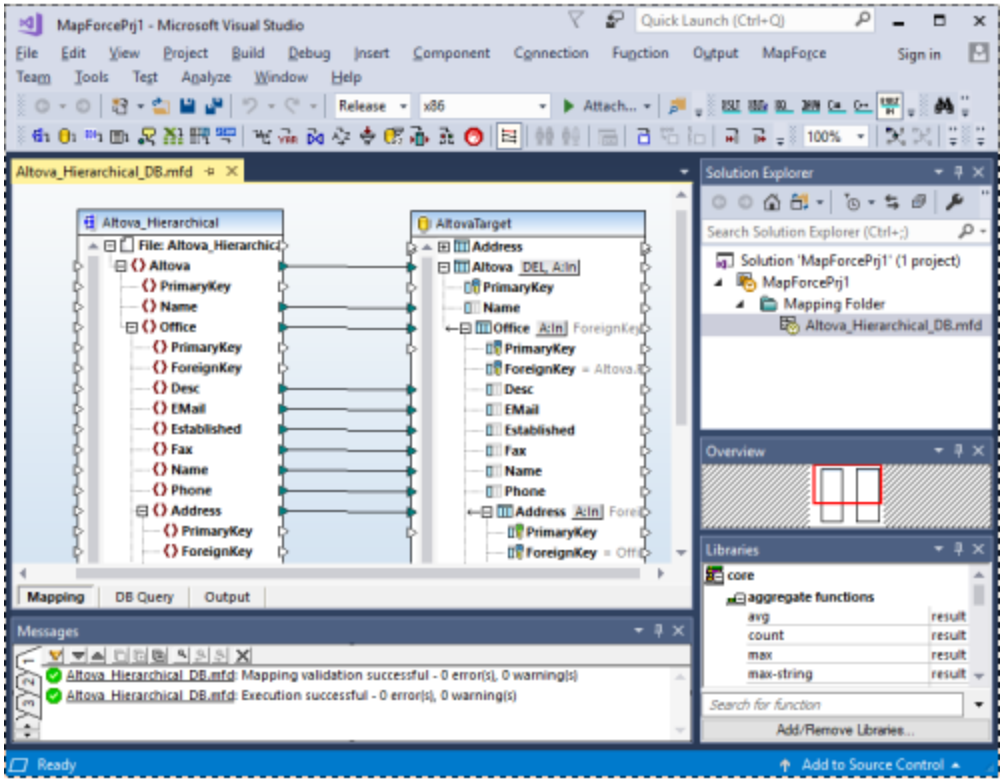
## 12.4 Variables d'Environnement

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. |

# 13 Plug-in MapForce pour Visual Studio

Vous pouvez intégrer MapForce 2024 dans les versions Microsoft Visual Studio 2012/2013/2015/2017/2019/2022. Cette intégration aide à combiner les capacités de mappage de MapForce avec l'environnement de développement de Visual Studio. Lorsque le plug-in de MapForce est activé, vous pouvez créer des mappages et des projets de mappage directement depuis Visual Studio (*voir la capture d'écran ci-dessous*).




## Installation

Pour installer le Plug-in MapForce pour Visual Studio, suivez les étapes suivantes :

1. Installer Microsoft Visual Studio 2012/2013/2015/2017/2019/2022. Veuillez noter qu'à partir de Visual Studio 2022, Visual Studio est mis à disposition uniquement comme application 64-bit.
2. Installez MapForce(Enterprise ou Professional Edition). Si vous avez installé Visual Studio 2022+, alors vous devez installer la version 64-bit de MapForce.
3. Téléchargez et exécutez le pack d'intégration MapForce pour Microsoft Visual Studio. Ce pack est disponible sur la page de téléchargement de MapForce (Enterprise et Professional Editions) sur www.altova.com

Une fois que le pack d'intégration a été installé, vous pourrez utiliser MapForce dans l'environnement de Visual Studio.

**Important**

Vous devez utiliser le pack d'intégration correspondant à votre version de MapForce (la version actuelle est 2024). La package d'intégration n'est pas spécifique à l'édition et peut donc être utilisé pour les deux éditions Enterprise et Professional.

## Information sur les menus et les fonctions

Lorsque le plug-in MapForce pour Visual Studio est activé, vous pouvez accéder aux menus et fonctions communs comme indiqué (*voir ci-dessous*). Vous pouvez personnaliser les menus et les barres d'outils MapForce depuis le menu **Tools | Customise** de Visual Studio.

**Note :** dans Visual Studio 2019 et plus élevé, la fonctionnalité de MapForce peut être accédée dans le menu **Extensions** de Visual Studio. Dans des versions antérieures de Visual Studio, les fonctions de MapForce sont disponibles dans des menus de niveau supérieur de Visual Studio.

### Thèmes

Vous pouvez sélectionner les thèmes de MapForce dans le menu **MapForce** de Visual Studio. Les options sont les thèmes Classique, Clair et Sombre.

### Créer et ouvrir des fichiers/projets

Si le plug-in de MapForce pour Visual Studio est activé, vous pouvez créer, ouvrir et travailler avec des mappages et des projets de mappage directement depuis Visual Studio. Pour créer un nouveau fichier design de mappage dans Visual Studio, utilisez la commande de menu **Fichier | Nouveau**. Pour créer un nouveau projet, utilisez la commande de menu **Fichier | Nouveau projet**. Pour ouvrir des fichiers de mappage existants ou des proejts, vous pouvez utiliser les menus suivants de Visual Studio : **Fichier | Ouvrir | Fichier** ou **Fichier | Ouvrir | Projet/Solution**. Puis, vous pouvez rechercher les types de fichiers liés à MapForce.

### Ressources globales

Les Ressources globales[881] MapForce sont disponibles dans le menu **MapForce | Gérer Ressources Globales** de Visual Studio. À partir de Visual Studio 2019, le menu correspondant est **Extensions | MapForce | Gérer les Ressources globales**.

### Débogage

Après avoir ouvert un fichier de mappage, les commandes de débogage de mappage sont disponibles dans le menu Déboguer[1078] et dans la barre d'outils **Déboguer**. À partir de Visual Studio 2019, le menu correspondant est **Extensions | MapForce | Debug**.

### Options MapForce

Les options MapForce sont disponibles dans le menu **Tools | Options MapForce** de Visual Studio.

### Personnalisation du volet de mappage

Lorsqu'un mappage de MapForce est dans le volet principal de Visual Studio, le menu **View | MapForce** devient disponible. Il comprend les mêmes options que le menu **View** de la version autonome de MapForce.

### Fenêtre de bibliothèques

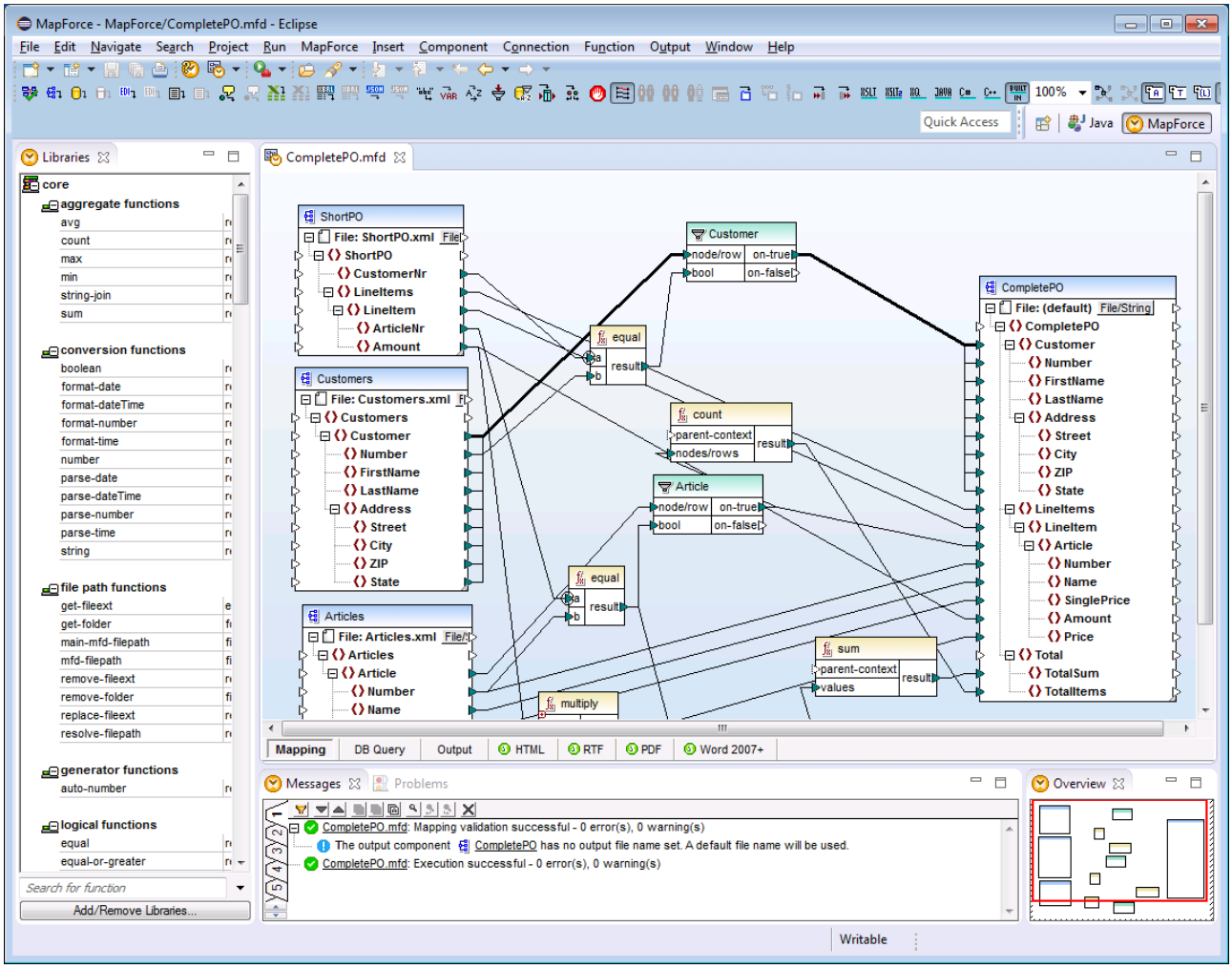
Si la fenêtre **Bibliothèques** MapForce n'est pas visible dans Visual Studio, vous pouvez l'activer depuis le menu **View | MapForce | Libraries Window** (ce menu devient disponible dans Visual Studio lorsqu'un fichier de mappage est ouvert). Une fois que la fenêtre **Bibliothèques** est activée, vous pouvez l'ancrer dans une position particulière dans l'interface.

- Aide et soutien

  Les menus **Aide**, **Centre de support**, **Vérifier les mises à jour** et **à propos de** MapForce sont disponibles dans le menu **Aide | Aide MapForce** de Visual Studio.

# 14 Plug-in MapForce pour Eclipse

Eclipse est un framework open source qui intègre différents types d'applications fournies sous la forme de plugins. Vous pouvez intégrer les éditions MapForce Enterprise et Professional dans les versions d'Eclipse 2024-03 (4.31), 2023-12 (4.30), 2023-09 (4.29), 2023-06 (4.28) et accéder aux fonctions MapForce directement depuis Eclipse.




*Plug-in MapForce Enterprise Edition pour Eclipse*

Les rubriques suivantes vous soutiennent dans l'installation et l'utilisation du plug-in MapForce pour Eclipse.

# 14.1 Installer le plug-in MapForce pour Eclipse

## Prérequis

- Eclipse 2024-03 (4.31), 2023-12 (4.30), 2023-09 (4.29), 2023-06 (4.28) (http://www.eclipse.org) exploitation 64-bit.
- Java Runtime Environment (JRE) ou Java Development Kit (JDK) pour la plateforme 64-bit.
- MapForce Enterprise ou Professional Edition 64-bit.

**Note :** tous les prérequis cités ci-dessus doivent être dotés de la plateforme 64-bit. L'intégration avec d'autres plateformes 32-bit n'est plus prise en charge, bien qu'elle puisse encore fonctionner.

Une fois que les pérequis ci-dessus sont en place, vous pouvez installer le Package d'intégration (64-bit) MapForce pour intégrer MapForce dans Eclipse. L'intégration peut être effectuée au cours de l'installation du Package d'intégration ou manuellement depuis Eclipse une fois que le Package d'intégration a été installé. Le Package d'intégration MapForce est disponible pour être téléchargé à l'adresse https://www.altova.com/components/download.

**Note :** Eclipse doit être fermé pendant que vous installez ou désinstallez le Package d'intégrationMapForce.

## Intégrez MapForce lors de l'installation du Package d'intégration

Vous pouvez intégrez MapForce dans Eclipse lors de l'installation du Package d'intégration MapForce. Pour ce faire, suivez les étapes suivantes :

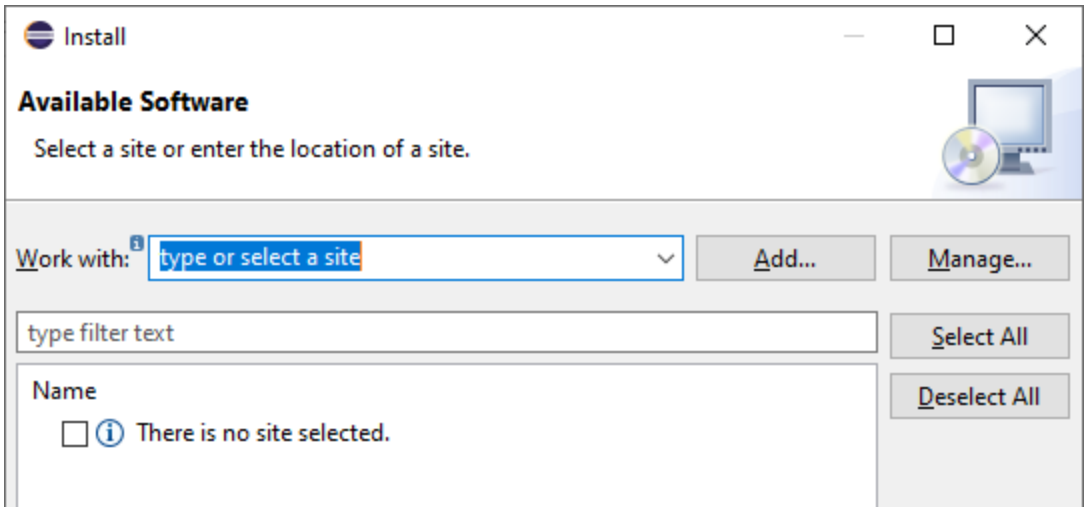
1. Exécutez le Package d'intégration MapForce pour lancer l'assistant d'installation.
2. Suivez les étapes initiales de l'installation avec l'assistant d'installation.
3. Dans l'étape d'intégration, sélectionnez *Permettre à l'assistant d'intégrer Altova MapForce plug-in dans Eclipse*, et cherchez le répertoire dans lequel le fichier exécutable pour Eclipse (`eclipse.exe`) se trouve.
4. Cliquez sur **Suivant** et terminez l'installation.

La perspective MapForce et les menus seront disponibles dans Eclipse la prochaine fois que vous le démarrez.

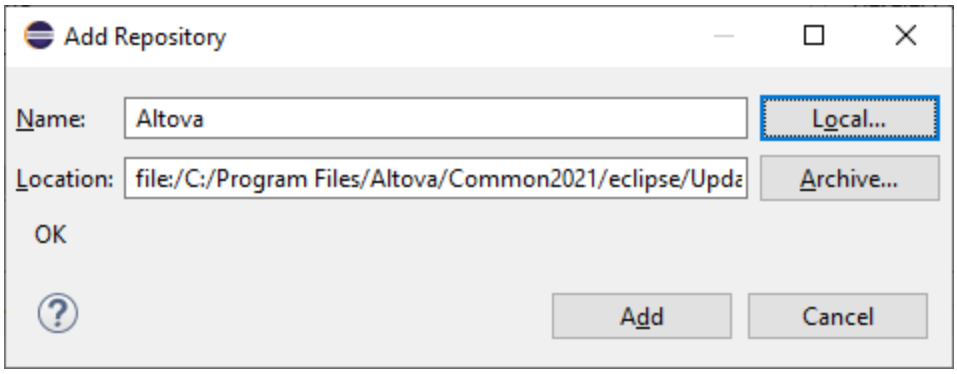
## Intégrer MapForce dans Eclipse manuellement

Une fois que vous avez installé le Package d'intégration MapForce, vous pouvez intégrer manuellement MapForce dans Eclipse comme suit:

1. Dans Eclipse, sélectionnez la commande de menu **Aide | Installer nouveau logiciel**.
2. Dans la boîte de dialogue Installer, cliquez sur **Ajouter**.




3. Dans la boîte de dialogue Ajouter Référentiel, cliquez sur **Local**. Recherchez le dossier `C:\Program Files\Altova\Common2024\eclipse\UpdateSite` et sélectionnez-le. Donner un nom au site (comme « Altova »).




4. Répétez les étapes 2 à 3 ci-dessus, en sélectionnant cette fois-ci le dossier `C:\Program Files\Altova\<% APPNAMESHORT%>\eclipse\UpdateSite` et en donnant un nom comme « Altova » MapForce".
5. Dans la boîte de dialogue Installer, sélectionnez *Uniquement sites locaux*. Puis, sélectionnez le dossier « catégorie Altova » et cliquez sur **Suivant**.
6. Revérifiez les éléments à installer puis cliquez sur **Suivant** pour procéder.
7. Pour accepter le contrat de licence, sélectionnez la case à cocher respective.
8. Cliquez sur **Terminer** et terminez l'installation.

**Note :** Si vous avez des problèmes avec le plug-in (icônes manquantes, par exemple), lancez Eclipse depuis la ligne de commande avec l'indicateur `-clean`.

## 14.2 La perspective MapForce

Dans Eclipse, une perspective est un affichage GUI qui est configuré avec la fonctionnalité d'une application spécifique. Une fois que MapForce a été intégré dans Eclipse, une nouvelle perspective, appelée MapForce, est rendue disponible dans Eclipse. Cette perspective est une GUI qui ressemble à la MapForce GUI et inclut un nombre de ses composants.

Lorsqu'un fichier dont le type de fichier est associé avec MapForce est ouvert (`.mfd`), ce fichier peut être édité dans la perspective MapForce. De même, un fichier d'un autre type de fichier peut être ouvert dans une autre perspective dans Eclipse. En outre, pour tout fichier actif, vous pouvez changer de perspective (*voir ci-dessous*), donc permettre d'éditer ou de traiter ce fichier dans un autre environnement.

Deux avantages principaux se présentent donc pour les perspectives :

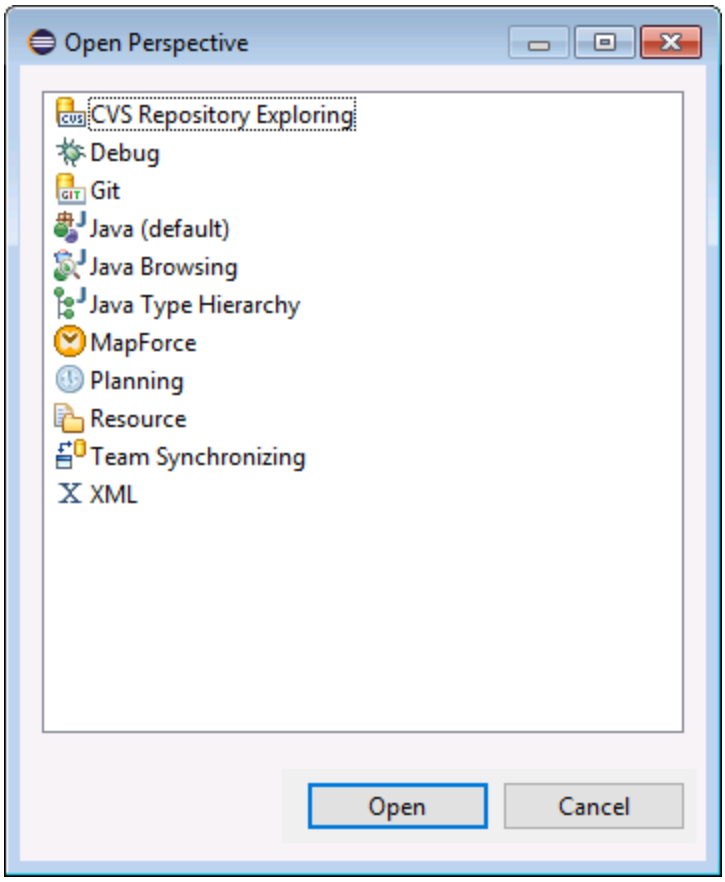
1. la possibilité de changer rapidement l'environnement de travail du fichier actif et
2. La possibilité de passer entre les fichiers sans devoir ouvrir un nouvel environnement de développement (l'environnement associé est disponible dans une perspective)

Travailler avec la perspective MapForce implique les procédures clé suivantes, qui sont décrites de manière plus détaillée ci-dessous :

- Passer à la perspective MapForce.
- Définir les préférences pour la perspective MapForce.
- Personnaliser la perspective MapForce.

### Passer à la perspective MapForce

Dans Eclipse, sélectionnez la commande **Window | Perspective | Open Perspective | Other**. Dans le dialogue qui s'ouvre (*capture d'écran ci-dessous*), sélectionnez **MapForce** et cliquez sur **Ouvrir**.




La fenêtre vide ou le document actif aura maintenant la perspective MapForce. Voici comment l'utilisateur change de perspective par le menu. Pour accéder à une perspective plus rapidement depuis une autre perspective, la perspective requise peut être recensée dans le sous-menu **Open Perspective** au-dessus de l'item **Other**. Ce paramètre est dans le dialogue de personnalisation (*voir plus bas ci-dessous*).

Les perspectives peuvent aussi être changées lorsqu'un fichier est ouvert ou est rendu actif. La perspective de l'application associée avec un type de fichier d'un fichier sera ouvert automatiquement lorsque ce fichier est ouvert pour la première fois. Avant de changer de perspective, un dialogue apparaît vous demandant si vous souhaitez associer automatiquement la perspective par défaut avec ce type de fichier. Vérifiez l'option *Do Not Ask Again* si vous souhaitez associer la perspective avec un filetype sans devoir y être invité chaque fois qu'un fichier de ce filetype est ouvert, puis cliquez sur **OK**.

## Les préférences pour la perspective MapForce

Pour accéder aux Préférences d'une perspective, sélectionnez la commande **Window | Preferences**. Dans la liste des perspectives du panneau de gauche, sélectionnez MapForce, puis sélectionnez les préférences requises. Terminez en cliquant sur **OK**.

Les préférences d'une perspective comprennent :

- Pour passer automatiquement à la perspective MapForce quand un fichier d'un type de fichier associé est ouvert (*voir ci-dessus*)
- Options pour inclure ou exclure des barres d'outils MapForce individuelles
- Accéder aux options MapForce.

## Personnalisez la perspective MapForce

Les options de personnalisation vous permettent de déterminer quels raccourcis et commandes sont inclus dans la perspective. Pour accéder au dialogue Personnaliser la Perspective, transformez la perspective en perspective active et sélectionnez la commande **Window | Perspective | Customize Perspective**.

- Dans les onglets *Toolbar Visibility* et *Menu Visibility*, vous pourrez spécifier quelles barres d'outils et menu doivent être affichés.
- Dans l'onglet *Action Set Availability*, vous pouvez ajouter des groupes de commandes à leurs menu parents et à la barre d'outils. Si vous souhaitez activer un groupe de commande, cochez sa case.
- Dans l'onglet *Shortcuts* du dialogue Customize Perspective, vous pouvez configurer des raccourcis pour les sous-menus. Sélectionnez le sous-menu dans la liste déroulante des sous-menus. Puis sélectionnez une catégorie de raccourcis, et cochez les raccourcis que vous souhaitez inclure à la perspective.

Cliquer sur **Appliquer et Fermer** pour terminer la personnalisation et pour que les modifications prennent effet.

## Aperçu des la perspective MapForce

Par défaut, la perspective MapForce dans Eclipse est organisée comme suit :

- La fenêtre de design de mappage est disponible en tant qu'éditeur Eclipse. Il dispose des mêmes onglets et fonctions dans l'édition autonome de MapForce.
- La fenêtre Bibliothèques (Libraries) est disponible en tant que mode Eclipse, à gauche de l'éditeur de mappage principal. Si cet affichage n'est pas visible, passer à la perspective MapForce, puis sélectionner la commande de menu **Window | Show View | Libraries**. Le mode Bibliothèques vous permet de travailler avec des fonctions prédéfinies ou définies par personnalisation et des bibliothèques de fonction.
- Le volet Messages est disponible en tant que mode Eclipse, sous l'éditeur de mappage principal. Si le mode Message n'est pas visible, passer à la perspective MapForce, puis choisir la commande de menu **Window | Show View | Messages**. Le mode Messages affiche les messages de validation, erreurs et avertissements.
- Le volet Aperçu est disponible en tant que mode Eclipse. Si le mode Aperçu n'est pas visible, passer à la perspective MapForce, puis sélectionner la commande de menu **Window | Show View | Overview**. Ce mode vous permet de naviguer rapidement dans une région particulière lorsque la zone de design de mappage est très grande.

## 14.3 Accéder aux menus et fonctions communs

Dans Eclipse, vous pouvez accéder la plupart des fonctions MapForce depuis les même menus que dans la version autonome, sauf pour certaines variations spécifique à Eclipse qui sont listées ci-dessous. Il s'agit de la configuration par défaut ; néanmoins, vous pouvez personnaliser les préférences d'interface depuis Eclipse, si vous le souhaitez (voir La Perspective MapForce 917).

**Note :** Dans Eclipse, certains groupes ou commandes de menu MapForce sont désactivées (ou non disponibles) si le contexte n'est pas pertinent. Par exemple, le menu **Insertion** devient disponible uniquement lorsqu'un fichier de design de mappage (.mfd) est activé dans Eclipse.

Pour plus d'informations concernant les menus de standard MapForce, voir Référence menu 1062.

### Commandes MapForce générales

Dans l'édition autonome de MapForce, les commandes applicables aux fichiers de design de mappage (comme **Validate**, **Deploy to FlowForce Server**, **Generate Code**, et autres) sont disponibles dans le menu **File**. Dans Eclipse, ces commandes sont disponibles dans le menu **MapForce**, ou dans la barre d'outils MapForce. Veuillez noter que les commandes pour ouvrir ou enregistrer les fichiers (y compris les fichiers de projet MapForce) sont disponibles dans le menu **File** d'Eclipse.

Les thèmes de MapForce peuvent être sélectionnés dans le menu **MapForce**.

*La barre d'outils MapForce dans Eclipse*

La touche de barre d'outils  ouvre le fichier d'aide MapForce.

La touche de barre d'outils  affiche des commandes spécifiques aux fichiers MapForce. Lorsque vous agrandissez cette touche, les commandes disponibles dépendent du type de fichier actif actuellement dans l'éditeur Eclipse. Par exemple, les commandes spécifiques au fichier de design de mappage (.mfd) sont disponibles lorsqu'un fichier de ce type est actif (dans focus) dans l'éditeur Eclipse.

### Ressources globales

Pour accéder ou gérer les Ressources globales, suivre une des étapes suivantes :

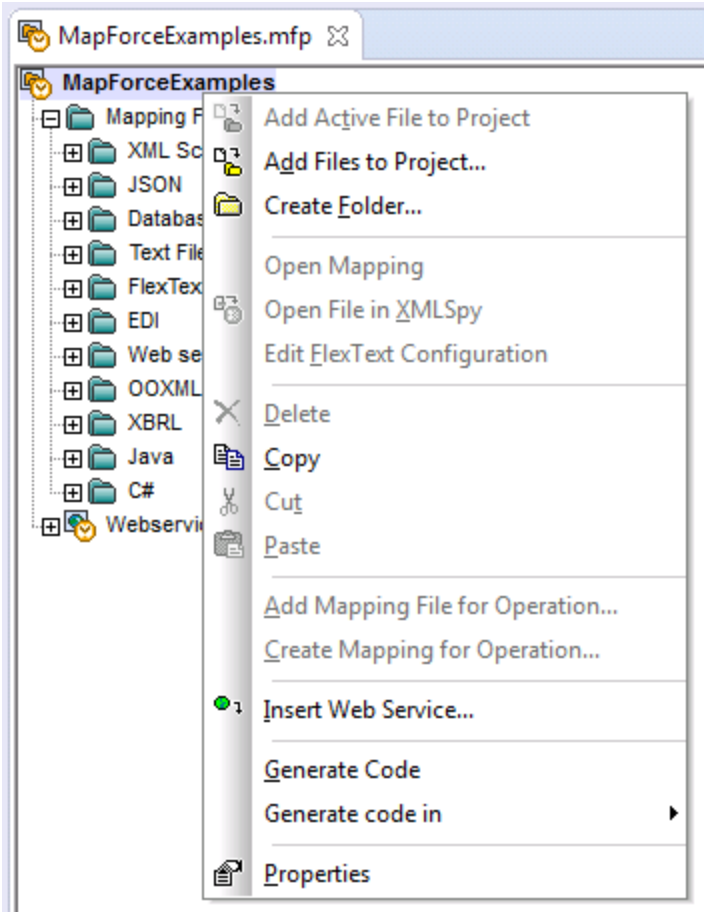
- Cliquez pour agrandir la touche de barre d'outils  MapForce , puis cliquez sur **Ressources globales**.
- Dans le menu MapForce, cliquez sur **Ressources globales**.

### Projets MapForce

Dans l'édition standard de MapForce, le menu **Projet** contient plusieurs commandes applicables aux fichiers de projet de mappage (.mfp). Dans Eclipse, ces commandes existent comme suit :

- Les commandes pour ouvrir ou enregistrer un projet sont disponibles depuis le menu Eclipse **File**.

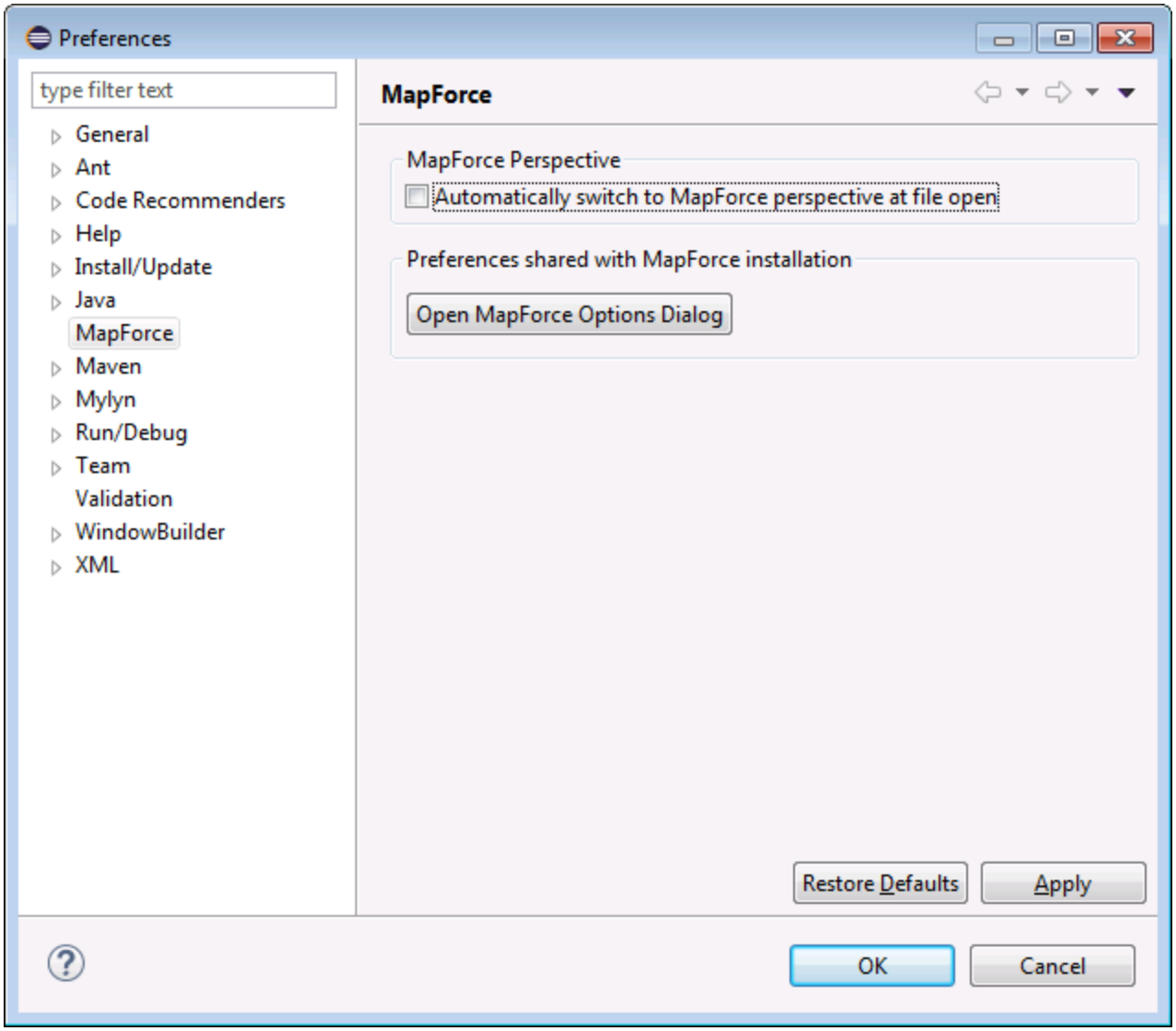
- D'autres commandes spécifiques au projet sont disponibles en tant que commandes contextuelles. Pour afficher les commandes contextuelles, créez ou ouvrez un fichier de projet MapForce (.mfp) dans Eclipse, puis cliquez avec la touche de droite sur le projet.




Veuillez noter qu'outre les projets MapForce standard (.mfp), dans Eclipse vous pouvez aussi créer des projets de type "MapForce/Eclipse". Ce type de projet est de nature double et peut être configuré pour la création et la génération automatique du code MapForce. Voir Travailler avec des mappages et des projets 924.

## Options MapForce

Des options MapForce sont disponibles depuis le menu **Window | Preferences**. Dans le dialogue Préférences, choisissez **MapForce**, puis cliquer sur **Ouvrir le dialogue d'options MapForce**.




*Dialogue Préférences*

## Fenêtre de bibliothèques

Dans Eclipse, la fenêtre MapForce Bibliothèques est disponible en tant qu'affichage. Cet affichage est situé par défaut à gauche de la fenêtre d'éditeur principal. (Tous les affichages liés à MapForce deviennent visibles dans l'interface Eclipse lorsque la perspective MapForce est activée, voir aussi La perspective MapForce 917).

## Version de plug-in MapForce

Pour consulter la version installée actuellement du plug-in MapForce pour Eclipse, sélectionner l'option de menu Eclipse **Aide | à propos d'Eclipse**. Puis, choisissez l'icône MapForce.

## Aide et soutien

Les menus MapForce Help, Support Center, Check for Updates et About sont disponibles dans le menu **Aide | Aide MapForce** d'Eclipse.

## 14.4 Travailler avec des mappages et des projets

Lorsque le plug-in MapForce pour Eclipse est installé, vous pouvez créer depuis Eclipse les mêmes mappages et les types de projet de mappage comme dans l'édition autonome de MapForce, depuis l'intérieur d'un projet Eclipse. Pour concevoir, tester, compiler et déployer des codes de mappages, vous pouvez soit créer un nouveau projet d'Eclipse ou utiliser un projet Eclipse existant (par exemple, un projet Java auquel vous pourrez souhaiter ajouter des mappages MapForce).

De plus, vous pouvez travailler avec tous vos mappages dans le cadre d'un type de projet spécifique qui devient disponible dans Eclipse après avoir installé le plug-in MapForce — le **projet MapForce/Eclipse**. À moins de choisir de la personnaliser, un Java Builder et un MapForce Code Generation Builder sont attribués par défaut au projet MapForce/Eclipse. De plus, il détient deux natures Eclipse : nature MapForce et la nature JDT (outils de développement Java). En résultat, un projet MapForce/Eclipse se comporte comme suit lorsque vous enregistrez ou modifiez une de ses ressources (comme le fichier de design de mappage) :

- Si l'option de menu **Project > Build automatically** est activée, le code de mappage est généré automatiquement. Lorsqu'un ou plusieurs fichiers de projet MapForce existent dans le projet MapForce/Eclipse, le langage de génération de code et les dossiers de cible de sortie sont déterminés par les paramètres dans chaque fichier de projet. Sinon, Eclipse vous demande de choisir un emplacement.
- Tous les messages d'erreur et de sortie sont affichés dans les affichages Messages et Problèmes.

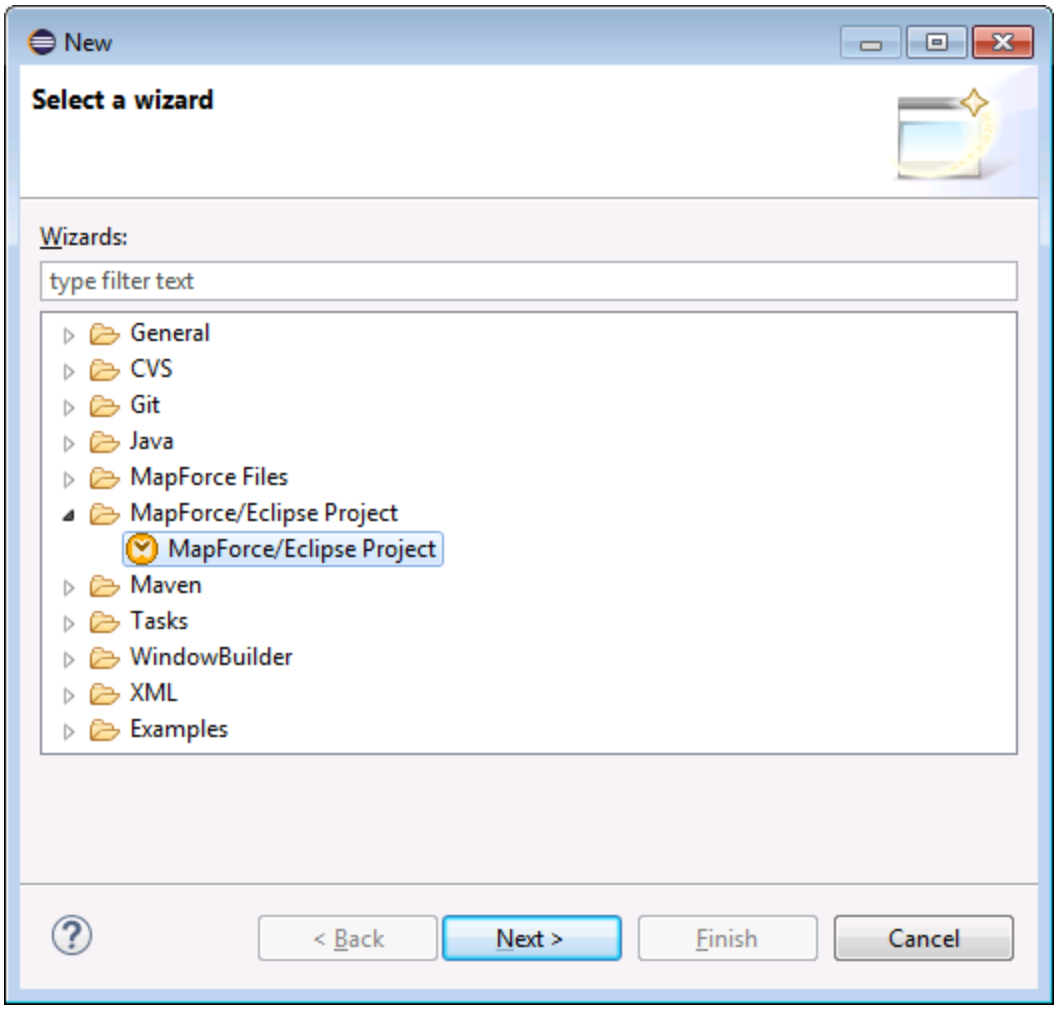
Cette section contient les rubriques suivantes :



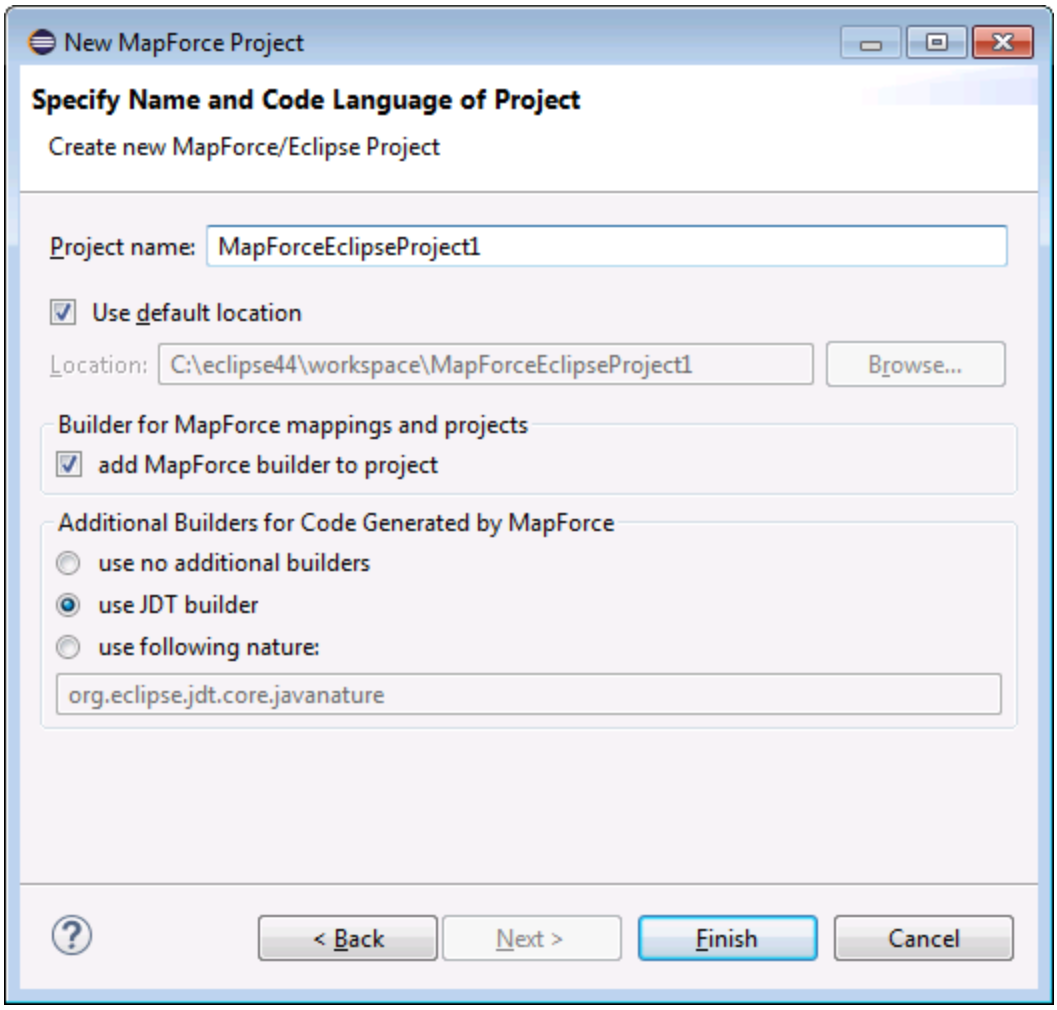
### 14.4.1 Créer un projet MapForce/Eclipse

**Pour créer un projet MapForce/Eclipse,** suivez les étapes ci-dessous **:**

1. Dans le menu **File**, cliquer **New | Other**.
2. Choisir la catégorie **MapForce/Eclipse Project**.




3. Cliquer sur **Next**.




4. Saisir un nom de projet et choisir un emplacement où sauvegarder le projet. Laisser les options **ajouter un générateur MapForce au projet** et **utiliser générateur JDT** comme elles sont.
5. Cliquer sur **Finish**.

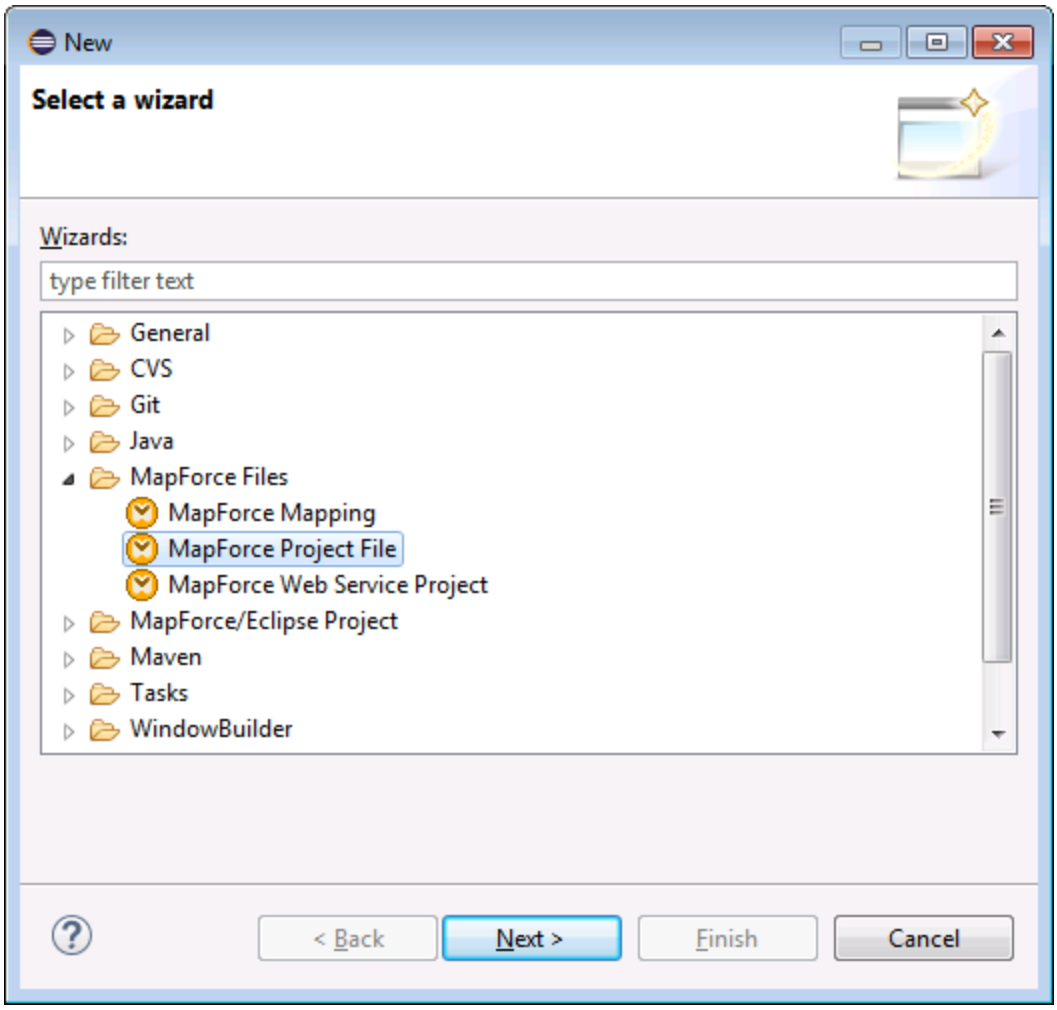
## 14.4.2 Créer de nouveaux mappages

Vous pouvez créer les types de fichier MapForce dans le cadre d'un projet Eclipse :

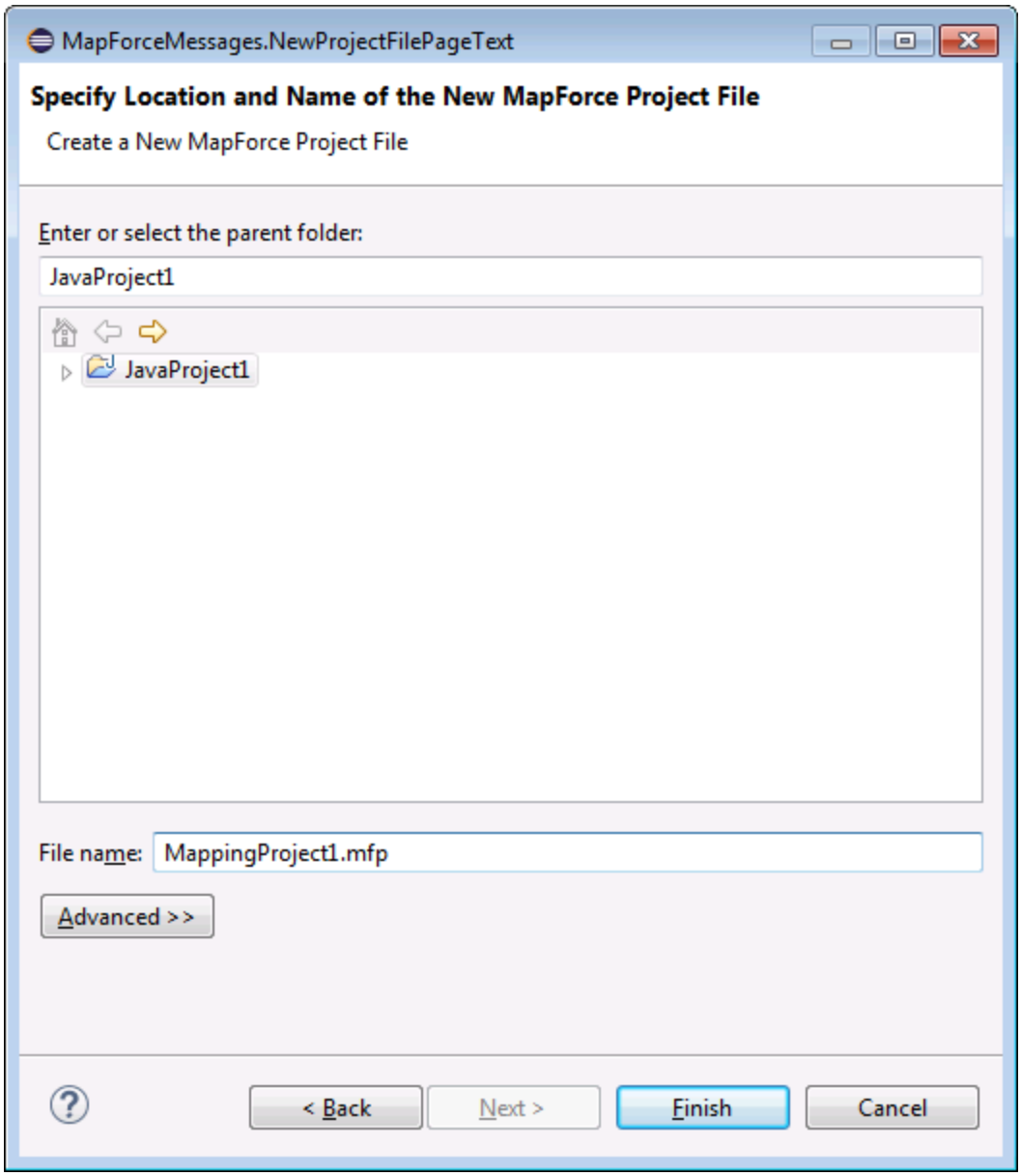
- mappages MapForce
- fichiers de projet MapForce
- projets de Service Web MapForce (disponible dans l'Édition MapForce Enterprise)

**Pour créer un de ces types de fichier dans le cadre d'un projet Eclipse :**

1. Créer un nouveau projet Eclipse ou ouvrir un projet existant.
2. Dans le menu **File**, cliquer sur **New**, puis cliquer sur **Other**.




3. Choisir le type de fichier requis depuis le dialogue d'assistant, puis cliquer sur **Next**.

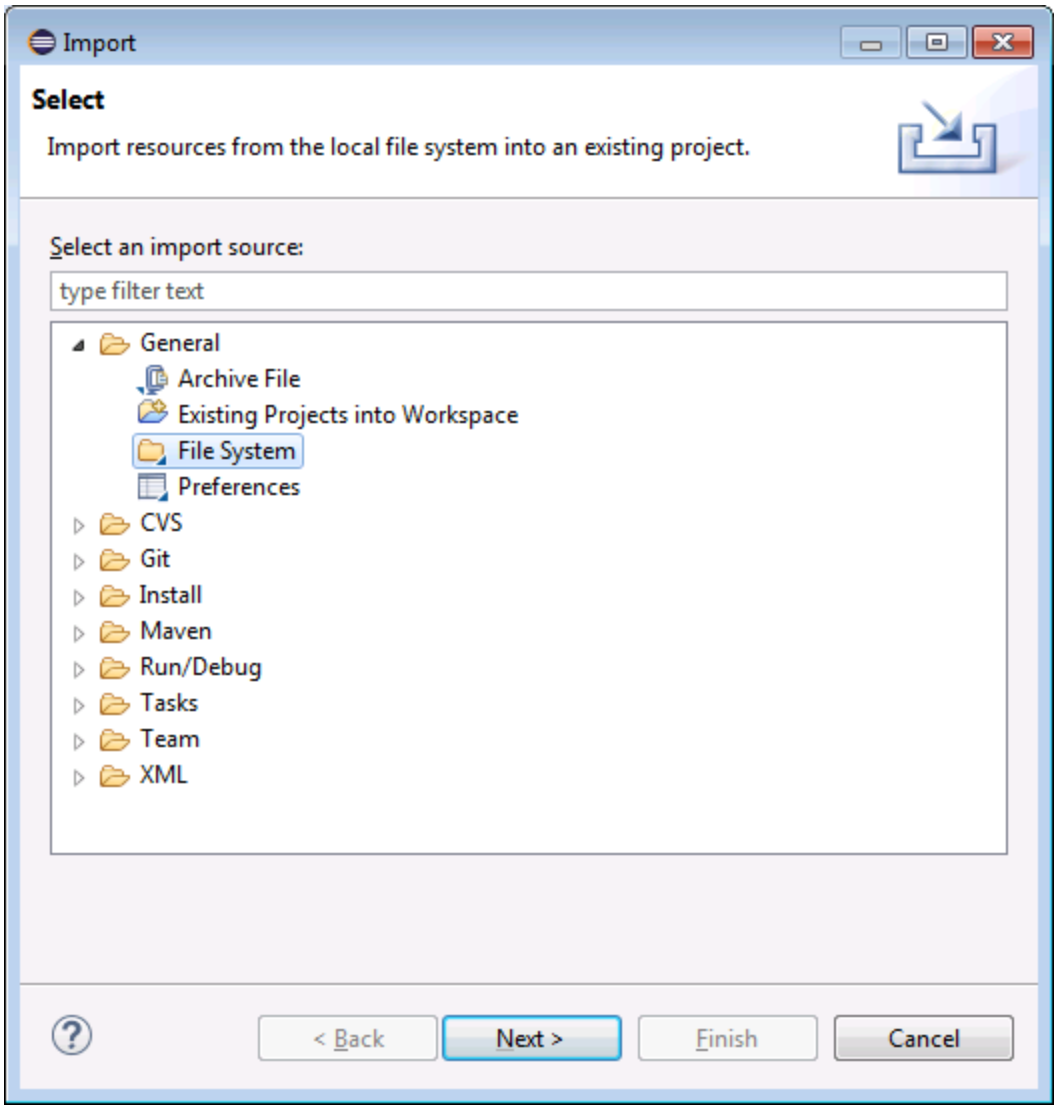



4. Choisir un dossier parent dans votre projet existant, et puis cliquer sur **Finish**.

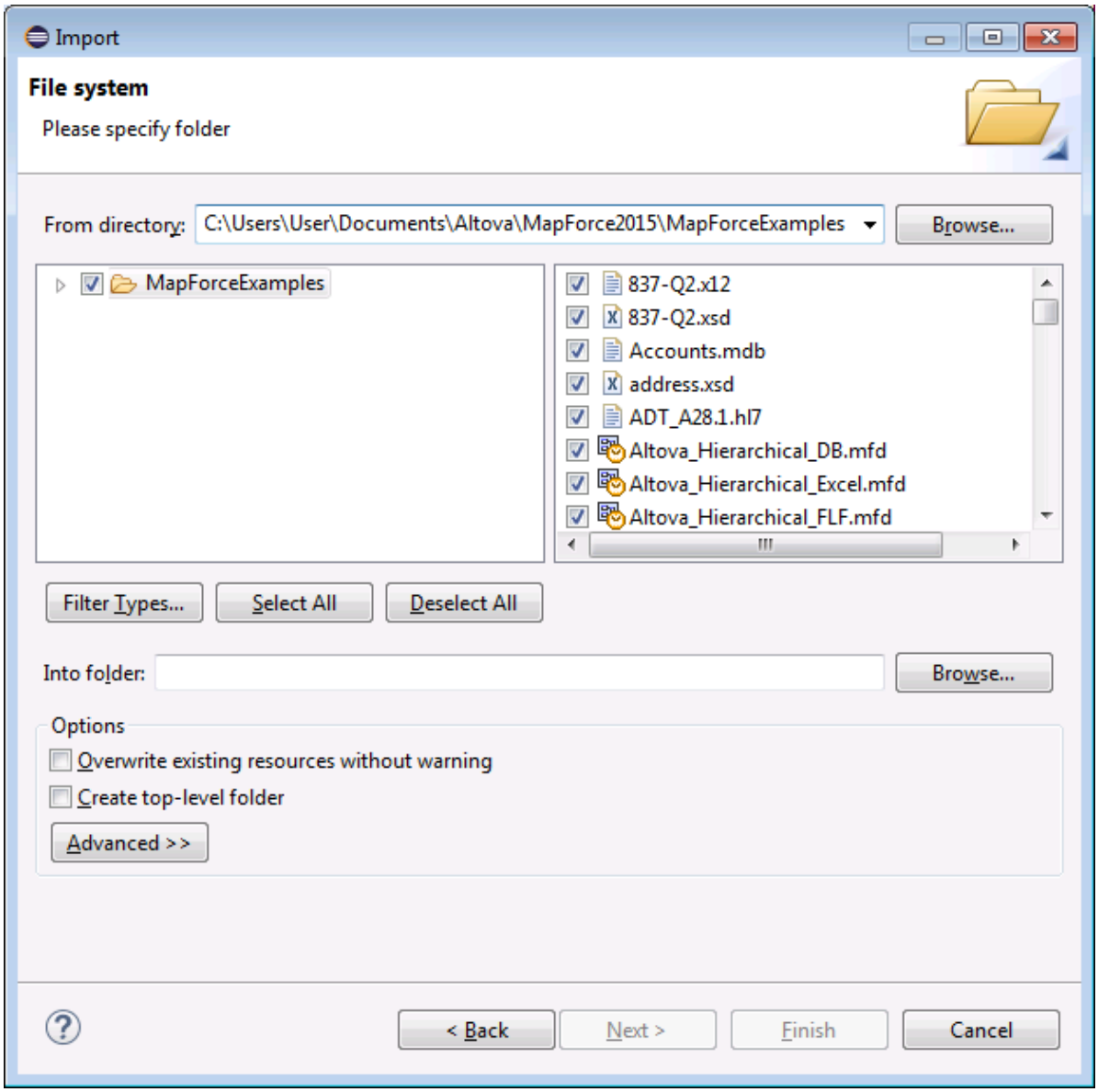
## 14.4.3 Importer des mappages existants dans un projet Eclipse

**Pour importer des mappages MapForce et leurs fichiers dépendants dans un projet Eclipse existants :**

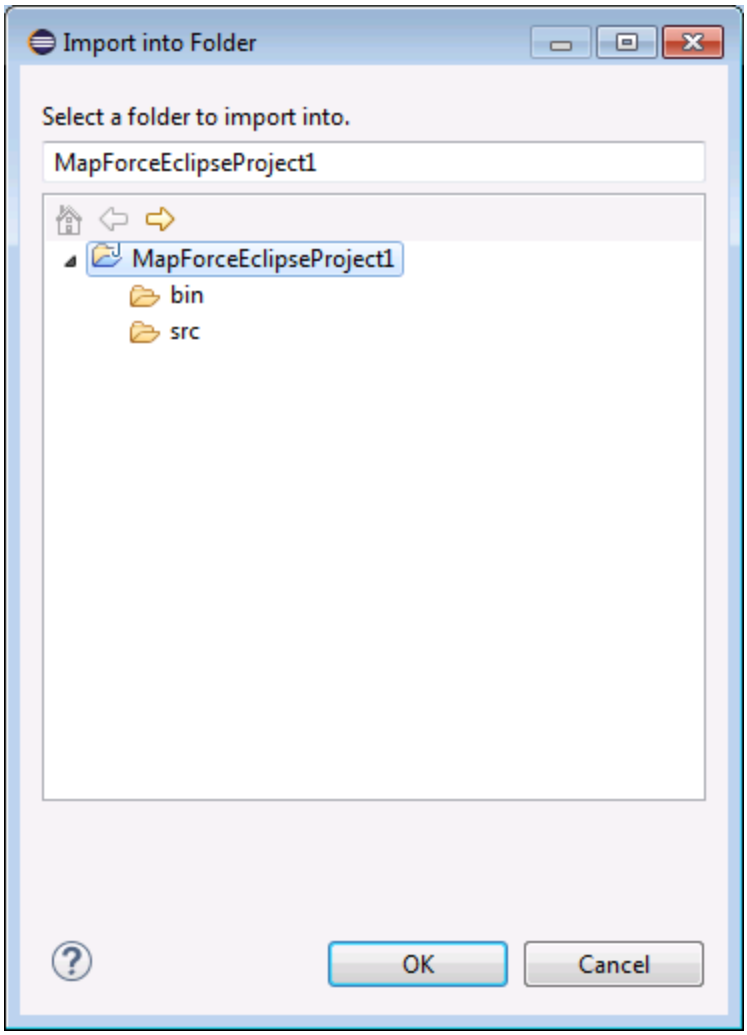
1. Ouvrir le projet dans lequel vous souhaitez importer les fichiers.
2. Dans le menu **File**, cliquer sur **Import**.




3. Sélectionner **File System**, puis cliquer sur **Next**.




4. À côté de **From directory**, chercher l'emplacement des fichiers que vous souhaitez importer, puis sélectionner les fichiers requis.
5. À côté de **Into folder**, cliquer sur **Browse**, puis sélectionner le projet dans lequel vous ajoutez les fichiers (dans cet exemple, *MapForceEclipseProject1*).




6. Cliquer sur **OK**, puis cliquer sur **Finish**.

## 14.4.4 Configurer un Build et une Génération automatique de code MapForce

La création et la génération de code MapForce automatique est activée par défaut dans tout projet MapForce/Eclipse (voir Créer un projet MapForce/Eclipse 924). Si vous souhaitez activer la création et la génération automatique du code MapForce dans un projet existant qui n'est pas de type *MapForce/Eclipse*, vous pouvez y parvenir en ajoutant manuellement le builder *MapForce Code Generation* et la nature *MapForce*.

**Pour ajouter le builder MapForce Code Generation à un projet :**

- Ajouter au fichier **.project** Eclipse les lignes marquées ci-dessous :

```
<buildSpec>
    <buildCommand>
```

```
        <name>org.eclipse.jdt.core.javabuilder</name>
        <arguments>
        </arguments>
      </buildCommand>
      <buildCommand>
        <name>com.altova.mapforceeclipseplugin.MapForceBuilder</name>
        <arguments>
        </arguments>
      </buildCommand>
   </buildSpec>
```

**Pour ajouter la nature MapForce à un projet :**

- Ajouter au fichier **.project** Eclipse les lignes marquées ci-dessous :

```
   <natures>
      <nature>org.eclipse.jdt.core.javanature</nature>
      <nature>com.altova.mapforceeclipseplugin.MapForceNature</nature>
   </natures>
```

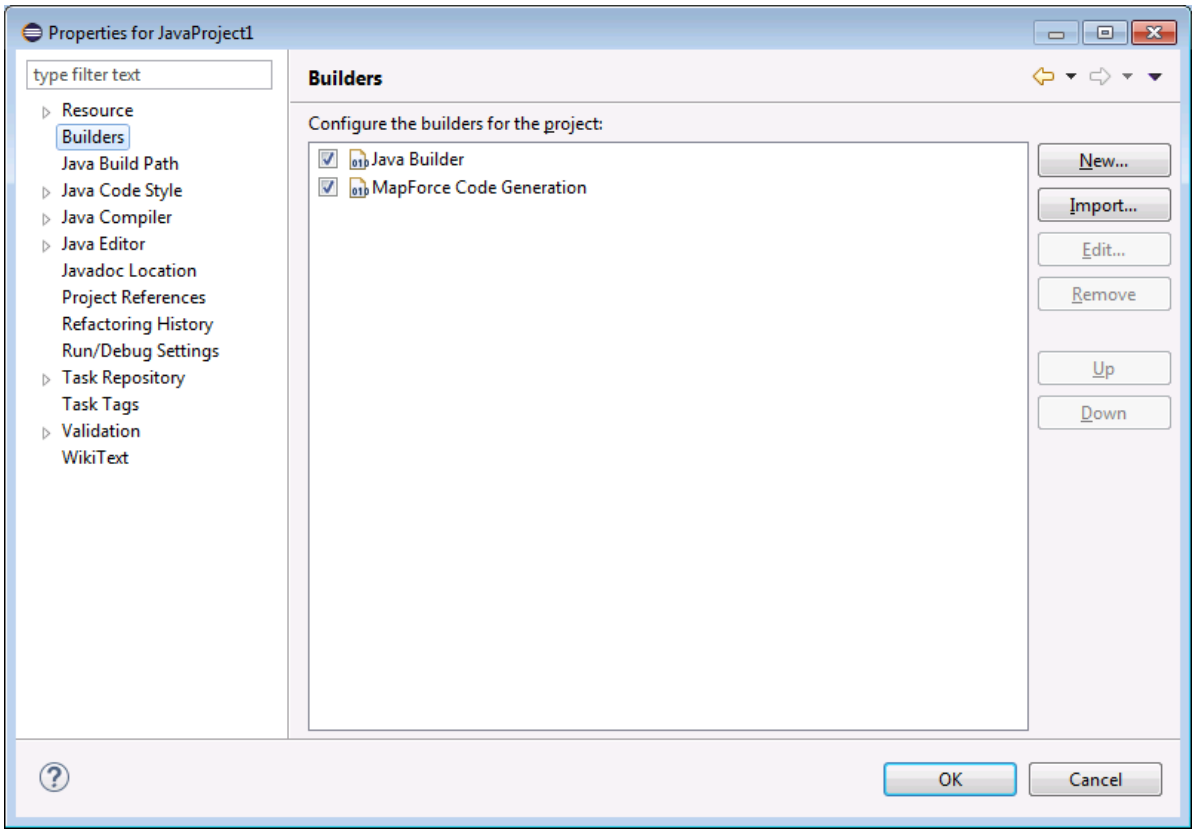
**Astuce :** Vous pouvez rapidement ouvrir le fichier **.project** depuis l'affichage Navigator d'Eclipse (Pour activer cet affichage, sélectionner la commande de menu **Window | Show View | Navigator**).

**Pour passer automatiquement la génération de code MapForce allumé/éteint :**

- Dans le menu **Project**, cliquer sur **Build automatically**.

**Pour désactiver le MapForce Code Generation builder:**

1. Dans le menu **Project**, cliquer sur **Properties**.
2. Cliquer sur **Builders**.




3. Cliquer pour décocher la case à cocher **MapForce Code Generation**.

# 14.5 Étendre le plug-in MapForce pour Eclipse

Le plug-in MapForce pour Eclipse fournit un point d'extension Eclipse avec l'ID **com.altova.mapforceeclipseplugin.MapForceAPI**. Vous pouvez utiliser cette extension pour adapter ou étendre les fonctions du plug-in MapForce. Le point d'extension vous donne accès à l'interface COM de la commande MapForce et de l'API MapForce 1107.

Le pack d'installation Eclipse MapForce contient un exemple simple d'un plug-in qui utilise ce point extension. Il vérifie dans les fichiers ouverts la présence d'événements de nouveaux mappages MapForce et définit le niveau de zoom de l'affichage de mappage à 70%.

La documentation JavaDoc du point d'extension est disponible dans le répertoire d'installation du plug-in MapForce (généralement, **C:\Program Files\Altova\MapForce2024\eclipse\docs**).

Avant d'installer et d'exécuter le plug-in MapForce échantillon, veuillez vous assurer que les exigences suivantes ont été remplies :

- Vous utilisez un pack d'intégration Java 64-bit, Eclipse 64-bit, MapForce 64-bit et MapForce 64-bit.
- Le plug-in JDT (Java Development Tools) est installé.
- Le PDE Eclipse (plug-in development environment) est installé.

### Pour importer le projet de plug-in MapForce échantillon dans votre espace de travail :

1. Lancer Eclipse.
2. Dans le menu **File**, cliquer sur **Import**.
3. Choisir **General | Existing projects into Workspace**, et cliquer sur **Next**.
4. Cliquer sur la touche **Browse**... située à côté du champ "'Select root directory" et choisir le répertoire de projet d'échantillon, par ex. **C:\Program Files\Altova\MapForce2024\eclipse\workspace\MapForceExtension.**
5. Choisir l'option **Copy projects into workspace**, et cliquer sur **Finish**. Un nouveau projet nommé "MapForceExtension" a été créé dans votre espace de travail.

### Pour exécuter le plug-in d'extension échantillon :

1. Passer à la perspective Java.
2. Dans le menu **Run**, cliquer sur **Run Configurations**.
3. Cliquer avec la touche de droite **Eclipse Application** et choisir **New**. (Si vous ne pouvez pas voir "Eclipse application" dans la liste, les Eclipse Plug-In Development Tools ne sont pas installés dans votre environnement Eclipse. Pour installer les Eclipse Plug-in Development Tools, cliquer sur **Install New Software** dans le menu **Help** et installer "Eclipse Plugin Development Tools" depuis le site de téléchargement "The Eclipse Project Updates".)
4. Saisir un nom pour votre nouvelle configuration (dans cet exemple, *SampleMapForcePlugin*), puis cliquer sur **Apply**.
5. Vérifier que le plug-in d'espace de travail **MapForceClient** est sélectionné dans l'onglet 'Plug-ins'.
6. Cliquer sur **Run**. Un nouveau Eclipse Workbench s'ouvre.
7. Ouvrir un mappage MapForce dans le nouveau Workbench. Il s'ouvrira avec un niveau de zoom par défaut de 70%.

# 15 Générateur de Code

Le générateur de code est une fonction MapForce intégrée qui vous permet de générer du code depuis des fichiers de mappage. Le résultat est une application complète proposant toutes les fonctions qui effectue l'opération de mappage pour vous. Une fois que vous générez le code, vous pouvez exécuter le mappage en exécutant l'application directement lors de la génération. Vous pouvez aussi importer le code généré dans votre propre application ou y ajouter vos propres fonctions.

*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<br><br>Frameworks cible :<br>• .NET Framework<br>• .NET Core 3.1<br>• NET 5.0<br>• NET 6.0<br>• NET 8.0 | Java SE JDK 8, 11, 17, 21 (y compris OpenJDK)<br>Eclipse 4.4 ou plus<br>Apache Ant (fichier build.xml) |
| **XML DOM implementations** | MSXML 6.0<br>Apache Xerces 3 | System.Xml | JAXP |
| **Database API** | ADO | ADO.NET | JDBC |

**Note:** Le code généré par MapForce peut être considéré thread-safe uniquement si le DOM XML tiers sous-jacent et les bibliothèques API de base de données le sont. Bien que la sécurité thread du code généré ne peut pas vraiment être prouvée ou garantie, il est probable que, dans la plupart des cas, plusieurs instances concurrentes du code de mappage pourront être exécutées avec succès.

*C++*
Vous pouvez configurer si vous souhaitez que la sortie générée C++ doit utiliser MSXML 6.0 ou Apache Xerces 3. MapForce génère des fichiers complets de projet (**`.vcproj`**) et de solution (**`.sln`**) pour toutes les versions prises en charge de Visual Studio (*voir table ci-dessous*). Le code généré prend en charge MFC on option.

Notez les exigences préalables suivantes :

- Pour compiler le code C++ généré, Windows SDK doit être installé sur votre ordinateur.
- Afin de pouvoir utiliser Xerces 3 pour C++, vous devrez l'installer et le construire utilisant la les instructions sous Apache Xerces page. Veuillez vous assurer d'ajouter la variable d'environnement XERCES3 qui pointe vers le répertoire dans lequel Xerces a été installé (par ex., `C:\xerces-c-3.2.2`). De même, la variable d'environnement PATH doit contenir le chemin dans lequel les binaires Xerces se trouvent ex.,`%XERCES3%\bin`).
- Lorsque vous construisez le code C++ pour Visual Studio et utilisez une bibliothèque Xerces précompilée pour Visual C++, vous devrez modifier le paramètre compilateur dans tous les projets de la solution. Suivez les étape ci-dessous :

a) Choisir tous les projets dans le Solution Explorer.
b) Cliquez sur **Properties** dans le menu **Projet**.
c) Cliquer sur **Configuration Properties | C/C++ | Language**.
d) Dans la liste des configurations, choisir *All Configurations*.
e) Modifier *Treat wchar_t as Built-in Type* en *No (/Zc:wchar_t-)*

*C#*
Le code C# généré peut être utilisé depuis tout langage de programmation compatible avec .NET, comme VB.NET, Managed C++ ou J#. Les fichiers de projet peuvent être générés pour toutes les versions prises en charge de Visual Studio (*voir table ci-dessus*).

*Java*
La sortie Java générée est écrite en conformité avec l'API Java de standard industriel pour XML Processing (JAXP) et comprend un fichier build Ant et des fichiers de projet pour les versions prises en charge de Java et Eclipse (*voir table ci-dessus*).

## Générer, créer, exécuter et intégrer du code

Pour les instructions relatives à la génération, construction et exécution du code, voir Générer, Créer, et Exécuter le code[937]. Pour les détails sur l'intégration du code généré par MapForce dans votre code personnalisé, voir Intégrer le code généré[943].

## Modèles de génération de code

Le code généré est créé via un modèle qui est écrit dans un langage de modèle appelé SPL[1043] SPL (Spy Programming Language). Vous pouvez personnaliser le modèle utilisé pour code-generation. Ceci peut être utile, par exemple, quand vous voulez personnaliser votre code selon les conventions écrites de votre entreprise ou remplacer les bibliothèques spécifiques dans le code généré.

## Exemples

Pour des exemples illustrant les capacités de génération de code, voir Exemple : Book Library[966] et Example: Purchase Order[990].

# 15.1 Générer, créer, exécuter du code

Ce chapitre décrit comment générer du code depuis un mappage et un projet, créer le code généré et l'exécuter. Il existe des situations pour lesquelles vous allez devoir modifier votre code C#/C++/Java généré pour l'intégrer dans votre code personnalisé. Pour les détails, voir Intégrer le code généré 943.

## 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 79. Pour les détails sur les paramètres de génération de code dans le dialogue **Options**, voir Génération 1089.
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 30.

## 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 1362).

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 la 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. Pour plus d'information sur les projets et paramètres liés au projet ainsi que les procédures, voir Projets 82.

## Information spécifique au langage

Cette sous-section décrit les particularités de générer du code dans différents langages de transformation. Cette sous-section explique comment créer le code C++, C# et Java généré et exécuter l'application. Vous pouvez aussi générer du code dans XSLT 1-3 et XQuery. Pour les détails, voir Génération de code 72.

*code C++ et C#*
Générer, créer et exécuter le code C++ et C# suivent la même logique. Les procédures générales sont décrites dans les sous-sections ci-dessous.

Une fois que vous avez généré le code C++ or C# code, la solution inclura les composants suivants :

- Les fichiers Solution (**.sln**) et Project (**.vcxproj** pour C++ et **.csproj** for C#) peuvent être ouverts dans Visual Studio
- Plusieurs bibliothèques signées Altova requises par le mappage (toutes préfixées avec `Altova`)
- Le projet de mappage principal (appelé `Mappage` par défaut), qui comprend l'application de mappage et des fichiers dépendants.

Notez que vous pouvez changer le nom par défaut du projet de mappage principal dans la boîte de dialogue Paramètres de mappage 79.

Une fois que vous avez généré le code C++/C#, les prochaines étapes seraient de créer le code et d'exécuter l'application. Il existe deux approches majeures pour créer le code généré : (i) dans Visual Studio et (ii) à la ligne de commande (*voir les détails ci-dessous*). Notez que, pour créer le code C#, vous devez avoir installé le SDK pertinent et une version de Visual Studio compatible. Pour le pack de téléchargement pour votre système d'exploitation et plateforme référez-vous au site web de Microsoft.

***Construire le code généré dans Visual Studio***
Pour créer le code généré C++/C#, suivez les instructions ci-dessous :

1. Ouvrir le fichier de solution généré (**.sln**) dans Visual Studio. Par défaut, le nom du fichier de solution est **Mapping.sln**, et est situé dans le sous-répertoire **Mappage** relatif au répertoire avec le code généré.
2. Choisir la configuration de build requise (par ex., Déboguer). Veuillez noter pour le code C++ : les builds Unicode soutiennent l'ensemble des caractères Unicode dans XML et d'autres fichiers. Les builds non-Unicode fonctionnent avec la page de code locale de votre installation Windows.
3. Cliquez sur **Créer Solution** dans le menu **Build**.

En conséquence, une fois le code créé, l'application command-line appelée **Mapping.exe** et ses fichiers y associés seront créés. L'application du mappage est située dans l'un des sous-répertoires relatifs au fichier **.sln**. Le nom du sous-répertoire dépend de la configuration créée sélectionnée : par ex., **Debug** (C++), **bin\Debug** (C#).

***Construire le code généré à la ligne commande***
Pour construire le code généré à la ligne de commande, basculez vers le répertoire avec le code généré et exécutez la commande suivante :

```
devenv Mapping.sln /Build "Debug|AnyCPU" /Project Mapping
```

La commande appelle Visual Studio et précise le nom du fichier de la solution à créer (**`Mapping.sln`** dans notre cas), le configuration désirée (Debug et tout CPU dans notre cas), et le nom du projet auquel appartient le fichier de la solution (`Mapping`). En conséquence, une fois le code créé, l'application command-line appelée **`Mapping.exe`** et ses fichiers y associés seront créés. L'application du mappage est située dans l'un des sous-répertoires relatifs au fichier **`.sln`**. Le nom du sous-répertoire dépend de la configuration créée sélectionnée : par ex., **`Debug`** (C++), **`bin\Debug`** (C#).

*Notes relatives à la construction du code C#*
Vous pouvez non seulement appeler Visual Studio, vous pouvez aussi appeler .NET pour construire le C# code généré. Suivez les instructions ci-dessous :

1. Assurez-vous de sélectionner la plate-forme correcte à cibler dans la section *Génération* du dialogue des **Options** (menu **Outils**). Pour les détails, voir Génération[1089].
2. Si vous ciblez .NET/.NET Core, configurez les variables d'environnement PATH et DOTNET_ROOT pour qu'elles pointent vers l'emplacement où .NET/.NET Core est installé. Ceci empêchera de possibles problèmes avec les commandes CLI.
3. Ouvrez une invite de commande et basculez vers le répertoire avec le code généré.
4. Exécuter la commande suivante :

   ```
   dotnet build Mapping\Mapping.sln --configuration Release
   ```

   Cette commande appelle .NET pour créer un fichier de solution appelé `Mapping.sln` avec la configuration `Release` et crée un programme d'exécution appelé **`Mapping.exe`** et ses fichiers associés dans le dossier de destination. Le nom du dossier de destination dépend de la configuration sélectionnée et la version .NET utilisée. Dans notre exemple, le programme d'exécution sera enregistré dans le dossier **`bin\Release\net8.0`**.

***Exécuter application***
Après avoir créé le code dans Visual Studio directement ou dans la ligne de commande, vous pouvez procéder à l'exécution de l'application. Pour exécuter l'application, double-cliquez sur **`Mapping.exe`** ou appelez le programme d'exécution depuis la ligne de commande. Une fois que vous avez exécuté le programme d'exécution, le résultat de la transformation du mappage sera la sortie du dossier de destination (par défaut, ceci est le dossier où l'exécutable est stocké).

Si vous construisez le code généré sur Linux, le programme d'exécution généré sera appelé `Mappage`, sans extension. Pour exécuter le programme d'exécution, vous aurez besoin de la commande suivante :

```
./Mappage
```

*Générer du code Java*
Lorsque vous générez un code Java, le projet Java inclura les composants suivants :

- Plusieurs packs Altova signés requis par le mappage (tous les préfixes avec `com.altova`)
- Le pack `com.mapforce` comprenant l'application de mappage et les fichiers dépendants, parmi lesquels les deux fichiers les plus importants qui précisent les points d'entrée de l'application sont les suivants :
  - L'application de mappage Java comme l'application du dialogue (`MappingApplication.java`)
  - L'application de mappage Java en tant qu'application de console (`MappingConsole.java`)
- Le fichier `build.xml` que vous pouvez exécuter avec Apache Ant pour compiler le projet et générer les fichiers JAR

Les noms par défaut de l'application de mappage et ses fichiers dépendants dans le pack `com.mapforce` sont préfixés avec **`Mappage`**. Vous pouvez changer ceci et d'autres paramètres dans la boîte de dialogue Paramètres de mappage 79.

Une fois que vous avez généré le code Java, les prochaines étapes seraient de créer et d'exécuter le code. Il existe deux approches majeures pour créer le code généré et exécuter l'application : (i) dans Eclipse et (ii) à la ligne de commande, utilisez Apache Ant. Les procédures générales sont décrites dans les sous-sections ci-dessous.

***Construire le code généré et exécuter l'application dans Eclipse***
Cette approche utilise le flux de travail d'Eclipse. Notez les exigences préalables suivantes :

- Installer le Java Development Kit (JDK) , Eclipse, et Apache Ant doit être installé sur votre système. Eclipse contient généralement une version bundled d'Ant, mais vous pouvez aussi installer Ant séparément.
- Pour exécuter Eclipse avec OpenJDK, la variable d'environnement PATH exige généralement l'inclusion du chemin dans le répertoire bin JDK **`bin`** (par exemple **`C:\Java\jdk-11.0.1\bin`**).
- La variable d'environnement JAVA_HOME doit pointer vers le répertoire d'installation JDK.
- La variable d'environnement ANT_HOME doit pointer vers l'emplacement d'Apache Ant.

Une fois que vous avez généré du code Java, la prochaine étape est d'importer le code Java généré dans Eclipse. Suivez les étape ci-dessous :

1. Dans le menu **File**, cliquez avant sur **Import** et sélectionnez **General | Existing Projects into Workspace**.
2. Cliquez sur **Suivant**.
3. Fournissez le chemin au code généré et cliquez sur **Finish**. Le projet Java créé par MapForce est maintenant disponible dans le mode Package Explorer. Si vous ne pouvez pas voir le mode Package Explorer, utilisez la commande de menu **Window | Show View | Package Explorer**.

Notez que, par défaut, le code est créé automatiquement à chaque fois qu'un changement est détecté. Vous pouvez aussi désactiver la fonction et créer un code, le cas échéant (voir la documentation Eclipse pour les détails).

Une fois que vous avez importé et créé votre code, la prochaine étape est d'exécuter l'application. Ce chapitre discute certaines des approches possibles pour exécuter l'application.

*Approche 1 : Exécuter le projet en tant qu'application*
Cette méthode vous permet d'exécuter votre projet Java comme une application GUI. Suivez les étapes ci-dessous :

1. Dans le mode Package Explorer d'Eclipse, cliquez avec la touche de droite sur le fichier **`MappingApplication.java`** dans le package **`com.mapforce`**.
2. Sélectionnez **Run As | Java application** depuis le menu contextuel.
3. Dans la fenêtre d'application MapForce qui apparaît, cliquer sur **Start** pour exécuter le mappage.

*Approche 2 : Exécuter un projet en tant qu'application de console*
Cette méthode vous permet d'exécuter votre projet Java comme une application (command-line) de console. Suivez les étapes ci-dessous :

1. Dans le mode Package Explorer d'Eclipse, cliquez avec la touche de droite sur le fichier **MappingConsole.java** dans le package **com.mapforce**.
2. Sélectionnez **Run As | Java application** depuis le menu contextuel.

En conséquence, après avoir exécuté l'application, l'application Java exécutera la transformation du mappage et générera un/des fichier/s de sortie dans le dossier de destination.

***Construire le code généré et exécuter l'application à la ligne de commande***

Pour pouvoir créer et exécuter le code généré à la ligne de commande, vous devez avoir les composants suivants installés et définir les variables d'environnement :

- Installer le Java Development Kit (JDK) et Apache Ant sur votre système.
- L'emplacement du répertoire Ant **bin** (par ex., **C:\apache-ant-1.10.5\bin**) devrait être ajouté à la variable d'environnement PATH. Ceci permettra d'exécuter Ant de manière pratique sans avoir à taper tout le chemin dans le programme d'exécution dans la ligne de commande.
- La variable d'environnement JAVA_HOME doit pointer vers l'emplacement de JDK.
- La variable d'environnement ANT_HOME doit pointer vers l'emplacement d'Apache Ant.

Pour construire le code généré avec Apache Ant, suivez les instructions suivantes :

1. Ouvrez une invite de commande et basculez vers le répertoire où le code généré, y compris le fichier build (**build.xml**), est stocké.
2. Exécuter la commande suivante :

   ```
   ant jar
   ```

   Cette commande créera le code généré et crée un fichier JAR (appelé **Mapping.jar** par défaut). L'objet du fichier JAR est de packager les fichiers Java **.class** et leurs métadonnées et ressources. Dans notre cas, l'archive JAR est prévue être utilisée comme programme Java exécutable ; pour cette raison, le fichier manifeste de l'archive inclut le point d'entrée de l'application (par par défaut, `com.mapforce.MappingConsole`).

Pour exécuter l'application Java, exécutez la commande suivante dans le répertoire où l'archive JAR est située :

```
java com.mapforce.MappingConsole Mapping.jar
```

Cette commande lance Java Virtual Machine, lance la classe principale appelée `com.mapforce.MappingConsole`, qui se réfère au point d'entrée de l'application Java, et exécute le programme contenu dans le fichier JAR appelé **Mapping.jar**. En conséquence, l'application Java exécutera la transformation du mappage et générera un/des fichier/s de sortie dans le dossier de destination. Si vous voulez lancer l'application comme application GUI, passez la valeur suivante pour l'argument de classe principal : `com.mapforce.MappingApplication`. Ceci ouvrira une fenêtre pop-up dans laquelle vous pourrez démarrer la transformation du mappage.

***Éviter certains problèmes possibles « out-of-memory »***

Les mappages complexes avec des schémas larges peuvent produire une grande quantité de code, qui peut entraîner une exception `java.lang.OutofMemory` pendant la compilation dans Ant. Éviter certains problèmes possibles « out-of-memory », suivez les étapes ci-dessous :

1. Ajouter la variable la variable d'environnement `ANT_OPTS`, qui définit des options Ant spécifiques comme la mémoire à attribuer à l'ordinateur, et configurer sa valeur comme suit : `-server -Xmx512m -Xms512m`.
2. Pour vous assurer que le compileur et le code généré soient exécutés dans le même processus qu'Ant, changer l'attribut `fork` dans **`build.xml`** en `false`.

Vous devrez éventuellement personnaliser les valeurs selon la quantité de mémoire disponible dans votre appareil et la taille du projet sur lequel vous travaillez. Pour plus de détails, voir votre documentation Java VM.

Lorsque vous exécutez la commande `ant jar`, vous recevrez éventuellement un message d'erreur de type "`[...] archive contains more than 65535 entities`". Afin d'éviter cela, il est recommandé d'utiliser Ant 1.9 ou plus et d'ajouter **`zip64mode="as-needed"`** dans l'élément `<jar>` dans `build.xml`.

***Prévenir des problèmes possibles avec les connexions JDBC***

Si vous avez généré le code Java depuis un mappage qui se connecte à une base de données par le biais de JDBC, vous allez avoir besoin d'ajouter le pilote JDBC comme entrée classpath à la configuration actuelle. Autrement, exécuter l'application pourrait engendrer une erreur. Pour les détails, voir Bases de données 160.

## 15.2 Intégrer du code généré

Même si le résultat de la génération de code est terminé et l'application entièrement fonctionnelle, vous allez devoir adapter le code généré par MapForce pour être capable de l'intégrer dans votre code personnalisé. Certains scénarios de modification du code incluent les points suivants :

- Modifier les fichiers source et cible pour l'application du mappage 943
- Définir un code de gestion d'erreur personnalisé 943
- Changer le type de données de l'entrée du mappage de manière programmatique dans C# - et le code - généré de Java (par exemple, du string au stream) 946
- Les bibliothèques de schema wrapper générées peuvent ensuite être intégrées dans votre application personnalisée pour lire, modifier ou écrire des documents XML par le biais d'un programme 953

### 15.2.1 Modifier Entrée/Sortie, définir la gestion d'erreur

Ce chapitre explique comment modifier les fichiers source et cible pour l'application de mappage dans le code Java, C#, et C++. Pour illustrer ces procédures, nous utilisons le mappage échantillon **`MapForceExamples\CompletePO.mfd`**. Le mappage consiste en trois composants source (**`ShortPO.xml`**, **`Customers.xml`** et **`Articles.xml`**) et un composant cible (**`CompletePO.xml`**).

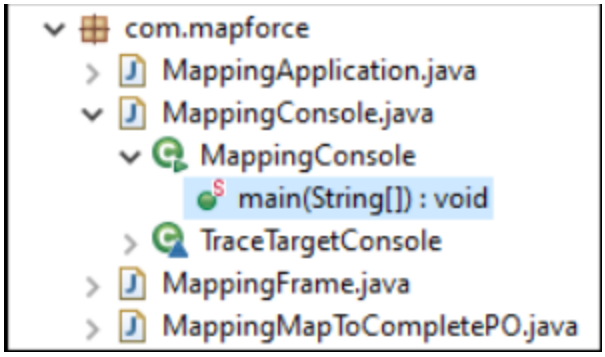
Dans le code généré, ces sources et ces cibles se traduiront en trois entrées et un paramètre de sortie fournis dans la méthode **`run`** qui exécute le mappage. Veuillez noter les points de base suivants concernant la génération de code :

- Le nombre de sources et de cibles dans le design de mappage correspond au nombre des paramètres de mappage dans la méthode **`run`** dans le code généré.
- Si vous changez le nombre de sources ou de cibles du mappage, vous devrez régénérer le code en conséquence.
- Si vous modifiez le code généré, puis régénérez le code dans le même emplacement, tous les changements seront écrasés.

#### Java

Cet exemple utilise Eclipse en tant que Java IDE. Pour commencer, générez du code Java depuis le mappage d'échantillon **`MapForceExamples\CompletePO.mfd`** puis importez le projet dans Eclipse. Pour information sur la génération du Java code et son importation dans Eclipse, voir Générer, créer, exécuter du code 939.

Pour éditer l'Application de la console Java générée, recherchez la méthode **`principale`** de votre application générée dans l'explorateur de votre projet (*voir la capture d'écran ci-dessous*). Par défaut, cette méthode est située dans la classe `MappingConsole` du package `com.mapforce`. Sinon, il se trouve dans la classe `MappingConsole` de votre package défini personnellement.




Pour éditer l'application de dialogue Java généré, situez l'endroit dans le code dans lequel la méthode **run** est invoquée depuis votre application générée. Par défaut, la méthode **run** est invoquée depuis la classe appelée `MappingFrame.java` du package `com.mapforce` (*capture d'écran ci-dessus*).

***Modifier les sources et cibles***
L'échantillon de code suivant illustre un extrait depuis la méthode**principale** dans l'application de console Java générée. Les sources et les cibles de chemin sont marquées en jaune et définies en tant que paramètres dans la méthode **run**. Si vous avez besoin de sources et/ou cibles, changez les valeurs des paramètres affichées ci-dessus.

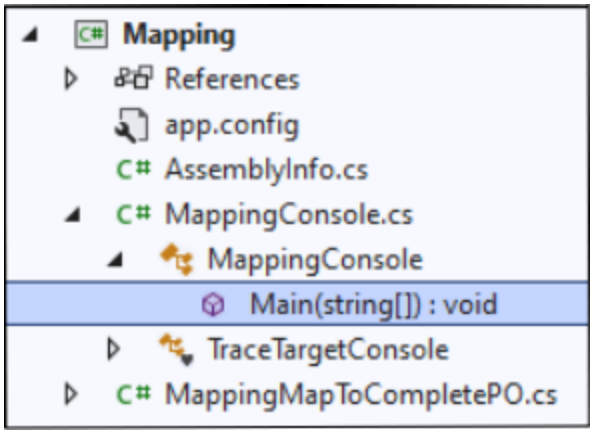
```
com.altova.io.Input Customers2Source =
com.altova.io.StreamInput.createInput("C:/Users/<UserName>/Documents/Altova/MapForce2024/
MapForceExamples/Customers.xml");
                    com.altova.io.Input Articles2Source =
com.altova.io.StreamInput.createInput("C:/Users/<UserName>/Documents/Altova/MapForce2024/
MapForceExamples/Articles.xml");
                    com.altova.io.Input ShortPO2Source =
com.altova.io.StreamInput.createInput("C:/Users/<UserName>/Documents/Altova/MapForce2024/
MapForceExamples/ShortPO.xml");
                    com.altova.io.Output CompletePO2Target = new
com.altova.io.FileOutput("CompletePO.xml");
```

***Définir la gestion d'erreur personnalisée***
Si vous avez besoin d'ajouter votre code personnalisé de gestion de l'erreur, modifiez l'instruction `catch` dans la méthode **principale** (application de console) ou dans `MappingFrame.java` (application GUI).

## C#

Cet exemple utilise Microsoft Visual Studio comme C# IDE. Pour commencer, générez C# depuis le mappage échantillon **MapForceExamples\CompletePO.mfd**, puis ouvrez la solution dans Visual Studio. Pour information sur la génération du code et son importation dans Visual Studio, voir Générer, créer, exécuter du code 938.

Pour éditer l'application de la C# générée, recherchez la méthode **principale** de votre application générée dans l'explorateur de votre Solution (*voir la capture d'écran ci-dessous*). Par défaut, le fichier de solution est appelé **Mapping.sln** et se trouve dans le sous-répertoire **Mapping** relatif au répertoire dans lequel vous avez enregistré le code généré.




***Modifier les sources et cibles***
La liste de code ci-dessous illustre un extrait depuis la méthode **`principale`** dans l'application de C# générée. Les sources et les cibles de chemin sont marquées en jaune et définies en tant que paramètres dans la méthode **`run`**. Si vous avez besoin de sources et/ou cibles, changez les valeurs des paramètres affichées ci-dessus.

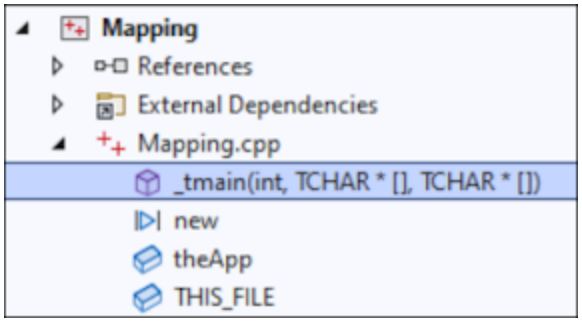
```
Altova.IO.Input Customers2Source =
Altova.IO.StreamInput.createInput("C:/Users/<UserName>/Documents/Altova/MapForce2024/MapF
orceExamples/Customers.xml");
                          Altova.IO.Input Articles2Source =
Altova.IO.StreamInput.createInput("C:/Users/<UserName>/Documents/Altova/MapForce2024/MapF
orceExamples/Articles.xml");
                          Altova.IO.Input ShortPO2Source =
Altova.IO.StreamInput.createInput("C:/Users/<UserName>/Documents/Altova/MapForce2024/MapF
orceExamples/ShortPO.xml");
                          Altova.IO.Output CompletePO2Target = new
Altova.IO.FileOutput("CompletePO.xml");
```

***Définir la gestion d'erreur personnalisée***
Si vous avez besoin d'ajouter votre code personnalisé de gestion de l'erreur, modifiez l'instruction `catch` dans la méthode **`principale`**.

## C++

Cet exemple utilise Microsoft Visual Studio comme C++ IDE. Pour commencer, générez C++ depuis le mappage échantillon **`MapForceExamples\CompletePO.mfd`**, puis ouvrez la solution dans Visual Studio. Pour information sur la génération du code et son importation dans Visual Studio, voir Générer, créer, exécuter du code 938.

Pour éditer l'application de la C++ générée, recherchez la méthode **`principale`** de votre application générée dans l'explorateur de Visual Studio (*voir la capture d'écran ci-dessous*). Par défaut, le fichier de solution est appelé **`Mapping.sln`** et se trouve dans le sous-répertoire **`Mapping`** relatif au répertoire dans lequel vous avez enregistré le code généré.




***Modifier les sources et cibles***
La liste de code ci-dessous illustre un extrait depuis la méthode **_tmain** dans l'application de C++ générée. Les sources et les cibles de chemin sont marquées en jaune et définies en tant que paramètres dans la méthode **Run**. Si vous avez besoin de sources et/ou cibles, changez les valeurs des paramètres affichées ci-dessus.

```
MappingMapToCompletePO MappingMapToCompletePOObject;
                    MappingMapToCompletePOObject.Run(

_T("C:/Users/<UserName>/Documents/Altova/MapForce2024/MapForceExamples/Customers.xml"),

_T("C:/Users/<UserName>/Documents/Altova/MapForce2024/MapForceExamples/Articles.xml"),

_T("C:/Users/<UserName>/Documents/Altova/MapForce2024/MapForceExamples/ShortPO.xml"),
              _T("CompletePO.xml"));
```

***Définir la gestion d'erreur personnalisée***
Si vous avez besoin d'ajouter votre code personnalisé de gestion de l'erreur, modifiez l'instruction `catch` dans la méthode **principale**.

## 15.2.2 Change Data Type of Input/Output

Un code généré par MapForce peut être intégré, ou adapté dans votre application spécifique, bien que le résultat de la génération de code est une application complète et entièrement fonctionnelle. Après avoir généré le code Java ou C# 937 avec MapForce, vous pouvez changer en option le type de données de l'entrée ou la sortie de mappage en éditant le code généré. Plus spécifiquement, vous pouvez utiliser les objets de paramètres de mappage des types autres que ceux générés par défaut. Par exemple, à la place que le mappage lise l'entrée d'un fichier sur le disque, vous pouvez fournir un string ou un objet stream comme entrée. Notez que cette fonction est spécifique au code généré dans C# ou uniquement Java.

Les types d'objet pris en charge comme entrée ou sortie sont recensés dans la première colonne de la table ci-dessous. Chaque colonne suivante précise les formats de données où ce type spécifique est pris en charge. Pour une définition plus précise de chaque type, voir la section « définition de types » ci-dessous.

| | XML | JSON* | Microsoft Excel* | EDI (inclut X12, HL7)* | FlexText* | CSV/Text |
|---|---|---|---|---|---|---|
| **Fichiers** | Oui | Oui | Oui | Oui | Oui | Oui |
| **Flux** | Oui | Oui | Oui | Oui | Oui | Oui |
| **Chaînes** | Oui | Oui | – | Oui | Oui | Oui |
| **Reader/Writer** | Oui | Oui | – | Oui | Oui | Oui |
| **Documents DOM** | Oui | – | – | – | – | – |

* *Formats pris en charge uniquement dans MapForce Enterprise Edition*

**Pour modifier le type de données d'une d'entrée ou sortie d'une fonction :**

1. Générer du code C# ou Java depuis un mappage.
2. Dans le code généré, trouvez l'appel dans la méthode **run** (dans Java) ou la méthode **Run** (dans C#), comme suit :

   a. Si vous utilisez C#, ouvrez le fichier **MappingConsole.cs**.
   b. Si vous utilisez Java, ouvrez **MappingConsole.java** (le programme de la console) ou le fichier **MappingFrame.java** (le programme GUI).

**Note :** le nom du fichier peut être différent si vous avez changé le nom de l'application en paramètres de mappage[79]. Par exemple, si vous le changez en « MyApp », le nom du fichier généré devient **MyAppConsole.js** et **MyAppConsole.java**, et **MyAppFrame.java**, respectivement.

3. Créez une instance du type requis (voir la section « définitions de type »).
4. Fournissez les objets déclarés comme paramètres dans la méthode **run** (dans Java) ou la méthode **Run** (dans C#), comme affiché dans l'exemple ci-dessous.

La méthode **run** est la méthode la plus importante des classes de mappage générées. Elle contient un paramètre pour chaque source *statique* ou composant d'entrée dans le mappage et un paramètre final pour le composant de sortie. Si votre mappage contient des composants qui traitent les multiples fichiers de manière dynamique[787], les paramètres respectifs n'apparaissent pas dans le code généré, car dans ce cas, les noms de fichiers sont traités de manière dynamique à l'intérieur du mappage.

## Définitions de type

Dans C#, les types que vous pouvez fournir en tant que paramètres dans la méthode **Run** sont des classes définies dans l'espace de noms `com.altova.io`). Les classes de base sont `Altova.IO.Input` et `Altova.IO.Output`, respectivement.

*Types C#*

| | |
|---|---|
| **Fichiers** | `Altova.IO.FileInput(nom de fichier String)`<br>`Altova.IO.FileOutput(nom de fichier String )` |

| | |
|---|---|
| **Flux** | `Altova.IO.StreamInput(System.IO.Stream stream)`<br>`Altova.IO.StreamOutput(System.IO.Stream stream)` |
| **Chaînes** | `Altova.IO.StringInput(contenu string)`<br>`Altova.IO.StringOutput(System.Text.StringBuilder sbuilder)` |
| **Reader/Writer** | `Altova.IO.ReaderInput(System.IO.TextReader reader)`<br>`Altova.IO.WriterOutput(System.IO.TextWriter writer)` |
| **Documents DOM** | `Altova.IO.DocumentInput(System.Xml.XmlDocument document)`<br>`Altova.IO.DocumentOutput(System.Xml.XmlDocument document)` |

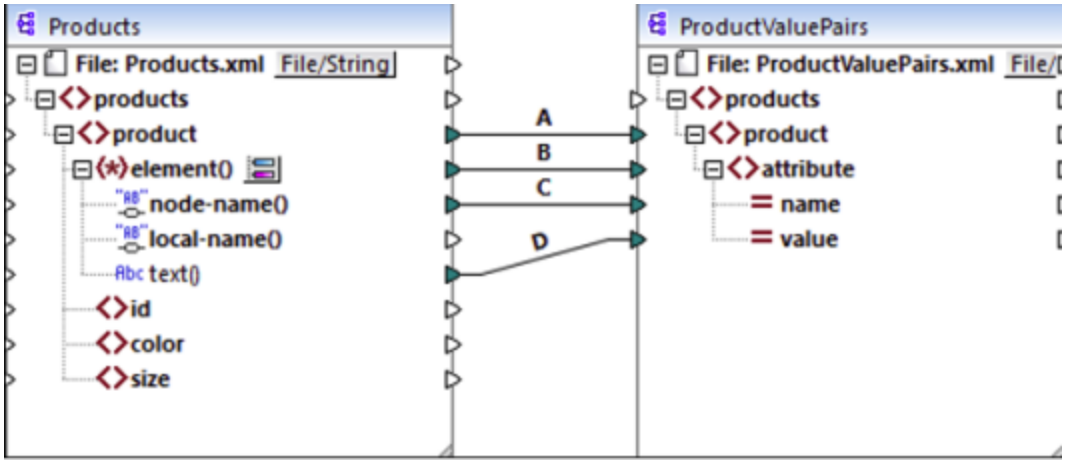
Dans Java, les types que vous pouvez fournir en tant que paramètres dans la méthode **`run`** sont des classes définies dans le package `com.altova.io`). Les classes de base sont `com.altova.io.Input` et `com.altova.io.Output`, respectivement.

*Type Java*

| | |
|---|---|
| **Fichiers** | `com.altova.io.FileInput(String filename)`<br>`com.altova.io.FileOutput(String filename)` |
| **Flux** | `com.altova.io.StreamInput(java.io.InputStream stream)`<br>`com.altova.io.StreamOutput(String filename)` |
| **Chaînes** | `com.altova.io.StringInput(String content)`<br>`com.altova.io.StringOutput()` |
| **Reader/Writer** | `com.altova.io.ReaderInput(java.io.Reader reader)`<br>`com.altova.io.WriterOutput(java.io.Writer writer)` |
| **Documents DOM** | `com.altova.io.DocumentInput(org.w3c.dom.Document document)`<br>`com.altova.io.DocumentOutput(org.w3c.dom.Document document)` |

## Exemple

Pour illustrer le changement de l'entrée et de la sortie par programmation, nous utiliserons le mappage **ConvertProducts.mfd** comme modèle. Après avoir installé et exécuté MapForce au moins une fois, vous pouvez trouver ce mappage dans le répertoire suivant : **C:\Users\<username>\Documents\Altova\MapForce2024\MapForceExamples\Tutorial.**

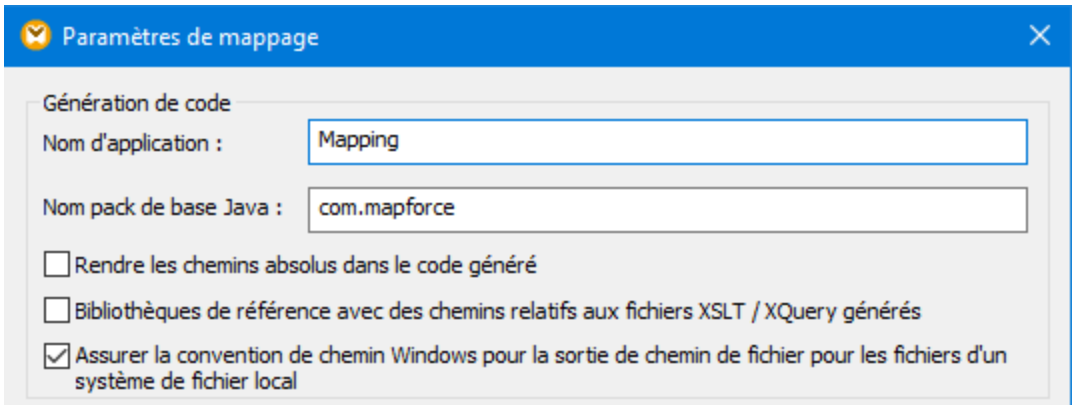



*ConvertProducts.mfd*

Tel qu'illustré ci-dessus, le mappage convertit les données depuis un document XML source vers un autre document XML. Nos objectifs sont les suivants :

1. Générer Java et le code de programme C# depuis ce mappage.
2. Changer le type de données du composant source en un type string.
3. Changer le type de données du composant cible en un type writer string.

Pour générer le code de programme, ouvrez le mappage **ConvertProducts.mfd** et exécutez la commande **Fichier | Générer code dans | C#** (ou **Java**). Pour l'exemple, nous partirons du fait que les paramètres de mappage de **ConvertProducts.mfd** sont les paramètres par défaut.




Cet exemple utilise les répertoires cible suivants pour le code généré (n'hésitez pas à modifier le chemin, le cas échéant) :

- **C:\codegen\cs\ConvertProducts**, for C#
- **C:\codegen\java\ConvertProducts**, for Java

Ayant généré le code de programme, ouvrez **MappingConsole.cs** (dans C#) ou **MappingConsole.java** (dans Java) et trouvez les lignes suivantes :

*C#*

```
Altova.IO.Input Products2Source = Altova.IO.StreamInput.createInput("Products.xml");
Altova.IO.Output ProductValuePairs2Target = new
Altova.IO.FileOutput("ProductValuePairs.xml");
```

*Java*

```
com.altova.io.Input Products2Source =
com.altova.io.StreamInput.createInput("Products.xml");
com.altova.io.Output ProductValuePairs2Target = new
com.altova.io.FileOutput("ProductValuePairs.xml");
```

Commenter les lignes ci-dessus et changez le code comme suit :

*C#*

```
//Altova.IO.Input Products2Source = Altova.IO.StreamInput.createInput("Products.xml");
//Altova.IO.Output ProductValuePairs2Target = new
Altova.IO.FileOutput("ProductValuePairs.xml");

Altova.IO.Input Products2Source = new Altova.IO.StringInput("<?xml version=\"1.0\"
encoding=\"UTF-8\"?>\r\n" +
                                    "<products
xmlns:xsi=\"http://www.w3.org/2001/XMLSchema-instance\"
xsi:noNamespaceSchemaLocation=\"products.xsd\">\r\n" +
                                    "   <product>\r\n" +
                                    "      <id>100</id>\r\n" +
                                    "      <color>blue</color>\r\n" +
                                    "      <size>XXL</size>\r\n" +
                                    "   </product>\r\n" +
                                    "</products>\r\n");

System.IO.StringWriter writer = new System.IO.StringWriter(new
System.Text.StringBuilder());
Altova.IO.Output ProductValuePairs2Target = new Altova.IO.WriterOutput(writer);

try
{
    MappingMapToProductValuePairsObject.Run(Products2Source, ProductValuePairs2Target);

    // Print out the writer object
    Console.Write(writer.ToString());
}
finally
{
    Products2Source.Close();
    ProductValuePairs2Target.Close();
}
```

*Java*

```
//com.altova.io.Input Products2Source =
com.altova.io.StreamInput.createInput("Products.xml");
//com.altova.io.Output ProductValuePairs2Target = new
com.altova.io.FileOutput("ProductValuePairs.xml");

com.altova.io.Input Products2Source = new com.altova.io.StringInput("<?xml
version=\"1.0\" encoding=\"UTF-8\"?>\r\n" +
               "<products xmlns:xsi=\"http://www.w3.org/2001/XMLSchema-instance\"
xsi:noNamespaceSchemaLocation=\"products.xsd\">\r\n" +
               "   <product>\r\n" +
               "      <id>100</id>\r\n" +
               "      <color>blue</color>\r\n" +
               "      <size>XXL</size>\r\n" +
               "   </product>\r\n" +
               "</products>\r\n");

java.io.StringWriter writer = new java.io.StringWriter();
com.altova.io.Output ProductValuePairs2Target = new com.altova.io.WriterOutput(writer);

try {
   MappingMapToProductValuePairsObject.run(Products2Source, ProductValuePairs2Target);

   // Print out the writer object
   System.out.print(writer.toString());

   } finally {
      (Products2Source).close();
      ProductValuePairs2Target.close();
}
```

Dans les listes de code C# et Java ci-dessus, les points suivants ont lieu :

- Les deux lignes originales qui fournissent l'entrée et la sortie à la méthode **run** sont commentées. En conséquence, l'application de mappage ne lit plus les données de **Products.xml**. En fait, nous ne devons pas copier ce fichier dans le répertoire de travail du programme.
- Le type `Products2Source` a été déclaré comme `StringInput` qui fournit le contenu du fichier d'entrée XML à être traité.
- Le type `ProductValuePairs2Target` a été déclaré comme type `WriterOutput` qui prend un « string writer » comme argument.
- Une fois que le mappage termine l'exécution, les contenus du « string writer » sont imprimés vers la console.

## Lignes directrices d'utilisation pour les streams et objets Reader/Writer

En utilisant les streams binaires ou les objets Reader/Writer comme entrée ou sortie au mappage, notez la chose suivante :

- Les objets stream binaires devraient être ouverts et prêts à être utilisés avant d'appeler la méthode **run**.
- Par défaut, la méthode **run** ferme le stream une fois la procédure terminée. Pour éviter ce comportement, insérer (ou décommentez) la ligne suivante avant d'appeler la méthode **run** :

*C#*

```
MappingMapToSomething.CloseObjectsAfterRun = false;
```

*Java*

```
MappingMapToSomething.setCloseObjectsAfterRun(false);
```

**Note :** Assurez-vous de changer `MappingMapToSomething` au nom de l'objet de mappage tel qu'applicable pour votre code généré.

## Lignes directrices d'utilisation pour strings

Dans Java, le constructeur de `StringOutput` ne prend pas d'argument. Le contenu du string produit par le mappage peut être accédé en appelant la méthode `getString()`, par exemple :

*Java*

```
com.altova.io.Input Products2Source =
com.altova.io.StreamInput.createInput("Products.xml");
com.altova.io.StringOutput ProductValuePairs2Target = new com.altova.io.StringOutput();

try {
   // Run the mapping
   MappingMapToProductValuePairsObject.run(Products2Source, ProductValuePairs2Target);
   // Get the string object
   String str = ProductValuePairs2Target.getString().toString();
}
```

Dans C#, le constructeur de `StringOutput` prend un type de paramètre `StringBuilder` que vous devez déclarer avant. Si l'objet `StringBuilder` contient déjà des données, la sortie de mappage y est ajoutée.

*C#*

```
Altova.IO.Input Products2Source = Altova.IO.StreamInput.createInput("Products.xml");
System.Text.StringBuilder sb = new System.Text.StringBuilder();
Altova.IO.Output ProductValuePairs2Target = new Altova.IO.StringOutput(sb);

try
{
   // Run the mapping
   MappingMapToProductValuePairsObject.Run(Products2Source, ProductValuePairs2Target);
   // Get the string output
   String str = sb.ToString();
}
```

Pour exécuter ces listes de codes, vous pouvez utiliser le même projet généré comme dans l'exemple précédent. Assurez-vous, toutefois, de copier le fichier **Products.xml** de **C:\Users\<username>\Documents\Altova\MapForce2024\MapForceExamples\Tutorials** dans votre répertoire de travail du programme, puisque le code de mappage lit les données depuis ce fichier.

### Lignes directrices d'utilisation pour les documents DOM

Lorsque vous utilisez les documents DOM comme entrée ou sortie de mappage, notez les points suivants :

- L'instance de document fournie comme paramètre au constructeur `DocumentOutput` doit être vide.
- Après avoir appelé **`run`**, le document DOM généré par le constructeur de `DocumentOutput` contient déjà la sortie de mappage, et vous pouvez manipuler le document, le cas échéant.

## 15.2.3 Générer du code depuis des Schémas XML ou des DTD

Lorsque vous générez du code depuis un mappage, MapForce génère une application complète qui exécute toutes les étapes du mappage automatiquement. En option, vous pouvez générer des bibliothèques pour tous les schémas XML utilisés dans le mappage. Celles-ci permettent à votre code de créer ou de lire facilement des instances XML qui sont utilisées ou créées par le code mappage.

Pour générer des bibliothèques pour tous les schémas XML utilisés dans le mappage, cocher la case **Generate Wrapper Classes** dans les options du générateur de code[1089]. La prochaine fois que vous générez du code, MapForce créera non seulement une application de mappage, mais aussi des classes de wrapper pour tous les schémas utilisés dans le mappage, comme suit :

| C++ ou C# | Java | Objectif |
|---|---|---|
| Altova | `com.altova` | Bibliothèque de base contenant un support de runtime commun, à l'identique pour chaque schéma. |
| AltovaXML | `com.altova.xml` | Bibliothèque de base contenant un support de runtime pour XML, à l'identique pour chaque schéma. |
| *[YourSchema]* | `com.YourSchema` | Une bibliothèque contenant des déclarations générées depuis le schema d'entrée, nommée comme le fichier de schéma ou DTD. Cette bibliothèque est un DOM (W3C Document Object Model) wrapper qui vous permet de lire, modifier ou créer des documents XML facilement et en toute sécurité. Toutes les données sont contenues à l'intérieur du DOM, et il existe des méthodes pour les données d'extraction depuis le DOM, et pour mettre à jour et créer des données dans le DOM.<br><br>Le code généré C++ prend en charge soit Microsoft MSXML ou Apache Xerces 3. La syntaxe pour utiliser le code généré est généralement semblable pour les deux mises en œuvre DOM, à l'exception de quelques légères différences (par exemple, Xerces prend en charge plus de fonctions surchargées).<br><br>Le code C# généré utilise la bibliothèque .NET standard System.XML comme mise en œuvre sous-jacente de DOM. |

| | | |
|---|---|---|
| | | Le code Java généré utilise JAXP (Java API pour le traitement XML) comme interface sous-jacente DOM. |

Tout en prototypant une application depuis un schéma XML changeant fréquemment, vous devrez éventuellement générer fréquemment du code dans le même répertoire, de manière à ce que les changements de schéma sont réfléchis immédiatement dans le code. Veuillez noter que l'application de test généré et les bibliothèques Altova sont écrasées à chaque fois que vous générez du code dans le même répertoire cible. C'est pourquoi il ne faut pas ajouter du code à l'application de test généré. Au lieu de cela, veuillez intégrer les bibliothèques dans votre projet (voir Intégrer des Bibliothèques Schema Wrapper 963).

En plus des bibliothèques de base ci-dessus, quelques bibliothèques sont également générées. Les bibliothèques prises en charge sont utilisées par les bibliothèques de base d'Altova et ne sont pas prévues pour des intégrations personnalisées, puisqu'elles sont sujettes au changement.

## Génération de nom et espaces de nom

MapForce génère des classes correspondant à tous les éléments déclarés ou types complexes qui redéfinissent tout type complexe de votre schéma XML, préservant la dérivation de classe, telle que définie par les extensions des types complexes dans votre schéma XML. Dans le cas de schémas complexes qui importent des composants de schéma de multiples espaces de noms, MapForce préserve cette information en générant les espaces de noms C# or C++ appropriés ou les packs Java.

Généralement, le générateur de code tente de préserver les noms pour les espaces de noms générés, classes et membres depuis le schéma XML d'origine. Les caractères qui ne sont pas valides en identifiants dans le langage cible sont remplacés par a "_". Les noms qui qui entreraient en collision avec les autres noms ou mots réservés sont rendus uniques en ajoutant un numéro. La génération de nom peut être influencée en changeant les paramètres par défaut dans le modèle SPL 1043.

Les espaces de noms depuis le schéma XML sont convertis en packs dans Java ou espaces de noms dans le code C# or C++, utilisant le préfixe d'espace de noms depuis le schéma comme espace de nom du code. La bibliothèque complète est intégrée dans un pack pu appareil de noms dérivé depuis le nom de fichier du schéma, donc vous pouvez utiliser le multiples bibliothèques générées dans un programme sans conflits de noms.

## Types de données

Le Schéma XML a un modèle de type de données plus élaboré que Java, C# ou C++. Le générateur de code convertit les types de schéma XML intégrés en des types primitifs spécifique au langage, ou pour des classes délivrées avec la bibliothèque d'Altova. Des types complexes et types dérivés dans le schéma sont convertis en classes dans la bibliothèque générée. Les facettes d'énumération de types simples sont converties en constantes symboliques.

Le mappage de types simples peut être configuré en modèle SPL, voir Référence SPL 1043.

Si vos fichiers d'instance XML utilisent vos types de schéma associés au temps et à la durée, ceux-ci sont convertis en classes natives d'Altova dans le code généré. Pour plus d'informations concernant les classes de bibliothèque, voir :

- Référence aux classes générées (C++) 997

- Référence aux classes générées (C#) 1013
- Référence aux classes générées (Java) 1028

Pour des informations sur le type de conversion et autres détails applicables à chaque langage, voir :

- À propos des Schema Wrapper Libraries (C++) 956
- À propos des Schema Wrapper Libraries (C#) 959
- À propos des Schema Wrapper Libraries (Java) 961

## Gestion de la mémoire

Une arborescence DOM est comprise de nœuds, qui sont toujours la propriété d'un document DOM spécifique - même si le nœud ne fait pas actuellement partie du contenu du document. Toutes les classes générées sont référencées aux nœuds DOM qu'ils représentent, et non aux valeurs. Ceci signifie qu'assigner une instance de la classe générée ne copie pas la valeur, ceci crée une référence supplémentaire aux mêmes données.

## Prise en charge du schéma XML

Les constructions de schéma XML suivantes sont traduites en code :

a) espaces de noms XML

b) Types simples :

- Types de schéma XML intégrés
- Les types simples dérivés par extension
- Les types simples dérivés par restriction
- Facettes
- Enumérations
- Motifs

c) Types complexes :

- Nœud anyType intégré
- Types complexes définis par l'utilisateur
- Dérivé par extension : Mappé vers les classes dérivées
- Dérivé par restriction
- Contenu complexe
- Contenu simple
- Contenu mixte

Les fonctions de schéma XML avancées suivantes ne son pas prises en charge (ou pas entièrement prises en charge) dans les classes wrapper générées :

- Caractères génériques : `xs:any` and `xs:anyAttribute`
- Modèles de contenu (séquence, choix, tout). Le compositeur du niveau supérieur est disponible dans SPL 1043, mais n'est pas appliqué par les classes générées.
- Valeurs par défaut et fixes pour les attributs. Celles-ci sont disponibles dans SPL 1043, mais ne sont pas définies ou appliquées par les classes générées.
- Attributs `xsi:type`, types abstraits. Si vous devez écrire l'attribut xsi:type, utilisez la méthode `SetXsiType()` de classes générées.
- Types d'union : pas toutes les combinaisons sont prises en charge.
- Les groupes de substitution sont partiellement pris en charge (resolus comme « choice »).

- Attribut `nillable="true"` et `xsi:nil`
- Contraintes uniques
- Contraintes d'identité (`key` and `keyref`)

## 15.2.3.1 À propos des Schema Wrapper Libraries (C++)

### Types de caractère

Le code C++ généré peut être compilé avec ou sans prise en charge Unicode. Selon ce paramètre, les types `string_type` et `tstring` seront tous les deux définis en tant que `std::string` ou `std::wstring`, consistant en des caractères larges ou étroits. Pour utiliser les caractères Unicode dans votre fichier XML qui ne sont pas représentables avec l'ensemble de caractères 8-bit actuels, la prise en charge Unicode doit être activée. Prêtez une attention particulière aux macros `_T()` . Ce macro garantit que les constantes de string sont stockées correctement, que vous compiliez pour les programmes Unicode ou non-Unicode.

### Types de données

Le mappage par défaut des types Schéma XML dans les types de données C++ est :

| Schéma XML | C++ | Remarques |
|---|---|---|
| `xs:string` | `string_type` | string_type est défini en tant que std::string ou std:wstring |
| `xs:boolean` | `bool` | |
| `xs:decimal` | `double` | C++ n'a pas de type décimal, donc double est utilisé. |
| `xs:float, xs:double` | `double` | |
| `xs:integer` | `__int64` | xs:integer a une plage illimitée, mappée sur __int64 pour des raisons d'efficacité. |
| `xs:nonNegativeInteger` | `unsigned __int64` | voir ci-dessus |
| `xs:int` | `int` | |
| `xs:unsignedInt` | `unsigned int` | |
| `xs:dateTime, date, time, gYearMonth, gYear, gMonthDay, gDay, gMonth` | `altova::DateTime` 998 | |
| `xs:duration` | `altova::Duration` 1001 | |
| `xs:hexBinary and xs:base64Binary` | `std::vector<unsigned char>` | L'encodage et le décodage des données binaires est effectué automatiquement. |
| `xs:anySimpleType` | `string_type` | |

Tous les types de Schéma XML qui ne sont pas contenus dans la liste sont des types dérivés, et mappés sur le même C++ que leur type de base respectif.

## Classes générées

Pour chaque type dans le schéma, une classe est générée qui contient un membre pour chaque attribut et élément du type. Les membres sont nommés comme les attributs ou éléments dans le schéma original (en cas d'une collision éventuelle, un nombre est ajouté). Pour les types simples, des opérateurs d'attribution et de conversion sont générés. Pour des types simples avec des facettes d'énumération, les méthodes `GetEnumerationValue()` et `SetEnumerationValue(int)` peuvent être utilisés ensemble avec des constantes générées pour chaque valeur d'énumération. De plus, la méthode `StaticInfo()` permet l'accès à l'information de schéma en tant qu'un des types suivants :

`altova::meta::SimpleType`[1006]
`altova::meta::ComplexType`[1005]

Les classes générées depuis des types complexes comprennent la méthode `SetXsiType()`, qui vous permet de définir l'attribut `xsi:type` du type. Cette méthode est utile si vous souhaitez créer des éléments d'instance XML d'un type dérivé.

Outre les classes pour les types déclarés dans le schéma XML, une classe de document (identifiée par "CDoc", voir ci-dessous) est générée. Elle contient tous les éléments root possibles, en tant que membres, ainsi que plusieurs autres méthodes. Pour plus d'informations concernant la classe, voir `[YourSchema]::[CDoc]`[1008] .

**Note:** Le véritable nom de la classe dépend du nom du schéma .xsd.

Pour chaque attribut ou élément membre d'un type de schéma, une nouvelle classe est générée. Pour plus d'informations concernant ces classes, voir :

`[YourSchema]::MemberAttribute`[1011]
`[YourSchema]::MemberElement`[1012]

**Note:** Les véritables noms de la classe dépendent du nom de l'attribut ou de l'élément membre du schéma.

Voir aussi Exemple : Utiliser les bibliothèques de Schema Wrapper[966].

## Gestion d'erreur

Les erreurs sont rapportées par exceptions. Les classes d'exception suivantes sont définies dans l'espace de noms `altova`:

| Classe | Classe de Base | Description |
|---|---|---|
| `Erreur` | `std::logic_error` | Erreur de logique de programme interne (indépendant des données d'entrée). |
| `Exception` | `std::runtime_error` | Classe de Base pour les erreurs de runtime. |

| | | |
|---|---|---|
| `InvalidArgumentsException` | `Exception` | Une méthode a été appelée avec des valeurs d'argument invalide. |
| `ConversionException` | `Exception` | Exception lancée lorsqu'une conversion de type échoue. |
| `StringParseException` | `ConversionException` | Une valeur dans l'espace lexical ne peut pas être converti dans un espace de valeur. |
| `ValueNotRepresentableException` | `ConversionException` | Une valeur dans l'espace de valeur ne peut pas être converti dans un espace lexical. |
| `OutOfRangeException` | `ConversionException` | Une valeur de source ne peut pas être représentée dans un domaine cible. |
| `InvalidOperationException` | `Exception` | Une opération a été tentée qui n'est pas valide dans le contexte donné. |
| `DataSourceUnavailableException` | `Exception` | Un problème est survenu pendant le chargement d'une instance XML. |
| `DataTargetUnavailableException` | `Exception` | Un problème est survenu pendant l'enregistrement d'une instance XML. |

Toutes les classes d'exception contiennent un texte de message et un pointeur dans une exception intérieure possible.

| Méthode | Objectif |
|---|---|
| `string_type message()` | Retourne une description textuelle de l'exception. |
| `std::exception inner()` | Retourne l'exception qui a causé cette exception, si disponible, ou NULL. |

## Accéder à l'information de schéma

La bibliothèque générée permet d'accéder à des informations de schéma statique par le biais des classes suivantes. `Toutes les méthodes sont déclarées en tant que` const. Les méthodes qui retournent une de ces classes des métadonnées retournent un objet NULL si la propriété respective n'existe pas.

`altova::meta::Attribute` 1004
`altova::meta::ComplexType` 1005
`altova::meta::Element` 1006
`altova::meta::SimpleType` 1006

### 15.2.3.2 À propos des Schema Wrapper Libraries (C#)

Le mappage par défaut des types Schéma XML dans les types de données C# est le suivant.

| Schéma XML | C# | Remarques |
|---|---|---|
| xs:string | string | |
| xs:boolean | bool | |
| xs:decimal | decimal | xs:decimal dispose une plage et une précision illimitée, mappé vers les décimales pour des raisons d'efficacité. |
| xs:float, xs:double | double | |
| xs:long | long | |
| xs:unsignedLong | ulong | |
| xs:int | int | |
| xs:unsignedInt | uint | |
| xs:dateTime, date, time, gYearMonth, gYear, gMonthDay, gDay, gMonth | Altova.Types.DateTime 1013 | |
| xs:duration | Altova.Types.Duration 1017 | |
| xs:hexBinary and xs:base64Binary | byte[] | L'encodage et le décodage des données binaires est effectué automatiquement. |
| xs:anySimpleType | string | |

Tous les types de Schéma XML qui ne sont pas contenus dans la liste sont des types dérivés, et mappés sur le même C# que leur type de base respectif.

#### Classes générées

Pour chaque type dans le schéma, une classe est générée qui contient un membre pour chaque attribut et élément du type. Les membres sont nommés comme les attributs ou éléments dans le schéma original (en cas d'une collision éventuelle, un nombre est ajouté). Pour les types simples, des opérateurs d'attribution et de conversion sont générés. Pour des types simples avec des facettes d'énumération, les méthodes `GetEnumerationValue()` et `SetEnumerationValue(int)` peuvent être utilisés ensemble avec des constantes générées pour chaque valeur d'énumération. De plus, la méthode `StaticInfo()` permet l'accès à l'information de schéma en tant qu'un des types suivants :

Altova.Xml.Meta.SimpleType 1022
Altova.Xml.Meta.ComplexType 1021

Les classes générées depuis des types complexes comprennent la méthode `SetXsiType()`, qui vous permet de définir l'attribut `xsi:type` du type. Cette méthode est utile si vous souhaitez créer des éléments d'instance XML d'un type dérivé.

Outre les classes pour les types déclarés dans le schéma XML, une classe de document (identifiée par "Doc", voir ci-dessous) est générée. Elle contient tous les éléments root possibles, en tant que membres, ainsi que plusieurs autres méthodes. Pour plus d'informations concernant la classe, voir [YourSchema].[Doc] 1023.

**Note:** Le véritable nom de la classe dépend du nom du schéma .xsd.

Pour chaque attribut ou élément membre d'un type de schéma, une nouvelle classe est générée. Pour plus d'informations concernant ces classes, voir :

[YourSchemaType].MemberAttribute 1026
[YourSchemaType].MemberElement 1027

**Note:** Les véritables noms de la classe dépendent du nom de l'attribut ou de l'élément membre du schéma.

## Gestion d'erreur

Les erreurs sont rapportées par exceptions. Les classes d'exception suivantes sont définies dans l'espace de noms Altova :

| Classe | Classe de Base | Description |
|---|---|---|
| `ConversionException` | `Exception` | Exception lancée lorsqu'une conversion de type échoue |
| `StringParseException` | `ConversionException` | Une valeur dans l'espace lexical ne peut pas être converti dans un espace de valeur. |
| `DataSourceUnavailableException` | `System.Exception` | Un problème est survenu pendant le chargement d'une instance XML. |
| `DataTargetUnavailableException` | `System.Exception` | Un problème est survenu pendant l'enregistrement d'une instance XML. |

NET suivantes sont utilisées communément :

| Classe | Description |
|---|---|
| `System.Exception` | Classe de Base pour les erreurs de runtime |
| `System.ArgumentException` | Une méthode a été appelée avec des valeurs d'argument invalide, ou bien une conversion de type a échoué. |
| `System.FormatException` | Une valeur dans l'espace lexical ne peut pas être converti dans un espace de valeur. |
| `System.InvalidCastException` | Une valeur ne peut pas être converie dans un autre type. |

| Classe | Description |
|---|---|
| System.OverflowException | Une valeur de source ne peut pas être représentée dans un domaine cible. |

### Accéder à l'information de schéma

La bibliothèque générée permet d'accéder à des informations de schéma statique par le biais des classes suivantes :

Altova.Xml.Meta.Attribute 1020
Altova.Xml.Meta.ComplexType 1021
Altova.Xml.Meta.Element 1021
Altova.Xml.Meta.SimpleType 1022

Les propriétés qui retournent une de ces classes des métadonnées retournent null si la propriété respective n'existe pas.

## 15.2.3.3 À propos des Schema Wrapper Libraries (Java)

Le mappage par défaut des types Schéma XML dans les types de données Java est le suivant :

| Schéma XML | Java | Remarques |
|---|---|---|
| xs:string | String | |
| xs:boolean | booléenne | |
| xs:decimal | java.math.BigDecimal | |
| xs:float, xs:double | double | |
| xs:integer | java.math.BigInteger | |
| xs:long | long | |
| xs:unsignedLong | java.math.BigInteger | Java ne dispose pas de types non signés. |
| xs:int | int | |
| xs:unsignedInt | long | Java ne dispose pas de types non signés. |
| xs:dateTime, date, time, gYearMonth, gYear, gMonthDay, gDay, gMonth | com.altova.types.DateTime 1028 | |
| xs:duration | com.altova.types.Duration 1032 | |
| xs:hexBinary and xs:base64Binary | byte[] | L'encodage et le décodage des données binaires est effectué |

| Schéma XML | Java | Remarques |
|---|---|---|
| | | automatiquement. |
| xs:anySimpleType | string | |

Tous les types de Schéma XML qui ne sont pas contenus dans la liste sont des types dérivés, et mappés sur le même type Java que leur type de base respectif.

## Classes générées

Pour chaque type dans le schéma, une classe est générée qui contient un membre pour chaque attribut et élément du type. Les membres sont nommés comme les attributs ou éléments dans le schéma original (en cas d'une collision éventuelle, un nombre est ajouté). Pour les types simples, des opérateurs d'attribution et de conversion sont générés. Pour des types simples avec des facettes d'énumération, les méthodes `GetEnumerationValue()` et `SetEnumerationValue(int)` peuvent être utilisés ensemble avec des constantes générées pour chaque valeur d'énumération. De plus, la méthode `StaticInfo()` permet l'accès à l'information de schéma en tant qu'un des types suivants :

`com.altova.xml.meta.SimpleType` 1038
`com.altova.xml.meta.ComplexType` 1036

Les classes générées depuis des types complexes comprennent la méthode `SetXsiType()`, qui vous permet de définir l'attribut `xsi:type` du type. Cette méthode est utile si vous souhaitez créer des éléments d'instance XML d'un type dérivé.

Outre les classes pour les types déclarés dans le schéma XML, une classe de document (identifiée par "Doc", voir ci-dessous) est générée. Elle contient tous les éléments root possibles, en tant que membres, ainsi que plusieurs autres méthodes. Pour plus d'informations concernant la classe, voir `com.[YourSchema].[Doc]` 1038.

**Note:** Le véritable nom de la classe dépend du nom du schéma .xsd.

Pour chaque attribut ou élément membre d'un type de schéma, une nouvelle classe est générée. Pour plus d'informations concernant ces classes, voir :

`com.[YourSchema].[YourSchemaType].MemberAttribute` 1041
`com.[YourSchema].[YourSchemaType].MemberElement` 1042

**Note:** Les véritables noms de la classe dépendent du nom de l'attribut ou de l'élément membre du schéma.

## Gestion d'erreur

Les erreurs sont rapportées par exceptions. Les classes d'exception suivantes sont définies dans l'espace de noms com.altova:

| Classe | Classe de Base | Description |
|---|---|---|
| SourceInstanceUnvailableException | Exception | Un problème est survenu pendant le chargement d'une instance XML. |

| Classe | Classe de Base | Description |
|---|---|---|
| `TargetInstanceUnavailableException` | Exception | Un problème est survenu pendant l'enregistrement d'une instance XML. |

De plus, les exceptions Java suivantes sont utilisées communément :

| Classe | Description |
|---|---|
| `java.lang.Error` | Erreur de logique de programme interne (indépendant des données d'entrée) |
| `java.lang.Exception` | Classe de Base pour les erreurs de runtime |
| `java.lang.IllegalArgumentsException` | Une méthode a été appelée avec des valeurs d'argument invalide, ou bien une conversion de type a échoué. |
| `java.lang.ArithmeticException` | Exception lancée lorsqu'une conversion de type numérique échoue. |

### Accéder à l'information de schéma

La bibliothèque générée permet d'accéder à des informations de schéma statique par le biais des classes suivantes :

`com.altova.xml.meta.Attribute` 1036
`com.altova.xml.meta.ComplexType` 1036
`com.altova.xml.meta.Element` 1037
`com.altova.xml.meta.SimpleType` 1038

Les propriétés qui retournent une de ces classes des métadonnées retournent null si la propriété respective n'existe pas.

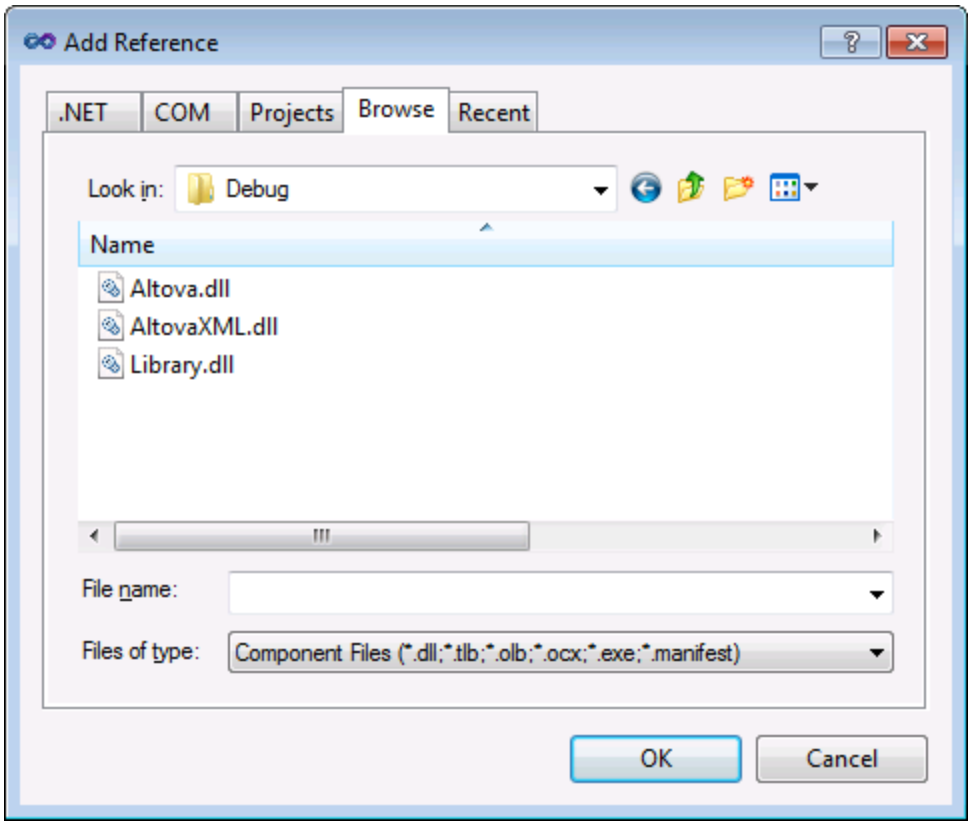
## 15.2.3.4 Intégrer des Bibliothèques Schema Wrapper

Pour utiliser les bibliothèques Altova dans votre projet personnalisé, veuillez vous référer aux bibliothèques depuis votre projet ou les inclure dans votre projet, comme indiqué ci-dessous pour chaque langage.

### C#

Pour intégrer les bibliothèques Altova dans un projet C# existant :

1. Une fois que MapForce génère du code depuis un schéma (par exemple, **YourSchema.xsd**), construire la solution générée **YourSchema.sln** dans Visual Studio. Cette solution se trouve dans un dossier de projet avec le même nom en tant que le schéma.
2. Cliquer avec la touche de droite dans le projet existant dans Visual Studio, et sélectionner **Ajouter Référence**.
3. Dans l'onglet Chercher, naviguer dans les bibliothèques suivantes : **Altova.dll**, **AltovaXML.dll** et **YourSchema.dll** situés dans le répertoire de sortie des projets générés (par exemple, **bin\Debug**).



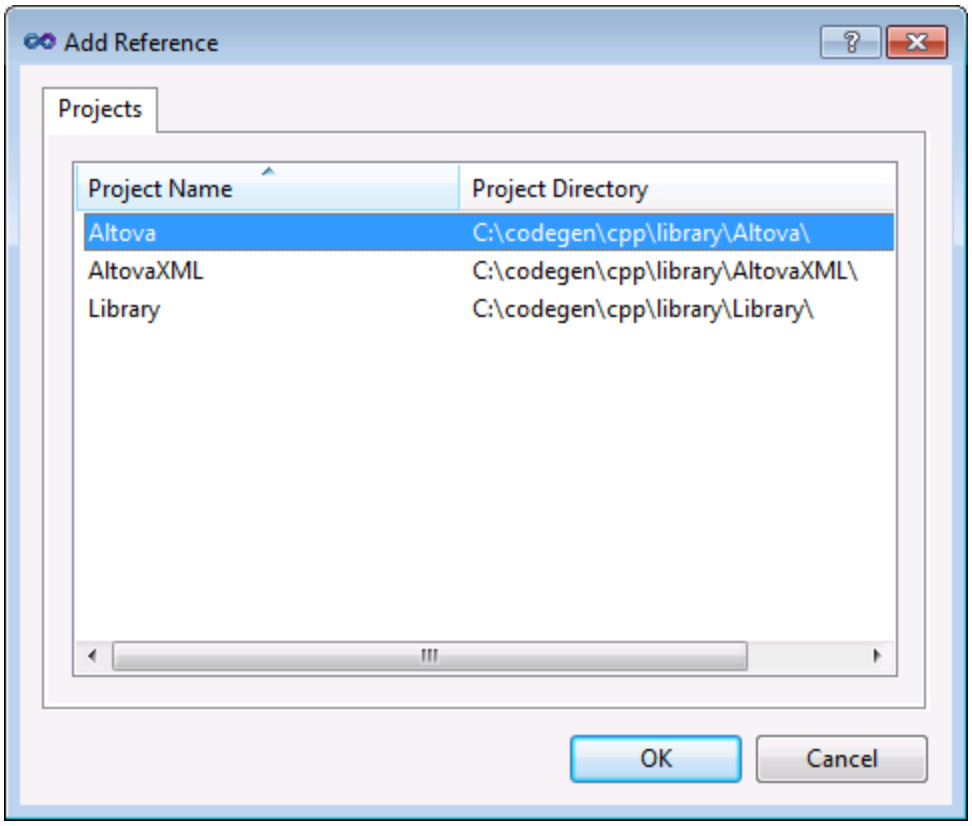

## C++

Le moyen le plus simple d'intégrer les bibliothèques dans un projet C++ existant est d'ajouter les fichiers de projet généré dans votre solution. Par exemple, assumons que vous avez généré du code depuis une schéma appelé **Library.xsd** et sélectionné **c:\codegen\cpp\library** en tant que répertoire cible. Les bibliothèques générés dans ce cas sont disponibles sous :

- c:\codegen\cpp\library\Altova.vcxproj
- c:\codegen\cpp\library\AltovaXML\AltovaXML.vcxproj
- c:\codegen\cpp\library\Library.vcxproj

Tout d'abord, ouvrir la solution **c:\codegen\cpp\library\Library.sln** générée et l'intégrer dans Visual Studio.

Ensuite ouvrir votre solution Visual Studio existante (dans Visual Studio 2010, dans cet exemple), cliquer avec la touche de droite, choisir **Add | Existing Project**, et ajouter les fichiers de projet recensés ci-dessus, un par un. Soyez patient tant que Visual Studio parse les fichiers. Ensuite cliquer avec la touche de droite sur votre projet et choisir **Propriétés**. Dans le dialogue Pages de propriété, sélectionner **Common Properties | Framework and References**, puis cliquer sur **Add New Reference**. Ensuite, sélectionner et ajouter chacun des projets suivants : *Altova*, *AltovaXML* et *Library*.




Voir aussi la documentation MSDN pour utiliser les fonctions depuis une bibliothèque personnalisée, comme applicables à votre version de Visual Studio, par exemple :

- Si vous avez choisi de générer des bibliothèques statiques, voir https://msdn.microsoft.com/en-us/library/ms235627(v=vs.100).aspx
- Si vous avez choisi de générer des bibliothèques dynamique, voir https://msdn.microsoft.com/en-us/library/ms235636(v=vs.100).aspx

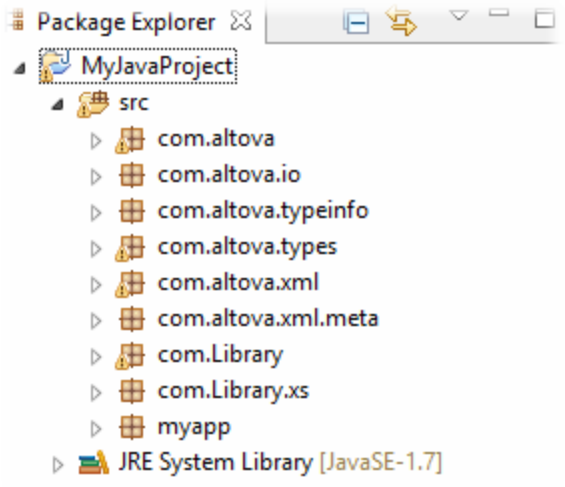
L'option de générer des bibliothèques statiques ou dynamiques est disponible dans les options de la génération (voir Génération 1089).

## Java

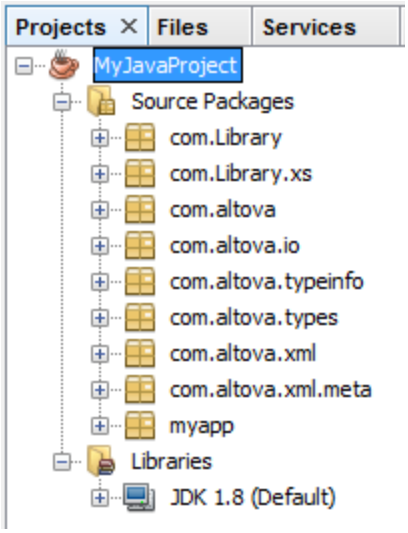
Un des moyens d'intégrer les packages Altova dans vos projets Java est de copier le répertoire **com** du code généré vers le répertoire qui stocke les packages de source de votre projet Java (par exemple, **C:\Workspace\MyJavaProject\src**). Par exemple, nous partons du principe de vous avez généré du code dans **c:\codegen\java\library**. Les classes Altova générées dans ce cas sont disponibles sous **c:\codegen\java\library\com**.

Une fois avoir copié les bibliothèques, actualiser le projet. Pour actualiser le projet dans Eclipse, le sélectionner dans le Package Explorer, et appuyer sur **F5**. Pour actualiser le projet dans NetBeans IDE 8.0, sélectionner la commande de menu **Source | Scan for External Changes**.

Une fois que vous avez effectué l'opération copy, les packages Altova sont disponibles dans le Package Explorer (dans le cas d'Eclipse), ou sous "Source Packages" dans le volet Projets (dans le cas de NetBeans IDE).




*Packages Altova dans Eclipse 4.4*




*Packages Altova dans NetBeans IDE 8.0.2*

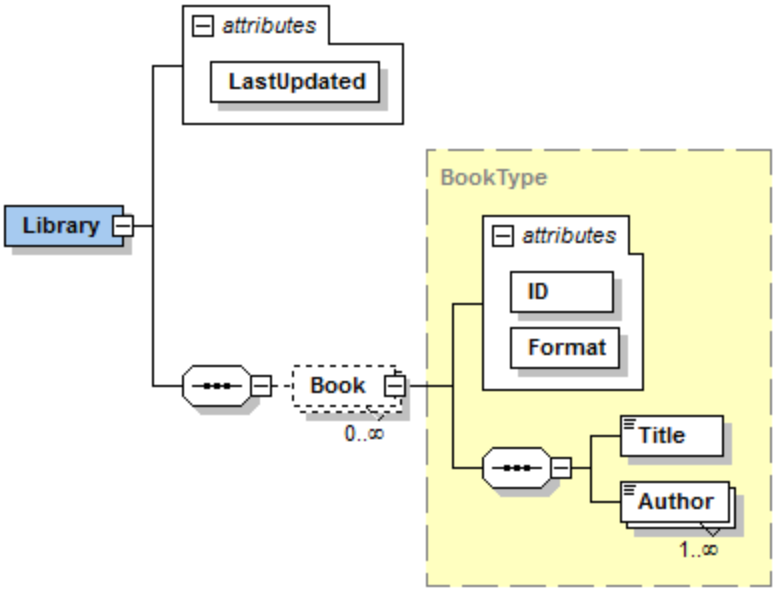
## 15.2.3.5 Exemple: Bibliothèques de livres

Cet exemple illustre comment utiliser les bibliothèques de schema wrapper générées afin d'écrire ou de lire des documents XML conforme en terme de programmation au schéma. Avant d'utiliser le code d'échantillon, veuillez prendre le temps de comprendre la structure du schéma d'exemple inclus.

Le schéma utilisé dans cet exemple décrit une bibliothèque de livres. La définition complète du schéma est affichée ci-dessous. Enregistrer cet extrait de code en tant que **Library.xsd** si vous souhaitez obtenir les mêmes résultats que dans cet exemple. Vous aurez besoin de ce schéma pour générer les bibliothèques de code utilisées dans cet exemple.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns="http://www.nanonull.com/LibrarySample"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.nanonull.com/LibrarySample" elementFormDefault="qualified"
attributeFormDefault="unqualified">
   <xs:element name="Library">
      <xs:complexType>
         <xs:sequence>
            <xs:element name="Book" type="BookType" minOccurs="0" maxOccurs="unbounded"/>
         </xs:sequence>
         <xs:attribute name="LastUpdated" type="xs:dateTime"/>
      </xs:complexType>
   </xs:element>
   <xs:complexType name="BookType">
      <xs:sequence>
         <xs:element name="Title" type="xs:string"/>
         <xs:element name="Author" type="xs:string" maxOccurs="unbounded"/>
      </xs:sequence>
      <xs:attribute name="ID" type="xs:integer" use="required"/>
      <xs:attribute name="Format" type="BookFormatType" use="required"/>
   </xs:complexType>
   <xs:complexType name="DictionaryType">
      <xs:complexContent>
         <xs:extension base="BookType">
            <xs:sequence>
               <xs:element name="FromLang" type="xs:string"/>
               <xs:element name="ToLang" type="xs:string"/>
            </xs:sequence>
         </xs:extension>
      </xs:complexContent>
   </xs:complexType>
   <xs:simpleType name="BookFormatType">
      <xs:restriction base="xs:string">
         <xs:enumeration value="Hardcover"/>
         <xs:enumeration value="Paperback"/>
         <xs:enumeration value="Audiobook"/>
         <xs:enumeration value="E-book"/>
      </xs:restriction>
   </xs:simpleType>
</xs:schema>
```

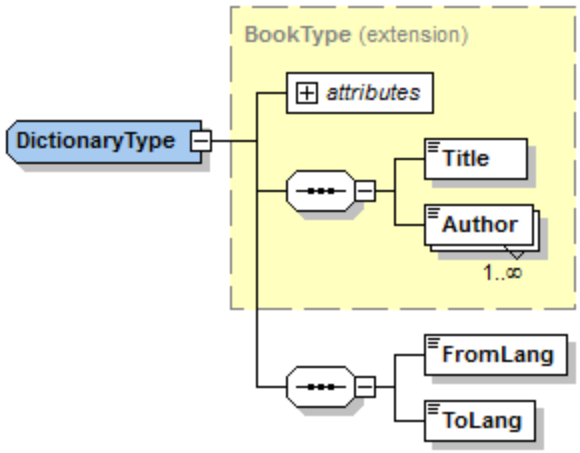
**Library** est un élément racide d'un `complexType` qui peut être représenté graphiquement comme suit dans le mode Schéma de XMLSpy:




Comme indiqué ci-dessus, la bibliothèque contient un attribut **LastUpdated** (défini en tant que `xs:dateTime`), et stocke une séquence de livres. Chaque livre est un `xs:complexType` et possède deux attributs : un **ID** (défini en tant que `xs:integer`), et un **Format**. Le format de chaque livre peut être hardcover, paperback, audiobook, ou e-book. Dans le schéma, **Format** est défini en tant que `xs:simpleType` qui utilise une énumération des valeurs mentionnées ci-dessus.

Chaque livre présente aussi un élément **Title** (défini en tant que `xs:string`), et un ou plusieurs **Author** éléments (définis en tant que `xs:string`).

La bibliothèque peut aussi contenir des livres qui sont des dictionnaires. Dictionaries ont le type `DictionaryType`, qui est dérivé par extension depuis `BookType`. En d'autres termes, un dictionnaire hérite de tous les attributs et éléments d'un Book (livre), plus deux éléments supplémentaires : **FromLang** et **ToLang**, comme illustré ci-dessous.



Les éléments **FromLang** et **ToLang** stockent le langage de source et de destination du dictionnaire.

Un fichier d'instance XML valide conformément au schéma ci-dessus peut donc ressembler à l'exemple affiché dans l'exemple ci-dessous (s'il se trouve dans le même répertoire que le fichier de schéma) :

```
<?xml version="1.0" encoding="utf-8"?>
<Library xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://www.nanonull.com/LibrarySample"
xsi:schemaLocation="http://www.nanonull.com/LibrarySample Library.xsd" LastUpdated="2016-
02-03T17:10:08.4977404">
 <Book ID="1" Format="E-book">
 <Title>The XMLSpy Handbook</Title>
 <Author>Altova</Author>
 </Book>
 <Book ID="2" Format="Paperback" xmlns:n1="http://www.nanonull.com/LibrarySample"
xsi:type="n1:DictionaryType">
 <Title>English-German Dictionary</Title>
 <Author>John Doe</Author>
 <FromLang>English</FromLang>
 <ToLang>German</ToLang>
 </Book>
</Library>
```

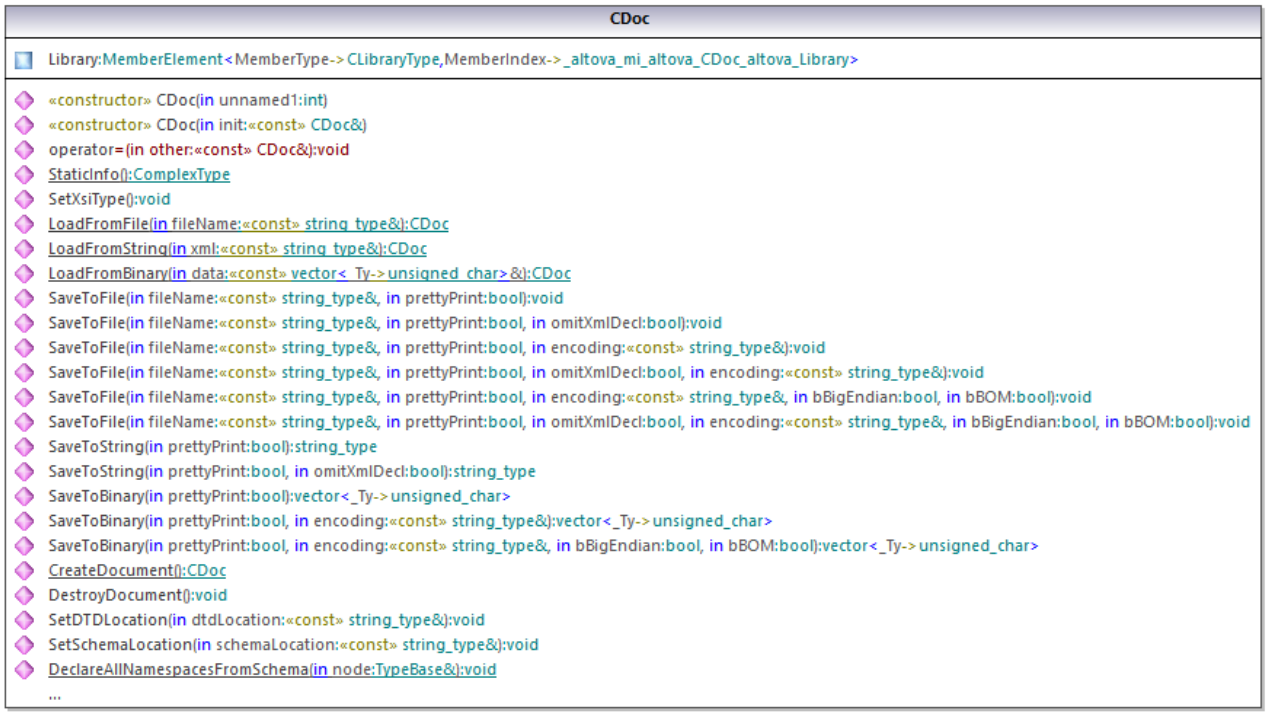
La rubrique suivante illustre comment lire ou écrire dans un tel fichier par le biais d'un programme. Pour commencer, générer le code de schema wrapper depuis le schéma ci-dessus, en utilisant les étapes décrites dans Générer du code depuis des Schémas XML ou des DTD 953.

#### 15.2.3.5.1 Lire et écrire des documents XML (C++)

Après avoir généré du code depuis le schéma d'exemple 966, une application C++ test est créée, avec plusieurs bibliothèques Altova de support.

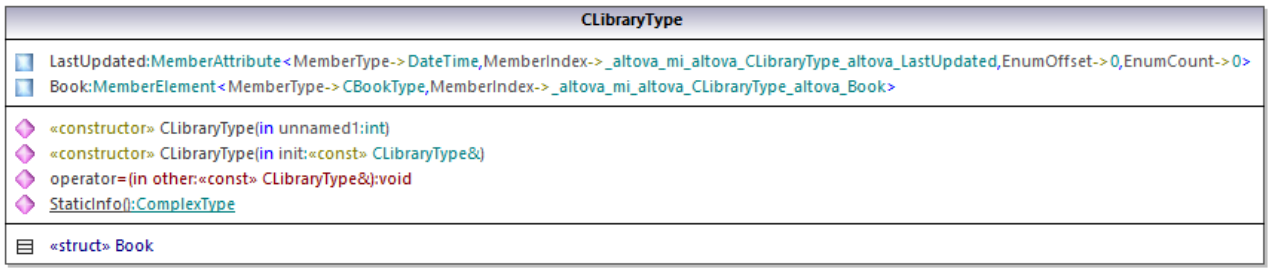
### À propos des bibliothèques C++ générées

La classe centrale du code généré est la classe `CDoc` qui représente le document XML. Une telle classe est générée pour tous les schémas et son nom dépend du nom de fichier de schéma. Comme indiqué dans le diagramme, cette classe fournit des méthodes pour charger les documents depuis des fichiers, des streams binaires ou des strings (ou pour enregistrer des documents dans des fichiers, des streams, des strings). Pour consulter une liste de tous les membres exposés par cette classe, voir la référence de classe ( `[YourSchema]::[CDoc]` 1008 ).

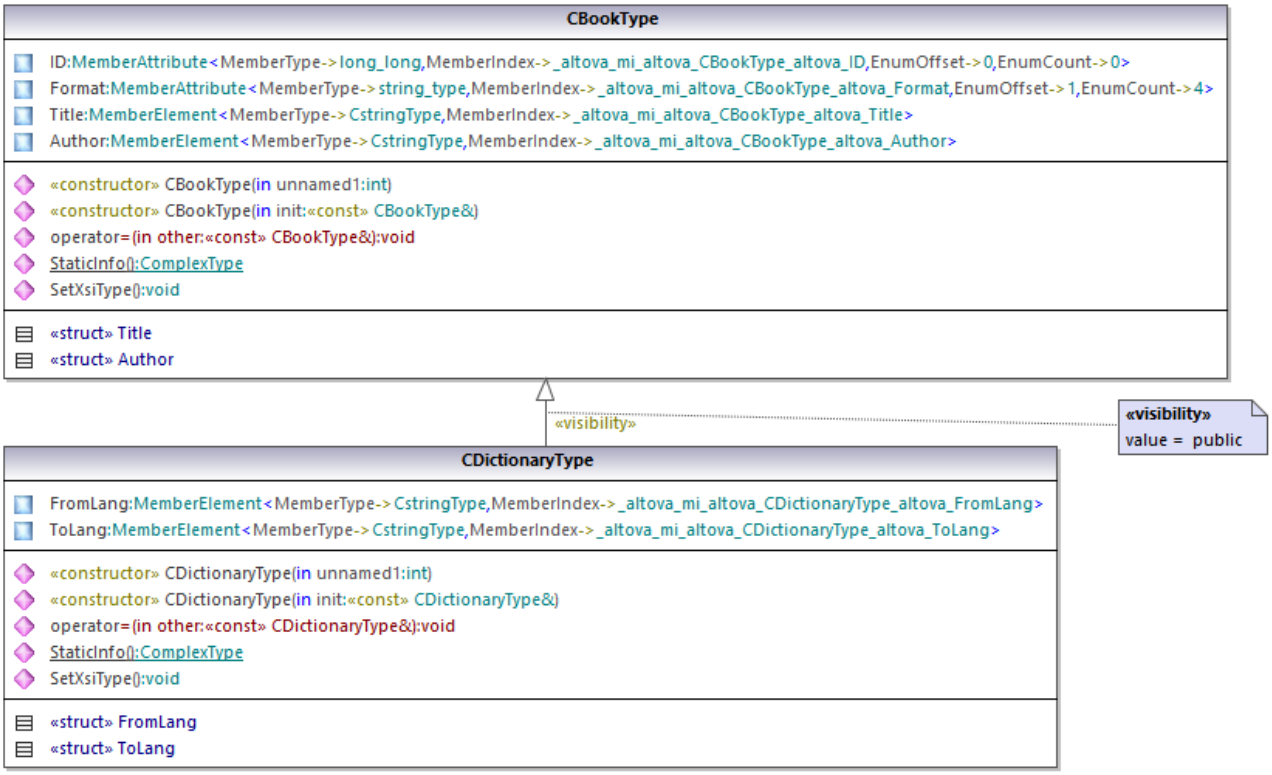



Le champ `Library` de la classe `CDoc` représente la racine réelle du document. **Library** est un élément dans le fichier XML, donc dans le code C++ il a une classe de modèle en tant que type (`MemberElement`). Le modèle de classe expose des méthodes et des propriétés pour interagir avec l'élément **Library** . En général, chaque attribut et chaque élément d'un type dans le schéma est saisi dans le code généré avec les classes de modèle `MemberAttribute` et `MemberElement`, respectivement. Pour plus d'informations, voir la référence de classe `[YourSchema]::MemberAttribute`[1011] et `[YourSchema]::MemberElement`[1012] .

La classe `CLibraryType` est générée depuis le type complexe **LibraryType** dans le schéma. Veuillez noter que la classe `CLibraryType` contient deux champs : `Book` et `LastUpdated`. Conformément à la logique déjà mentionnée ci-dessus, ces champs correspondent à l'élément **Book** et à l'attribut **LastUpdated** dans le schéma, et vous permettent de manipuler par le biais d'un programme (apposer, supprimer, etc.) des éléments et des attributs dans le document d'instance XML.




Puisque le `DictionaryType` est un type complexe dérivé depuis **BookType** dans le schéma, cette relation se retrouve aussi dans les classes générées. Comme illustré dans le diagramme, la classe `CDictionaryType` hérite la classe `CBookType`.




Si votre schéma XML définit des types simples en tant qu'énumérations, les valeurs énumérées deviennent disponibles en tant que valeurs `enum` dans le code généré. Dans le schéma utilisé dans ces exemple, un format de livre peut être hardcover, paperback, e-book, etc. Dans le code généré, ces valeurs peuvent donc être disponibles via un `enum` membre de la classe `CBookFormatType`.

## Écrire un document XML

1. Ouvrir la solution **LibraryTest.sln** dans Visual Studio générée depuis le schéma Library mentionné plus tôt dans cet exemple.

Tout en prototypant une application depuis un schéma XML changeant fréquemment, vous devrez éventuellement générer fréquemment du code dans le même répertoire, de manière à ce que les changements de schéma sont réfléchis immédiatement dans le code. Veuillez noter que l'application de test généré et les bibliothèques Altova sont écrasées à chaque fois que vous générez du code dans le même répertoire cible. C'est pourquoi il ne faut pas ajouter du code à l'application de test généré. Au lieu de cela, veuillez intégrer les bibliothèques dans votre projet (voir Intégrer des Bibliothèques Schema Wrapper 963).

2. Dans Solution Explorer, ouvrir le fichier **LibraryTest.cpp**, et éditer la méthode `Example()` comme indiqué ci-dessous.

```
#include <ctime> // required to get current time
using namespace Doc; // required to work with Altova libraries
```

```
void Example()
{
   // Create a new, empty XML document
   CDoc libDoc = CDoc::CreateDocument();

   // Create the root element <Library> and add it to the document
   CLibraryType lib = libDoc.Library.append();

   // Get current time and set the "LastUpdated" attribute using Altova classes
   time_t t = time(NULL);
   struct tm * now = localtime( & t );
   altova::DateTime dt = altova::DateTime(now->tm_year + 1900, now->tm_mon + 1, now-
>tm_mday, now->tm_hour, now->tm_min, now->tm_sec);
   lib.LastUpdated = dt;

   // Create a new <Book> and add it to the library
   CBookType book = lib.Book.append();

   // Set the "ID" attribute of the book
   book.ID = 1;

   // Set the "Format" attribute of the <Book> using an enumeration constant
   book.Format.SetEnumerationValue( CBookFormatType::k_Paperback );

   // Add the <Title> and <Author> elements, and set values
   book.Title.append() = _T("The XML Spy Handbook");
   book.Author.append() = _T("Altova");

   // Append a dictionary (book of derived type) and populate its attributes and elements
   CDictionaryType dictionary = CDictionaryType(lib.Book.append().GetNode());
   dictionary.ID = 2;
   dictionary.Format.SetEnumerationValue( CBookFormatType::k_E_book);
   dictionary.Title.append() = _T("English-German Dictionary");
   dictionary.Author.append() = _T("John Doe");
   dictionary.FromLang.append() = _T("English");
   dictionary.ToLang.append() = _T("German");

   // Since dictionary a derived type, set the xsi:type attribute of the book element
   dictionary.SetXsiType();

   // Optionally, set the schema location
   libDoc.SetSchemaLocation(_T("Library.xsd"));

   // Save the XML document to a file with default encoding (UTF-8),
   // "true" causes the file to be pretty-printed.
   libDoc.SaveToFile(_T("GeneratedLibrary.xml"), true);

   // Destroy the document
   libDoc.DestroyDocument();
}
```

3. Appuyer sur **F5** pour lancer le débogage. Si le code a été exécuté avec succès, un fichier **GeneratedLibrary.xml** est créé dans le répertoire de sortie de solution.

## Lire un document XML

1. Ouvrir la solution **LibraryTest.sln** dans Visual Studio.
2. Enregistrer le code ci-dessous en tant que **Library1.xml** dans un répertoire qui peut être lu par le code de programme (par exemple, le même répertoire que **LibraryTest.sln**).

```
<?xml version="1.0" encoding="utf-8"?>
<Library xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://www.nanonull.com/LibrarySample"
xsi:schemaLocation="http://www.nanonull.com/LibrarySample Library.xsd" LastUpdated="2016-
02-03T17:10:08.4977404">
 <Book ID="1" Format="E-book">
 <Title>The XMLSpy Handbook</Title>
 <Author>Altova</Author>
 </Book>
 <Book ID="2" Format="Paperback" xmlns:n1="http://www.nanonull.com/LibrarySample"
xsi:type="n1:DictionaryType">
 <Title>English-German Dictionary</Title>
 <Author>John Doe</Author>
 <FromLang>English</FromLang>
 <ToLang>German</ToLang>
 </Book>
</Library>
```

3. Dans Solution Explorer, ouvrir le fichier **LibraryTest.cpp**, et éditer la méthode `Example()` comme indiqué ci-dessous.

```
using namespace Doc;
void Example()
{
   // Load XML document
   CDoc libDoc = CDoc::LoadFromFile(_T("Library1.xml"));

   // Get the first (and only) root element <Library>
   CLibraryType lib = libDoc.Library.first();

   // Check whether an element exists:
   if (!lib.Book.exists())
   {
      tcout << "This library is empty." << std::endl;
      return;
   }

   // iteration: for each <Book>...
   for (Iterator<CBookType> itBook = lib.Book.all(); itBook; ++itBook)
   {
      // output values of ISBN attribute and (first and only) title element
      tcout << "ID: " << itBook->ID << std::endl;
      tcout << "Title: " << tstring(itBook->Title.first()) << std::endl;

      // read and compare an enumeration value
```

```
      if (itBook->Format.GetEnumerationValue() == CBookFormatType::k_Paperback)
         tcout << "This is a paperback book." << std::endl;

      // for each <Author>...
      for (CBookType::Author::iterator itAuthor = itBook->Author.all(); itAuthor; +
+itAuthor)
         tcout << "Author: " << tstring(itAuthor) << std::endl;

      // alternative: use count and index
      for (unsigned int j = 0; j < itBook->Author.count(); ++j)
         tcout << "Author: " << tstring(itBook->Author[j]) << std::endl;
   }

   // Destroy the document
   libDoc.DestroyDocument();
}
```

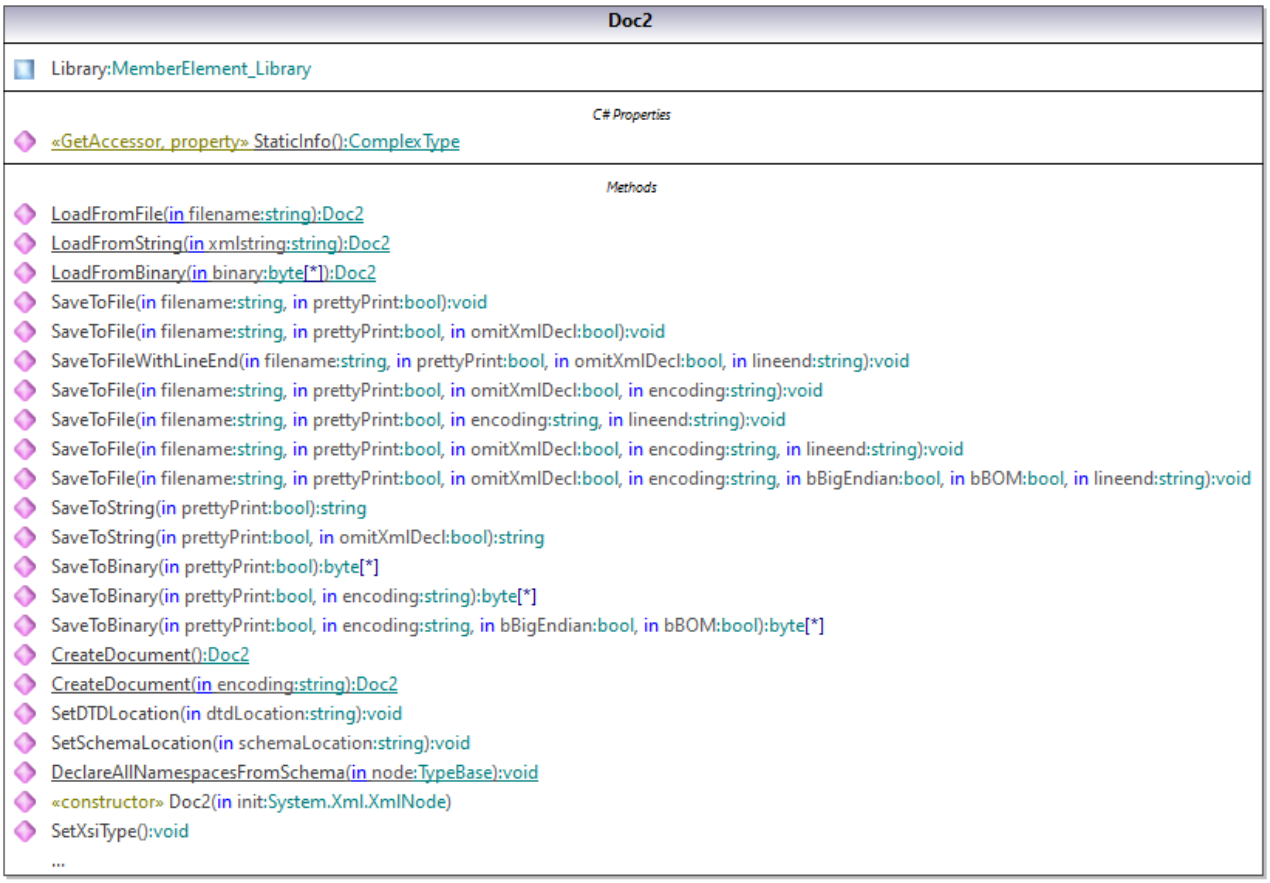
4. Press **F5** to start debugging.

### 15.2.3.5.2 Lire et écrire des documents XML (C#)

Après avoir généré du code depuis le schéma de bibliothèque (voir Schéma d'exemple 966), une application C# test est créée, avec plusieurs bibliothèques Altova.

#### À propos des bibliothèques C# générées

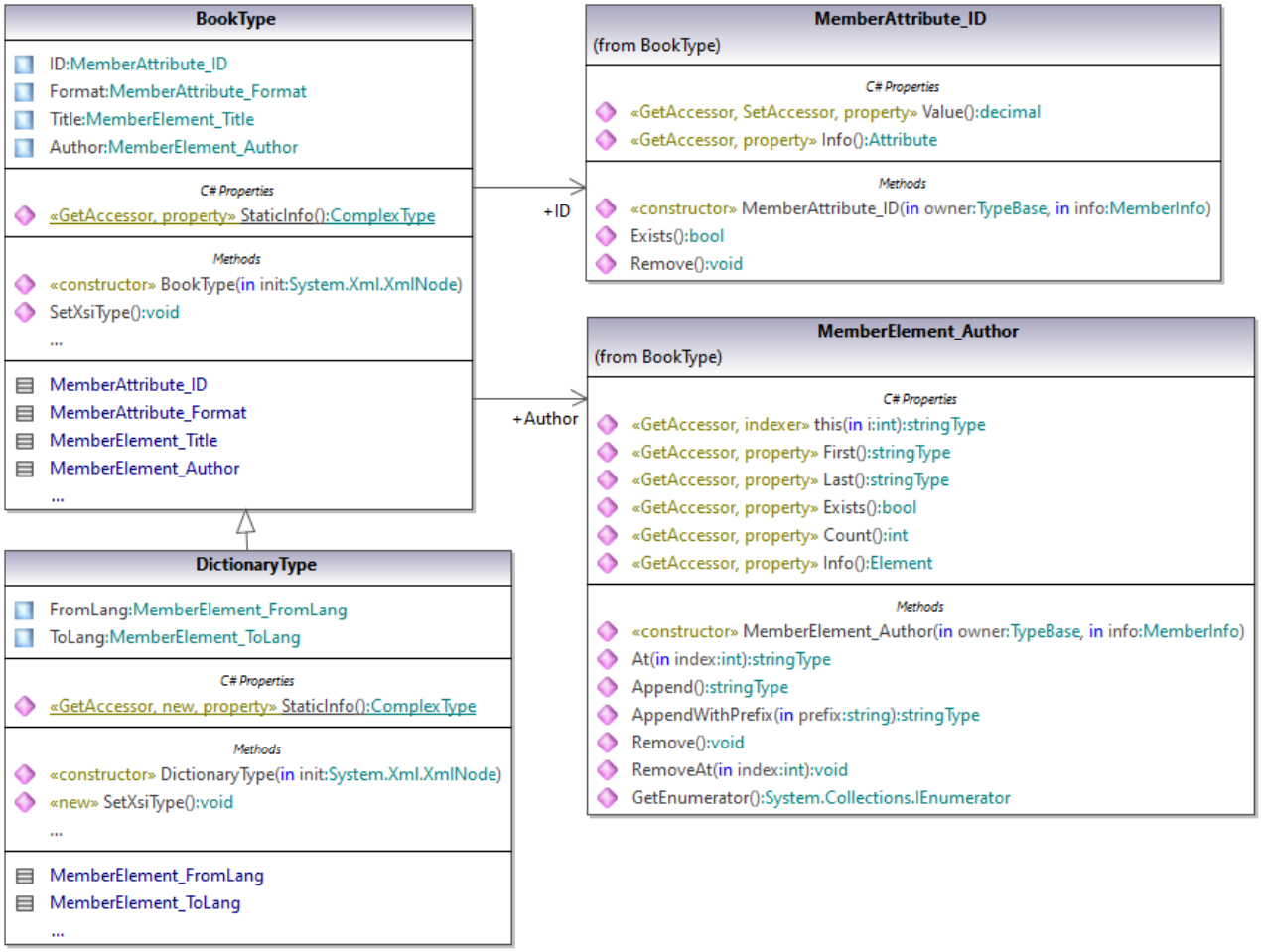
La classe centrale du code généré est la classe `Doc2`qui représente le document XML. Une telle classe est générée pour tous les schémas et son nom dépend du nom de fichier de schéma. Veuillez noter que cette classe est appelée `Doc2` pour éviter un conflit possible avec le nom d'espace de noms. Comme indiqué dans le diagramme, cette classe fournit des méthodes pour charger les documents depuis des fichiers, des streams binaires ou des strings (ou pour enregistrer des documents dans des fichiers, des streams, des strings). Pour consulter une description de cette classe, voir la référence de classe ( [YourSchema].[Doc] 1023 ).




Le membre `Library` de la classe `Doc2` représente la véritable racine du document.

Conformément aux règles de génération de code mentionnées dans À propos des Schema Wrapper Libraries (C#) 959, les membres de classes sont générés pour chaque attribut et pour chaque élément d'un type. Dans le code généré, le nom de ce type de classes de membre est préfixé avec `MemberAttribute_` et `MemberElement_`, respectivement. Des exemples de telles classes sont `MemberAttribute_ID` et `MemberElement_Author`, générés depuis l'élément **Author** et l'attribut **ID** d'un livre, respectivement (dans le diagramme ci-dessous, ils s'agit de classes imbriquées sous `BookType`). Ces classes vous permettent de manipuler à l'aide d'un programme les éléments et les attributs correspondants dans l'instance du document XML (par exemple, apposer, supprimer, définir une valeur, etc). Pour plus d'informations, voir la référence de classe `[YourSchemaType].MemberAttribute` 1026 et `[YourSchemaType].MemberElement` 1027 .

Puisque le **DictionaryType** est un type complexe dérivé depuis **BookType** dans le schéma, cette relation se retrouve aussi dans les classes générées. Comme illustré dans le diagramme, la classe `DictionaryType` hérite la classe `BookType`.




Si votre schéma XML définit des types simples en tant qu'énumérations, les valeurs énumérées deviennent disponibles en tant que valeurs `Enum` dans le code généré. Dans le schéma utilisé dans ces exemple, un format de livre peut être hardcover, paperback, e-book, etc. Dans le code généré, ces valeurs peuvent donc être disponibles via un `Enum` membre de la classe BookFormatType.

## Écrire un document XML

1. Ouvrir la solution **LibraryTest.sln** dans Visual Studio généré depuis le schéma Library mentionné plus tôt dans cet exemple.

Tout en prototypant une application depuis un schéma XML changeant fréquemment, vous devrez éventuellement générer fréquemment du code dans le même répertoire, de manière à ce que les changements de schéma sont réfléchis immédiatement dans le code. Veuillez noter que l'application de test généré et les bibliothèques Altova sont écrasées à chaque fois que vous générez du code dans le même répertoire cible. C'est pourquoi il ne faut pas ajouter du code à l'application de test généré. Au lieu de cela, veuillez intégrer les bibliothèques dans votre projet (voir Intégrer des Bibliothèques Schema Wrapper 963).

2. Dans Solution Explorer, ouvrir le fichier **LibraryTest.cs**, et éditer la méthode `Example()` comme indiqué ci-dessous.

```
protected static void Example()
{
    // Create a new XML document
    Doc2 doc = Doc2.CreateDocument();
    // Append the root element
    LibraryType root = doc.Library.Append();

    // Create the generation date using Altova DateTime class
    Altova.Types.DateTime dt = new Altova.Types.DateTime(System.DateTime.Now);
    // Append the date to the root
    root.LastUpdated.Value = dt;

    // Add a new book
    BookType book = root.Book.Append();
    // Set the value of the ID attribute
    book.ID.Value = 1;
    // Set the format of the book (enumeration)
    book.Format.EnumerationValue = BookFormatType.EnumValues.eHardcover;
    // Set the Title and Author elements
    book.Title.Append().Value = "The XMLSpy Handbook";
    book.Author.Append().Value = "Altova";

    // Append a dictionary (book of derived type) and populate its attributes and
elements
    DictionaryType dictionary = new DictionaryType(root.Book.Append().Node);
    dictionary.ID.Value = 2;
    dictionary.Title.Append().Value = "English-German Dictionary";
    dictionary.Format.EnumerationValue = BookFormatType.EnumValues.eE_book;
    dictionary.Author.Append().Value = "John Doe";
    dictionary.FromLang.Append().Value = "English";
    dictionary.ToLang.Append().Value = "German";
    // Since it's a derived type, make sure to set the xsi:type attribute of the
book element
    dictionary.SetXsiType();

    // Optionally, set the schema location (adjust the path if
    // your schema is not in the same folder as the generated instance file)
    doc.SetSchemaLocation("Library.xsd");

    // Save the XML document with the "pretty print" option enabled
    doc.SaveToFile("GeneratedLibrary.xml", true);
}
```

3. Appuyer sur **F5** pour lancer le débogage. Si le code a été exécuté avec succès, un fichier **GeneratedLibrary.xml** est créé dans le répertoire de sortie de solution (généralement, **bin/Debug**).

## Lire un document XML

1. Ouvrir la solution **LibraryTest.sln** dans Visual Studio.

2. Enregistrer le code-ci-dessous en tant que **Library.xml** dans le répertoire de sortie du projet (par défaut, **bin/Debug**). C'est le fichier qui sera lu par le code du programme.

```
<?xml version="1.0" encoding="utf-8"?>
<Library xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://www.nanonull.com/LibrarySample"
xsi:schemaLocation="http://www.nanonull.com/LibrarySample Library.xsd" LastUpdated="2016-
02-03T17:10:08.4977404">
 <Book ID="1" Format="E-book">
 <Title>The XMLSpy Handbook</Title>
 <Author>Altova</Author>
 </Book>
 <Book ID="2" Format="Paperback" xmlns:n1="http://www.nanonull.com/LibrarySample"
xsi:type="n1:DictionaryType">
 <Title>English-German Dictionary</Title>
 <Author>John Doe</Author>
 <FromLang>English</FromLang>
 <ToLang>German</ToLang>
 </Book>
</Library>
```

3. Dans Solution Explorer, ouvrir le fichier **LibraryTest.cs**, et éditer la méthode `Example()` comme indiqué ci-dessous.

```
protected static void Example()
{
    // Load the XML file
    Doc2 doc = Doc2.LoadFromFile("Library.xml");
    // Get the root element
    LibraryType root = doc.Library.First;

    // Read the library generation date
    Altova.Types.DateTime dt = root.LastUpdated.Value;
    string dt_as_string = dt.ToString(DateTimeFormat.W3_dateTime);
    Console.WriteLine("The library generation date is: " + dt_as_string);

    // Iteration: for each <Book>...
    foreach (BookType book in root.Book)
    {
      // Output values of ID attribute and (first and only) title element
      Console.WriteLine("ID:    " + book.ID.Value);
      Console.WriteLine("Title: " + book.Title.First.Value);

      // Read and compare an enumeration value
      if (book.Format.EnumerationValue == BookFormatType.EnumValues.ePaperback)
         Console.WriteLine("This is a paperback book.");

      // Iteration: for each <Author>
      foreach (xs.stringType author in book.Author)
         Console.WriteLine("Author: " + author.Value);

      // Determine if this book is of derived type
```

```
      if (book.Node.Attributes.GetNamedItem("xsi:type") != null)
      {
         // Find the value of the xsi:type attribute
         string xsiTypeValue =
book.Node.Attributes.GetNamedItem("xsi:type").Value;
         // Get the namespace URI and the lookup prefix of this namespace
         string namespaceUri = book.Node.NamespaceURI;
         string prefix = book.Node.GetPrefixOfNamespace(namespaceUri);

         // if this book has DictionaryType
         if (namespaceUri == "http://www.nanonull.com/LibrarySample" &&
xsiTypeValue.Equals(prefix + ":DictionaryType"))
         {
            // output additional fields
            DictionaryType dictionary = new DictionaryType(book.Node);
            Console.WriteLine("Language from: " +
dictionary.FromLang.First.Value);
            Console.WriteLine("Language to: " + dictionary.ToLang.First.Value);
         }
         else
         {
            throw new Exception("Unexpected book type");
         }
      }
    }

    Console.ReadLine();
}
```

4. Appuyer sur **F5** pour lancer le débogage. Si le code a été exécuté avec succès, **Library.xml** sera lu par le code de programme, et ses contenus seront affichés en tant que sortie de console.

## Lire et écrire des éléments et des attributs

Les valeurs des éléments et des attributs peut être accédé en utilisant la propriété `Value` de la classe d'élément ou d'attribut du membre généré, par exemple :

```
// Output values of ID attribute and (first and only) title element
Console.WriteLine("ID:    " + book.ID.Value);
Console.WriteLine("Title: " + book.Title.First.Value);
```

Pour obtenir la valeur de l'élément **Title** dans cet exemple particulier, nous avons aussi utilisé la méthode `First()`, étant donné que c'est le premier (et seul) élément **Title** d'un livre. Dans les cas où vous devez choisir un élément spécifique depuis une liste par index, utiliser la méthode `At()`.

La classe générée pour chaque élément de membre d'un type met en place l'interface standard `System.Collections.IEnumerable`. Cela permet de boucler dans plusieurs éléments du même type. Dans cet exemple particulier, vous pouvez boucler à travers tous les livres d'un objet Bibliothèque comme suit :

```
// Iteration: for each <Book>...
foreach (BookType book in root.Book)
```

```
{
    // your code here...
}
```

Pour ajouter un nouvel élément, utiliser la méthode `Append()`. Par exemple, le code suivant appose l'élément racine au document :

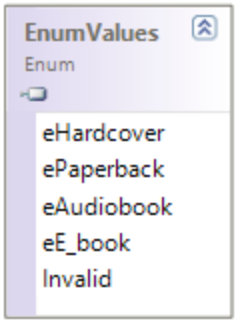
```
// Append the root element to the library
LibraryType root = doc.Library.Append();
```

Vous pouvez définir la valeur d'un attribut (comme ID dans cet exemple) comme suit :

```
// Set the value of the ID attribute
book.ID.Value = 1;
```

## Lire et écrire des valeurs d'énumération

Si votre schéma XML définit des types simples en tant qu'énumérations, les valeurs énumérées deviennent disponibles en tant que valeurs `Enum` dans le code généré. Dans le schéma utilisé dans ces exemple, un format de livre peut être hardcover, paperback, e-book, etc. Dans le code généré, ces valeurs peuvent donc être disponibles via un `Enum`.




Pour attribuer des valeurs d'énumération à un objet, utiliser du code comme celui ci-dessous :

```
// Set the format of the book (enumeration)
book.Format.EnumerationValue = BookFormatType.EnumValues.eHardcover;
```

Vous pouvez lire ces valeurs d'énumération provenant de documents d'instance XML comme suit :

```
// Read and compare an enumeration value
if (book.Format.EnumerationValue == BookFormatType.EnumValues.ePaperback)
Console.WriteLine("This is a paperback book.");
```

Si une condition "if" n'est pas suffisante, créer un interrupteur pour déterminer chaque valeur d'énumération et traiter comme requis.

## Travailler avec des types xs:dateTime et xs:duration

Si le schéma à partir duquel vous avez généré du code utilise les types heure et durée comme `xs:dateTime`, ou `xs:duration`, ils sont convertis en classes natives Altova dans un code généré. C'est pourquoi, pour écrire une valeur date ou durée vers le document XML, procéder comme suit :

1. Construit un objet `Altova.Types.DateTime`[1013] ou `Altova.Types.Duration`[1017] (soit depuis `System.DateTime`, ou en utilisant des parties comme des heures et des minutes, voir `Altova.Types.DateTime`[1013] et `Altova.Types.Duration`[1017] pour plus d'informations).
2. Définir l'objet en tant que la valeur de l'élément ou de l'attribut requis, par exemple :

```
// Create the library generation date using Altova DateTime class
Altova.Types.DateTime dt = new Altova.Types.DateTime(System.DateTime.Now);
// Append the date to the root
root.LastUpdated.Value = dt;
```

Pour lire une date ou une durée depuis un document XML, procéder comme suit :

1. Déclarer la valeur d'élément (ou d'attribut) comme objet `Altova.Types.DateTime`[1013] ou `Altova.Types.Duration`[1017] .
2. Formate l'élément ou l'attribut requis, par exemple :

```
// Read the library generation date
Altova.Types.DateTime dt = root.LastUpdated.Value;
string dt_as_string = dt.ToString(DateTimeFormat.W3_dateTime);
Console.WriteLine("The library generation date is: " + dt_as_string);
```

Pour plus d'informations, voir la référence de classe `Altova.Types.DateTime`[1013] et `Altova.Types.Duration`[1017] .

## Travailler avec des types dérivés

Si votre schéma XML définit des types dérivés, vous pouvez préserver la dérivation de type dans des documents XML que vous créez ou chargez par le biais d'un programme. Prendre le schéma utilisé dans cet exemple, l'extrait de code suivant illustre comment créer un nouveau livre de type dérivé`DictionaryType`:

```
// Append a dictionary (book of derived type) and populate its attributes and elements
DictionaryType dictionary = new DictionaryType(root.Book.Append().Node);
dictionary.ID.Value = 2;
dictionary.Title.Append().Value = "English-German Dictionary";
dictionary.Author.Append().Value = "John Doe";
dictionary.FromLanguage.Append().Value = "English";
dictionary.ToLanguage.Append().Value = "German";

// Since it's a derived type, make sure to set the xsi:type attribute of the book element
dictionary.SetXsiType();
```

Veuillez noter qu'il est important de définir l'attribut `xsi:type` du livre récemment créé. Cela garantit que le type de livre sera interprété correctement par le schéma losrque le document XML est validé.

Lorsque vous chargez des données depuis un document XML, l'extrait de code suivant montre comment identifier un livre de type dérivé `DictionaryType` dans l'instance XML chargée. Premièrement, le code trouve la valeur de l'attribut `xsi:type` du nœud de livre. Si l'espace de noms URI de ce nœud est `http://www.nanonull.com/LibrarySample` et si le préfixe et type de lookup URI correspond à la valeur de l'attribut `xsi:type`, alors il s'agit d'un dictionnaire :

```
      // Determine if this book is of derived type
      if (book.Node.Attributes.GetNamedItem("xsi:type") != null)
      {
          // Find the value of the xsi:type attribute
          string xsiTypeValue = book.Node.Attributes.GetNamedItem("xsi:type").Value;
          // Get the namespace URI and the lookup prefix of this namespace
          string namespaceUri = book.Node.NamespaceURI;
          string prefix = book.Node.GetPrefixOfNamespace(namespaceUri);

          // if this book has DictionaryType
          if (namespaceUri == "http://www.nanonull.com/LibrarySample" &&
xsiTypeValue.Equals(prefix + ":DictionaryType"))
          {
             // output additional fields
             DictionaryType dictionary = new DictionaryType(book.Node);
             Console.WriteLine("Language from: " + dictionary.FromLang.First.Value);
            Console.WriteLine("Language to: " + dictionary.ToLang.First.Value);
          }
          else
          {
              throw new Exception("Unexpected book type");
          }
      }
```

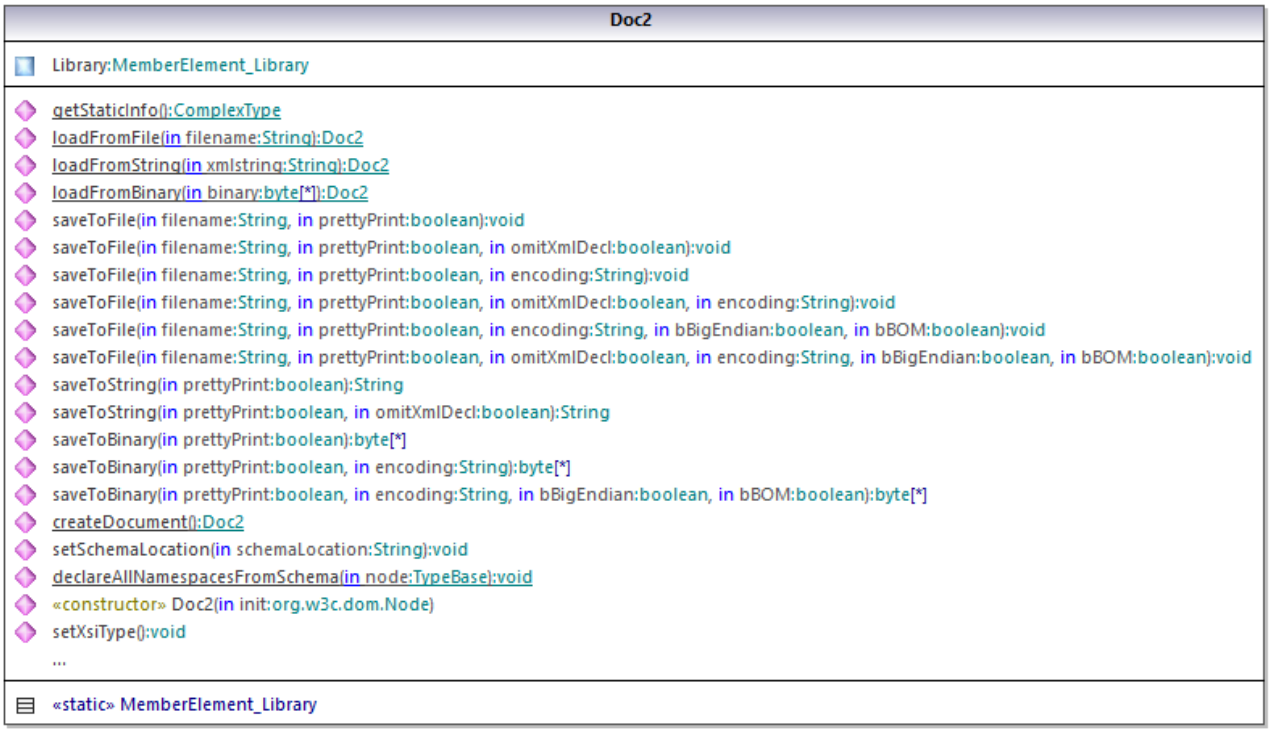
### 15.2.3.5.3 Lire et écrire des documents XML (Java)

Après avoir généré du code depuis le schéma de bibliothèque (voir Schéma d'exemple 966), un projet de text Java est créé, avec plusieurs bibliothèques Altova.

#### À propos des bibliothèques Java générées

La classe centrale du code généré est la classe `Doc2` qui représente le document XML. Une telle classe est générée pour tous les schémas et son nom dépend du nom de fichier de schéma. Veuillez noter que cette classe est appelée `Doc2` pour éviter un conflit possible avec le nom d'espace de noms. Comme indiqué dans le diagramme, cette classe fournit des méthodes pour charger les documents depuis des fichiers, des streams binaires ou des strings (ou pour enregistrer des documents dans des fichiers, des streams, des strings). Pour consulter une description de cette classe, voir la référence de classe `com.[YourSchema].[Doc]` 1038.

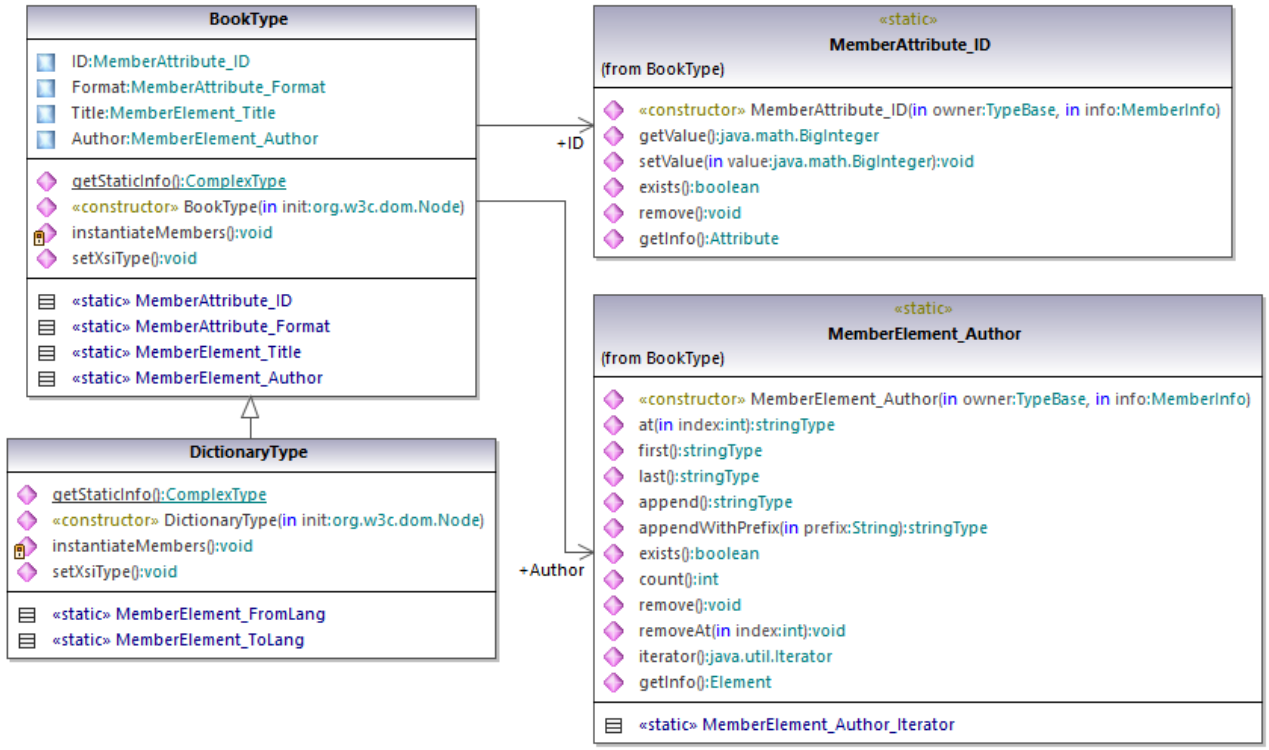
 




Le membre `Library` de la classe `Doc2` représente la véritable racine du document.

Conformément aux règles de génération de code mentionnées dans À propos du code Java généré 961, les membres de classes sont générés pour chaque attribut et pour chaque élément d'un type. Dans le code généré, le nom de ce type de classes de membre est préfixé avec `MemberAttribute_` et `MemberElement_`, respectivement. Dans le diagramme ci-dessus, des exemples de ce type de classes sont `MemberAttribute_ID` et `MemberElement_Author`, générés depuis l'élément **Author** et l'attribut **ID** d'un livre, respectivement. Ces classes vous permettent de manipuler à l'aide d'un programme les éléments et les attributs correspondants dans l'instance du document XML (par exemple, apposer, supprimer, définir une valeur, etc). Pour plus d'informations, voir la référence de classe `com.[YourSchema].[YourSchemaType].MemberAttribute` 1041 et `com.[YourSchema].[YourSchemaType].MemberElement` 1042 .

Puisque le **DictionaryType** est un type complexe dérivé depuis **BookType** dans le schéma, cette relation se retrouve aussi dans les classes générées. Comme illustré dans le diagramme, la classe `DictionaryType` hérite la classe `BookType`.




Si votre schéma XML définit des types simples en tant qu'énumerations, les valeurs énumérées deviennent disponibles en tant que valeurs `Enum` dans le code généré. Dans le schéma utilisé dans ces exemple, un format de livre peut être hardcover, paperback, e-book, etc. Dans le code généré, ces valeurs peuvent donc être disponibles via un `Enum` membre de la classe BookFormatType.

## Écrire un document XML

1. Dans le menu **Fichier** d'Eclipse, cliquer sur **Import**, choisir **Existing Projects into Workspace**, et cliquer sur **Next**.
2. À côté de **Select root directory**, cliquer sur **Browse**, choisir le répertoire dans lequel vous souhaitez générer le code Java, et cliquer sur **Finish**.
3. Dans Eclipse Package Explorer, agrandir le package **com.LibraryTest** et ouvrir le fichier **LibraryTest.java**.

Tout en prototypant une application depuis un schéma XML changeant fréquemment, vous devrez éventuellement générer fréquemment du code dans le même répertoire, de manière à ce que les changements de schéma sont réfléchis immédiatement dans le code. Veuillez noter que l'application de test généré et les bibliothèques Altova sont écrasées à chaque fois que vous générez du code dans le même répertoire cible. C'est pourquoi il ne faut pas ajouter du code à l'application de test généré. Au lieu de cela, veuillez intégrer les bibliothèques dans votre projet (voir Intégrer des Bibliothèques Schema Wrapper 963).

4. Éditer la méthode `Example()` comme indiqué ci-dessous.

```
protected static void example() throws Exception {
      // create a new, empty XML document
      Doc2 libDoc = Doc2.createDocument();

      // create the root element <Library> and add it to the document
      LibraryType lib = libDoc.Library.append();

      // set the "LastUpdated" attribute
      com.altova.types.DateTime dt = new com.altova.types.DateTime(DateTime.now());
      lib.LastUpdated.setValue(dt);

      // create a new <Book> and populate its elements and attributes
      BookType book = lib.Book.append();
      book.ID.setValue(java.math.BigInteger.valueOf(1));
      book.Format.setEnumerationValue( BookFormatType.EPAPERBACK );
      book.Title.append().setValue("The XML Spy Handbook");
      book.Author.append().setValue("Altova");

      // create a dictionary (book of derived type)  and populate its elements and
attributes
      DictionaryType dict = new DictionaryType(lib.Book.append().getNode());
      dict.ID.setValue(java.math.BigInteger.valueOf(2));
      dict.Title.append().setValue("English-German Dictionary");
      dict.Format.setEnumerationValue(BookFormatType.EE_BOOK);
      dict.Author.append().setValue("John Doe");
      dict.FromLang.append().setValue("English");
      dict.ToLang.append().setValue("German");
      dict.setXsiType();

      // set the schema location (this is optional)
      libDoc.setSchemaLocation("Library.xsd");

      // save the XML document to a file with default encoding (UTF-8). "true" causes the
file to be pretty-printed.
      libDoc.saveToFile("Library1.xml", true);
 }
```

5. Construire le projet Java et l'exécuter. Si le code a été exécuté avec succès, un fichier **Library1.xml** est créé dans le répertoire de projet.

## Lire un document XML

1. Dans le menu **Fichier** d'Eclipse, cliquer sur **Import**, choisir **Existing Projects into Workspace**, et cliquer sur **Next**.
2. À côté de **Select root directory**, cliquer sur **Browse**, choisir le répertoire dans lequel vous souhaitez générer le code Java, et cliquer sur **Finish**.
3. Enregistrer le code ci-dessous en tant que **Library1.xml** dans un répertoire local (vous devrez vous référer au chemin du fichier **Library1.xml** depuis l'échantillon de code ci-dessous).

```
<?xml version="1.0" encoding="utf-8"?>
<Library xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://www.nanonull.com/LibrarySample"
```

```
xsi:schemaLocation="http://www.nanonull.com/LibrarySample Library.xsd" LastUpdated="2016-
02-03T17:10:08.4977404">
 <Book ID="1" Format="E-book">
 <Title>The XMLSpy Handbook</Title>
 <Author>Altova</Author>
 </Book>
 <Book ID="2" Format="Paperback" xmlns:n1="http://www.nanonull.com/LibrarySample"
xsi:type="n1:DictionaryType">
 <Title>English-German Dictionary</Title>
 <Author>John Doe</Author>
 <FromLang>English</FromLang>
 <ToLang>German</ToLang>
 </Book>
</Library>
```

4. Dans Eclipse Package Explorer, agrandir le package **com.LibraryTest** et ouvrir le fichier **LibraryTest.java**.
5. Éditer la méthode `Example()` comme indiqué ci-dessous.

```
protected static void example() throws Exception {
     // load XML document from a path, make sure to adjust the path as necessary
     Doc2 libDoc = Doc2.loadFromFile("Library1.xml");

     // get the first (and only) root element <Library>
     LibraryType lib = libDoc.Library.first();

     // check whether an element exists:
     if (!lib.Book.exists()) {
        System.out.println("This library is empty.");
        return;
     }

     // read a DateTime schema type
     com.altova.types.DateTime dt = lib.LastUpdated.getValue();
     System.out.println("The library was last updated on: " + dt.toDateString());

     // iteration: for each <Book>...
     for (java.util.Iterator itBook = lib.Book.iterator(); itBook.hasNext();) {
        BookType book = (BookType) itBook.next();
        // output values of ID attribute and (first and only) title element
        System.out.println("ID: " + book.ID.getValue());
        System.out.println("Title: " + book.Title.first().getValue());

        // read and compare an enumeration value
        if (book.Format.getEnumerationValue() == BookFormatType.EPAPERBACK)
           System.out.println("This is a paperback book.");

        // for each <Author>...
        for (java.util.Iterator itAuthor = book.Author.iterator(); itAuthor
              .hasNext();)
           System.out.println("Author: " + ((com.Doc.xs.stringType)
itAuthor.next()).getValue());
```

```
        // find the derived type of this book
        // by looking at the value of the xsi:type attribute, using DOM
        org.w3c.dom.Node bookNode = book.getNode();
        if (bookNode.getAttributes().getNamedItem("xsi:type") != null) {
           // Get the value of the xsi:type attribute
           String xsiTypeValue =
bookNode.getAttributes().getNamedItem("xsi:type").getNodeValue();

           // Get the namespace URI and lookup prefix of this namespace
           String namespaceUri = bookNode.getNamespaceURI();
           String lookupPrefix = bookNode.lookupPrefix(namespaceUri);

           // If xsi:type matches the namespace URI and type of the book node
           if (namespaceUri == "http://www.nanonull.com/LibrarySample"
                 && (  xsiTypeValue.equals(lookupPrefix + ":DictionaryType" )))    {
              // ...then this is a book of derived type (dictionary)
              DictionaryType dictionary = new DictionaryType(   book.getNode());
              // output the value of the "FromLang" and "ToLang" elements
              System.out.println("From language: " +
dictionary.FromLang.first().getValue());
              System.out.println("To language: " + dictionary.ToLang.first().getValue());
           }
           else
           {
              // throw an error
              throw new java.lang.Error("This book has an unknown type.");
           }
        }
     }
   }
```

6. Construire le projet Java et l'exécuter. Si le code est exécuté avec succès, **Library1.xml** sera lu par le code de programme, et ses contenus seront affichés dans le mode Console.

## Lire et écrire des éléments et des attributs

Les valeurs des éléments et des attributs peut être accédé en utilisant la méthode `getValue()` de la classe d'élément ou d'attribut du membre généré, par exemple :

```
// output values of ID attribute and (first and only) title element
System.out.println("ID: " + book.ID.getValue());
System.out.println("Title: " + book.Title.first().getValue());
```

Pour obtenir la valeur de l'élément **Title** dans cet exemple particulier, nous avons aussi utilisé la méthode `first()`, étant donné que c'est le premier (et seul) élément **Title** d'un livre. Dans les cas où vous devez choisir un élément spécifique depuis une liste par index, utiliser la méthode `at()`.

Pour itérer dans plusieurs éléments, utiliser soit une itération basée sur l'index ou `java.util.Iterator`. Par exemple, vous pouvez itérer dans les livres d'une bibliothèque comme suit :

```
// index-based iteration
for (int j = 0; j < lib.Book.count(); ++j ) {
   // your code here
}

// alternative iteration using java.util.Iterator
for (java.util.Iterator itBook = lib.Book.iterator(); itBook.hasNext();) {
      // your code here
   }
```

Pour ajouter un nouvel élément, utiliser la méthode `Append()`. Par exemple, le code suivant appose un élément de racine vide**Library** dans le document :

```
// create the root element <Library> and add it to the document
LibraryType lib = libDoc.Library3.append();
```

Une fois qu'un élément est apposé, vous pouvez définir la valeur d'un de ses éléments ou un attribut en utilisant la méthode setValue().

```
// set the value of the Title element
book.Title.append().setValue("The XML Spy Handbook");
// set the value of the ID attribute
book.ID.setValue(java.math.BigInteger.valueOf(1));
```

## Lire et écrire des valeurs d'énumération

Si votre schéma XML définit des types simples en tant qu'énumerations, les valeurs énumérées deviennent disponibles en tant que valeurs `Enum` dans le code généré. Dans le schéma utilisé dans ces exemple, un format de livre peut être hardcover, paperback, e-book, etc. Dans le code généré, ces valeurs peuvent donc être disponibles via un `Enum` (voir le diagramme de classe BookFormatType `ci-dessus)`. Pour attribuer des valeurs d'énumération à un objet, utiliser du code comme celui ci-dessous :

```
// set an enumeration value
book.Format.setEnumerationValue( BookFormatType.EPAPERBACK );
```

Vous pouvez lire ces valeurs d'énumération provenant de documents d'instance XML comme suit :

```
// read an enumeration value
if (book.Format.getEnumerationValue() == BookFormatType.EPAPERBACK)
 System.out.println("This is a paperback book."
```

Si une condition "if" n'est pas suffisante, créer un interrupteur pour déterminer chaque valeur d'énumération et traiter comme requis.

## Travailler avec des types xs:dateTime et xs:duration

Si le schéma à partir duquel vous avez généré du code utilise les types heure et durée comme `xs:dateTime`, ou `xs:duration`, ils sont convertis en classes natives Altova dans un code généré. C'est pourquoi, pour écrire une valeur date ou durée vers le document XML, procéder comme suit :

1. Construit un objet com.altova.types.DateTime[1028] ou com.altova.types.Duration[1032] .
2. Définir l'objet en tant que la valeur de l'élément ou de l'attribut requis, par exemple :

```
// set the value of an attribute of DateTime type
com.altova.types.DateTime dt = new com.altova.types.DateTime(DateTime.now());
lib.LastUpdated.setValue(dt);
```

Pour lire une date ou une durée depuis un document XML :

1. Déclarer la valeur d'élément (ou d'attribut) comme objet com.altova.types.DateTime[1028] ou com.altova.types.Duration[1032] .
2. Formate l'élément ou l'attribut requis, par exemple :

```
// read a DateTime type
com.altova.types.DateTime dt = lib.LastUpdated.getValue();
 System.out.println("The library was last updated on: " + dt.toDateString());
```

Pour plus d'informations, voir la référence de classe com.altova.types.DateTime[1028] et com.altova.types.Duration[1032] .

## Travailler avec des types dérivés

Si votre schéma XML définit des types dérivés, vous pouvez préserver la dérivation de type dans des documents XML que vous créez ou chargez par le biais d'un programme. Prendre le schéma utilisé dans cet exemple, l'extrait de code suivant illustre comment créer un nouveau livre de type dérivé`DictionaryType`:

```
// create a dictionary (book of derived type) and populate its elements and attributes
DictionaryType dict = new DictionaryType(lib.Book.append().getNode());
dict.ID.setValue(java.math.BigInteger.valueOf(2));
dict.Title.append().setValue("English-German Dictionary");
dict.Format.setEnumerationValue(BookFormatType.EE_BOOK);
dict.Author.append().setValue("John Doe");
dict.FromLang.append().setValue("English");
dict.ToLang.append().setValue("German");
dict.setXsiType();
```

Veuillez noter qu'il est important de définir l'attribut `xsi:type` du livre récemment créé. Cela garantit que le type de livre sera interprété correctement par le schéma losrque le document XML est validé.

Lorsque vous chargez des données depuis un document XML, l'extrait de code suivant montre comment identifier un livre de type dérivé `DictionaryType` dans l'instance XML chargée. Premièrement, le code trouve la valeur de l'attribut `xsi:type` du nœud de livre. Si l'espace de noms URI de ce nœud est `http://www.nanonull.com/LibrarySample` et si le préfixe et type de lookup URI correspond à la valeur de l'attribut `xsi:type`, alors il s'agit d'un dictionnaire :

```
        // find the derived type of this book
 // by looking at the value of the xsi:type attribute, using DOM
 org.w3c.dom.Node bookNode = book.getNode();
 if (bookNode.getAttributes().getNamedItem("xsi:type") != null) {
```

```
// Get the value of the xsi:type attribute
String xsiTypeValue = bookNode.getAttributes().getNamedItem("xsi:type").getNodeValue();

// Get the namespace URI and lookup prefix of the book node
String namespaceUri = bookNode.getNamespaceURI();
String lookupPrefix = bookNode.lookupPrefix(namespaceUri);

// If xsi:type matches the namespace URI and type of the book node
if (namespaceUri == "http://www.nanonull.com/LibrarySample"
&& ( xsiTypeValue.equals(lookupPrefix + ":DictionaryType" ))) {
// ...then this is a book of derived type (dictionary)
DictionaryType dictionary = new DictionaryType( book.getNode());
// output the value of the "FromLang" and "ToLang" elements
System.out.println("From language: " + dictionary.FromLang.first().getValue());
System.out.println("To language: " + dictionary.ToLang.first().getValue());
}
else
{
// throw an error
throw new java.lang.Error("This book has an unknown type.");
}
}
```

### 15.2.3.6 Exemple: Purchase Order

Cet exemple illustre comment travailler avec le code de programme généré depuis un schéma XML "main" qui importe d'autres schémas. Chacun des schémas importés a un espace de noms cible différent. L'objectif est de créer par programme un document XML dans lequel tous les éléments sont préfixés conformément à leur espace de noms. Plus précisément, le document XML créé depuis votre code C++, C# ou Java doit ressembler à l'exemple ci-dessous :

```
<?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/OrderTypes"
            xmlns:c="http://NamespaceTest.com/CustomerTypes"
            xmlns:cmn="http://NamespaceTest.com/CommonTypes"
            xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <p:OrderDetail>
        <o:Item>
            <o:ProductName>Lawnmower</o:ProductName>
            <o:Quantity>1</o:Quantity>
            <o:UnitPrice>148.42</o:UnitPrice>
        </o:Item>
    </p:OrderDetail>
    <p:PaymentMethod>VISA</p:PaymentMethod>
    <p:CustomerDetails>
        <c:Name>Alice Smith</c:Name>
        <c:DeliveryAddress>
            <cmn:Line1>123 Maple Street</cmn:Line1>
```

```
            <cmn:Line2>Mill Valley</cmn:Line2>
        </c:DeliveryAddress>
        <c:BillingAddress>
            <cmn:Line1>8 Oak Avenue</cmn:Line1>
            <cmn:Line2>Old Town</cmn:Line2>
        </c:BillingAddress>
    </p:CustomerDetails>
</p:Purchase>
```

Le schéma principal utilisé dans ces exemple est appelé **Main.xsd**. Comme indiqué dans l'extrait de code ci-dessous, il importe trois autres schémas : **CommonTypes.xsd**, **CustomerTypes.xsd** et **OrderTypes.xsd**. Pour obtenir les mêmes résultats que dans cet exemple, enregistrer tous les extraits de code ci-dessous dans des fichiers et utiliser les mêmes noms de fichier que ci-dessus. Veuillez noter que le schéma mappe chacun des préfixes **ord**, **pur**, **cmn**, et **cust** vers des espaces de noms (Order types, Purchase types, Common types, et Customer types, respectivement). Cela signifie que, dans le code généré, les classes correspondant à Orders, Purchases, Customers, etc, seront disponibles sous leur espace de noms respectif.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
      targetNamespace="http://NamespaceTest.com/Purchase"
      xmlns:ord="http://NamespaceTest.com/OrderTypes"
      xmlns:pur="http://NamespaceTest.com/Purchase"
      xmlns:cmn="http://NamespaceTest.com/CommonTypes"
      xmlns:cust="http://NamespaceTest.com/CustomerTypes"
      elementFormDefault="qualified">
    <xs:import schemaLocation="CommonTypes.xsd"
namespace="http://NamespaceTest.com/CommonTypes" />
    <xs:import schemaLocation="CustomerTypes.xsd"
namespace="http://NamespaceTest.com/CustomerTypes" />
    <xs:import schemaLocation="OrderTypes.xsd"
namespace="http://NamespaceTest.com/OrderTypes" />
    <xs:element name="Purchase">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="OrderDetail" type="ord:OrderType" />
                <xs:element name="PaymentMethod" type="cmn:PaymentMethodType" />
                <xs:element ref="pur:CustomerDetails" />
            </xs:sequence>
        </xs:complexType>
    </xs:element>
    <xs:element name="CustomerDetails" type="cust:CustomerType" />
</xs:schema>
```

*Main.xsd*

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
      targetNamespace="http://NamespaceTest.com/CommonTypes"
      elementFormDefault="qualified">
   <xs:complexType name="AddressType">
      <xs:sequence>
```

```
        <xs:element name="Line1" type="xs:string"/>
        <xs:element name="Line2" type="xs:string"/>
      </xs:sequence>
   </xs:complexType>
   <xs:simpleType name="PriceType">
      <xs:restriction base="xs:decimal">
         <xs:fractionDigits value="2"/>
      </xs:restriction>
   </xs:simpleType>
   <xs:simpleType name="PaymentMethodType">
      <xs:restriction base="xs:string">
         <xs:enumeration value="VISA"/>
         <xs:enumeration value="MasterCard"/>
         <xs:enumeration value="Cash"/>
         <xs:enumeration value="AMEX"/>
      </xs:restriction>
   </xs:simpleType>
</xs:schema>
```

*CommonTypes.xsd*

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
      targetNamespace="http://NamespaceTest.com/CustomerTypes"
      xmlns:cmn="http://NamespaceTest.com/CommonTypes"
      elementFormDefault="qualified">
    <xs:import schemaLocation="CommonTypes.xsd"
namespace="http://NamespaceTest.com/CommonTypes" />
    <xs:complexType name="CustomerType">
        <xs:sequence>
            <xs:element name="Name" type="xs:string" />
            <xs:element name="DeliveryAddress" type="cmn:AddressType" />
            <xs:element name="BillingAddress" type="cmn:AddressType" />
        </xs:sequence>
    </xs:complexType>
</xs:schema>
```

*CustomerTypes.xsd*

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
      targetNamespace="http://NamespaceTest.com/OrderTypes"
      xmlns:cmn="http://NamespaceTest.com/CommonTypes"
      elementFormDefault="qualified">
    <xs:import schemaLocation="CommonTypes.xsd"
namespace="http://NamespaceTest.com/CommonTypes" />
    <xs:complexType name="OrderType">
        <xs:sequence>
            <xs:element maxOccurs="unbounded" name="Item">
                <xs:complexType>
                    <xs:sequence>
```

```
                    <xs:element name="ProductName" type="xs:string" />
                    <xs:element name="Quantity" type="xs:int" />
                    <xs:element name="UnitPrice" type="cmn:PriceType" />
                </xs:sequence>
            </xs:complexType>
        </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:schema>
```

*OrderTypes.xsd*

Pour terminer cet exemple, suivre les étapes suivantes :

1. Enregistrer tous les schémas depuis les extraits de code ci-dessus dans les fichiers sur le disque, en vous assurant que vous préservez les noms de fichier indiqués.
2. Générer le code de schema wrapper depuis le schéma **Main.xsd** ci-dessus, en utilisant les étapes décrites dans Générer du code depuis des Schémas XML ou des DTD[953]. Une fois cette étape achevée , vous aurez généré un programme compilable dans le langage de votre choix (C++, C#, ou Java).
3. Ajouter du code dans votre programme C++, C# ou Java depuis un des extraits de code d'exemple, le cas échéant :

   - Espaces de noms XML et Préfixes (C++)[993]
   - Espaces de noms XML et Préfixes (C#)[995]
   - Espaces de noms XML et Préfixes (Java)[996]

### 15.2.3.6.1 Espaces de noms XML et Préfixes (C++)

Après avoir généré du code depuis le schéma d'exemple[990], une application C++ test est créée, avec plusieurs bibliothèques Altova de support. Rappelez-vous que le schéma d'exemple (**Main.xsd**) a plusieurs déclarations d'espace de noms. Par conséquent, le code généré comprend des espaces de noms qui correspondent aux alias d'espace de noms (préfixes) provenant du schéma : `Main::ord`, `Main::pur`, `Main::cmn` et `Main::cust`.

En général, afin de contrôler les espaces de noms XML et les préfixes avec l'aide des bibliothèques de schéma, vous disposez des méthodes suivantes :

- DeclareAllNamespacesFromSchema()[1008]. Appeler cette méthode si vous souhaitez déclarer les mêmes espaces de noms dans votre instance XML que dans le schéma. Sinon, si vous souhaitez utiliser d'autres espaces de noms que ceux utilisés dans cet exemple, vous devriez utiliser `DeclareNamespace()`. La méthode `DeclareAllNamespacesFromSchema()` n'est pas utilisée dans cet exemple parce que nous souhaitons utiliser spécifiquement des éléments XML avec des préfixes qui sont légèrement différents de ceux déclarés dans le schéma.
- DeclareNamespace()[1010]. Appeler cette méthode pour créer ou contourner l'attribut de préfixe d'espace de noms existant sur un élément. L'élément doit déjà être créé en utilisant soit la méthode `append()` soit `appendWithPrefix()`, comme illustré ci-dessous.
- appendWithPrefix().[1012] Utiliser cette méthode pour apposer un élément d'instance avec un préfixe spécifique. Pour créer l'instance XML illustrée dans cet exemple, il a suffit d'appeler cette méthode pour l'élément root uniquement. Tous les autres éléments ont été apposés en utilisant uniquement

append()[1012], et leurs préfixes ont été ajoutés automatiquement sur la base de leurs espaces de noms, conformément aux règles ci-dessus.

L'extrait de code ci-dessous vous montre comment créer un document XML avec plusieurs déclarations d'espace de noms et des noms d'élément préfixés. En particulier, il génère une instance Purchase Order comme illustré dans l'Example : Purchase Order[990]. Plus important encore, à des fins d'illustration, certains préfixes sont contournés dans l'instance XML (c'est à dire qu'ils ne sont tout à fait identiques que ceux déclarés dans le schéma).

```
void Example()
{
   // Create the XML document and append the root element
   Main::pur::CMain doc = Main::pur::CMain::CreateDocument();
   Main::pur::CPurchaseType purchase = doc.Purchase.appendWithPrefix(_T("p"));

   // Set schema location
   doc.SetSchemaLocation(_T("Main.xsd"));

   // Declare namespaces on root element
   purchase.DeclareNamespace(_T("o"), _T("http://NamespaceTest.com/OrderTypes"));
   purchase.DeclareNamespace(_T("c"), _T("http://NamespaceTest.com/CustomerTypes"));
   purchase.DeclareNamespace(_T("cmn"), _T("http://NamespaceTest.com/CommonTypes"));

   // Append the OrderDetail element
   Main::ord::COrderType order = purchase.OrderDetail.append();
   Main::ord::CItemType item = order.Item.append();
   item.ProductName.append() = _T("Lawnmower");
   item.Quantity.append() = 1;
   item.UnitPrice.append() = 148.42;

   // Append the PaymentMethod element
   Main::cmn::CPaymentMethodTypeType paymentMethod = purchase.PaymentMethod.append();
   paymentMethod.SetEnumerationValue(Main::cmn::CPaymentMethodTypeType::k_VISA);

   // Append the CustomerDetails element
   Main::cust::CCustomerType customer = purchase.CustomerDetails.append();
   customer.Name.append() = _T("Alice Smith");
   Main::cmn::CAddressType deliveryAddress = customer.DeliveryAddress.append();
   deliveryAddress.Line1.append() = _T("123 Maple Street");
   deliveryAddress.Line2.append() = _T("Mill Valley");
   Main::cmn::CAddressType billingAddress = customer.BillingAddress.append();
   billingAddress.Line1.append() = _T("8 Oak Avenue");
   billingAddress.Line2.append() = _T("Old Town");

   // Save to file and release object from memory
   doc.SaveToFile(_T("Main1.xml"), true);
   doc.DestroyDocument();
}
```

### 15.2.3.6.2 Espaces de noms XML et Préfixes (C#)

Après avoir généré du code depuis le schéma de bibliothèque (voir Schéma d'exemple[990]), une application C# test est créée, avec plusieurs bibliothèques Altova. Rappelez-vous que le schéma d'exemple (**Main.xsd**) a plusieurs déclarations d'espace de noms. Par conséquent, le code généré comprend des espaces de noms qui correspondent aux alias d'espace de noms (préfixes) provenant du schéma : **`Main.ord`**, **`Main.pur`**, **`Main.cmn`**, et **`Main.cust`**.

En général, afin de contrôler les espaces de noms XML et les préfixes avec l'aide des bibliothèques de schéma, vous disposez des méthodes suivantes :

- DeclareAllNamespacesFromSchema()[1023]. Appeler cette méthode si vous souhaitez déclarer les mêmes espaces de noms dans votre instance XML que dans le schéma. Sinon, si vous souhaitez utiliser d'autres espaces de noms que ceux utilisés dans cet exemple, vous devriez utiliser **`DeclareNamespace()`**. La méthode **`DeclareAllNamespacesFromSchema()`** n'est pas utilisée dans cet exemple parce que nous souhaitons utiliser spécifiquement des éléments XML avec des préfixes qui sont légèrement différents de ceux déclarés dans le schéma.
- DeclareNamespace()[1025]. Appeler cette méthode pour créer ou contourner l'attribut de préfixe d'espace de noms existant sur un élément. L'élément doit déjà être créé en utilisant soit la méthode **`Append()`** soit **`AppendWithPrefix()`**, comme illustré ci-dessous.
- AppendWithPrefix().[1027] Utiliser cette méthode pour apposer un élément d'instance avec un préfixe spécifique. Pour créer l'instance XML illustrée dans cet exemple, il a suffit d'appeler cette méthode pour l'élément root uniquement. Tous les autres éléments ont été apposés en utilisant uniquement Append()[1027], et leurs préfixes ont été ajoutés automatiquement sur la base de leurs espaces de noms, conformément aux règles ci-dessus.

L'extrait de code ci-dessous vous montre comment créer un document XML avec plusieurs déclarations d'espace de noms et des noms d'élément préfixés. En particulier, il génère une instance Purchase Order comme illustré dans l'Example : Purchase Order[990]. Plus important encore, à des fins d'illustration, certains préfixes sont contournés dans l'instance XML (c'est à dire qu'ils ne sont tout à fait identiques que ceux déclarés dans le schéma).

```
protected static void Example()
{
   // Create the XML document and append the root element
   pur.Main2 doc = pur.Main2.CreateDocument();
   pur.PurchaseType purchase = doc.Purchase.AppendWithPrefix("p");

   // Set schema location
   doc.SetSchemaLocation(@"Main.xsd");

   // Declare namespaces on root element
   purchase.DeclareNamespace("o", "http://NamespaceTest.com/OrderTypes");
   purchase.DeclareNamespace("c", "http://NamespaceTest.com/CustomerTypes");
   purchase.DeclareNamespace("cmn", "http://NamespaceTest.com/CommonTypes");

   // Append the OrderDetail element
   ord.OrderType order = purchase.OrderDetail.Append();
   ord.ItemType item = order.Item.Append();
   item.ProductName.Append().Value = "Lawnmower";
   item.Quantity.Append().Value = 1;
```

```
    item.UnitPrice.Append().Value = 148.42M;

    // Append the PaymentMethod element
    cmn.PaymentMethodTypeType paymentMethod = purchase.PaymentMethod.Append();
    paymentMethod.EnumerationValue = cmn.PaymentMethodTypeType.EnumValues.eVISA;

    // Append the CustomerDetails element
    cust.CustomerType customer = purchase.CustomerDetails.Append();
    customer.Name.Append().Value = "Alice Smith";
    cmn.AddressType deliveryAddress = customer.DeliveryAddress.Append();
    deliveryAddress.Line1.Append().Value = "123 Maple Street";
    deliveryAddress.Line2.Append().Value = "Mill Valley";
    cmn.AddressType billingAddress = customer.BillingAddress.Append();
    billingAddress.Line1.Append().Value = "8 Oak Avenue";
    billingAddress.Line2.Append().Value = "Old Town";

    // Save to file
    doc.SaveToFile("PurchaseOrder.xml", true);
}
```

### 15.2.3.6.3 Espaces de noms XML et Préfixes (Java)

Après avoir généré du code depuis le schéma de bibliothèque (voir Schéma d'exemple 990), une application Java de test est créée, avec plusieurs bibliothèques Altova. Rappelez-vous que le schéma d'exemple (**Main.xsd**) a plusieurs déclarations d'espace de noms. Par conséquent, le code généré comprend des espaces de noms qui correspondent aux alias d'espace de noms (préfixes) provenant du schéma : **`com.Main.ord`**, **`com.Main.pur`**, **`com.Main.cmn`**, et **`com.Main.cust`**.

En général, afin de contrôler les espaces de noms XML et les préfixes avec l'aide des bibliothèques de schéma, vous disposez des méthodes suivantes :

- declareAllNamespacesFromSchema() 1038. Appeler cette méthode si vous souhaitez déclarer les mêmes espaces de noms dans votre instance XML que dans le schéma. Sinon, si vous souhaitez utiliser d'autres espaces de noms que ceux utilisés dans cet exemple, vous devriez utiliser **`declareNamespace()`**. La méthode **`declareAllNamespacesFromSchema()`** n'est pas utilisée dans cet exemple parce que nous souhaitons utiliser spécifiquement des éléments XML avec des préfixes qui sont légèrement différents de ceux déclarés dans le schéma.
- declareNamespace() 1040. Appeler cette méthode pour créer ou contourner l'attribut de préfixe d'espace de noms existant sur un élément. L'élément doit déjà être créé en utilisant soit la méthode **`append()`** soit **`appendWithPrefix()`**, comme illustré ci-dessous.
- appendWithPrefix(). 1042 Utiliser cette méthode pour apposer un élément d'instance avec un préfixe spécifique. Pour créer l'instance XML illustrée dans cet exemple, il a suffit d'appeler cette méthode pour l'élément root uniquement. Tous les autres éléments ont été apposés en utilisant uniquement append() 1042, et leurs préfixes ont été ajoutés automatiquement sur la base de leurs espaces de noms, conformément aux règles ci-dessus.

L'extrait de code ci-dessous vous montre comment créer un document XML avec plusieurs déclarations d'espace de noms et des noms d'élément préfixés. En particulier, il génère une instance Purchase Order comme illustré dans l'Example : Purchase Order 990. Plus important encore, à des fins d'illustration, certains préfixes sont contournés dans l'instance XML (c'est à dire qu'ils ne sont tout à fait identiques que ceux déclarés dans le schéma).

```
protected static void example() throws Exception {
   // Create the XML document and append the root element
   com.Main.pur.Main2 doc = com.Main.pur.Main2.createDocument();
   com.Main.pur.PurchaseType purchase = doc.Purchase.appendWithPrefix("p");

   // Set schema location
   doc.setSchemaLocation("Main.xsd");

   // Declare namespaces on root element
   purchase.declareNamespace("o", "http://NamespaceTest.com/OrderTypes");
   purchase.declareNamespace("c", "http://NamespaceTest.com/CustomerTypes");
   purchase.declareNamespace("cmn", "http://NamespaceTest.com/CommonTypes");

   // Append the OrderDetail element
   com.Main.ord.OrderType order = purchase.OrderDetail.append();
   com.Main.ord.ItemType item = order.Item.append();
   item.ProductName.append().setValue("Lawnmower");
   item.Quantity.append().setValue(1);
   java.math.BigDecimal price = new java.math.BigDecimal("148.42");
   item.UnitPrice.append().setValue(price);

   // Append the PaymentMethod element
   com.Main.cmn.PaymentMethodTypeType paymentMethod = purchase.PaymentMethod.append();
   paymentMethod.setEnumerationValue(com.Main.cmn.PaymentMethodTypeType.EVISA);

   // Append the CustomerDetails element
   com.Main.cust.CustomerType customer = purchase.CustomerDetails.append();
   customer.Name.append().setValue("Alice Smith");
   com.Main.cmn.AddressType deliveryAddress = customer.DeliveryAddress.append();
   deliveryAddress.Line1.append().setValue("123 Maple Street");
   deliveryAddress.Line2.append().setValue("Mill Valley");
   com.Main.cmn.AddressType billingAddress = customer.BillingAddress.append();
   billingAddress.Line1.append().setValue("8 Oak Avenue");
   billingAddress.Line2.append().setValue("Old Town");

   // Save to file
   doc.saveToFile("PurchaseOrder.xml", true);
}
```

## 15.2.4 Référence aux classes générées (C++)

Ce chapitre contient une description des classes C++ avec MapForce depuis un Schéma XML ou DTD (voir Générer du code depuis des Schémas XML ou DTDs 953 ). Vous pouvez intégrer ces classes dans votre code pour lire, modifier et écrire des documentation XML.

**Note:** Le code généré peut inclure d'autres classes de support, qui ne sont pas recensées ici et qui sont sujettes à des modifications.

### 15.2.4.1 altova::DateTime

Cette classe vous permet de traiter des attributs XML ou des éléments qui ont des types de date et d'heure, comme `xs:dateTime`.

#### Constructeurs

| Nom | Description |
|---|---|
| `DateTime()` | Initialise une nouvelle instance de la classe `DateTime` à 12:00:00 minuit, le 1er janvier, 0001. |
| `DateTime(__int64 value, short timezone)` | Initialise une nouvelle instance de la classe `DateTime`. Le paramètre `value` représente le nombre de tics (intervales de 100 nanosecondes) qui se sont écoulés depuis 12:00:00 minuit, le 1er janvier, 0001. |
| `DateTime(int year, unsigned char month, unsigned char day, unsigned char hour, unsigned char minute, double second)` | Initialise une nouvelle instance de la classe `DateTime` à l'année, mois, jour, heure, minute et seconde fournis en tant qu'argument. |
| `DateTime(int year, unsigned char month, unsigned char day, unsigned char hour, unsigned char minute, double second, short timezone)` | Initialise une nouvelle instance de la classe `DateTime` à l'année, mois, jour, heure, minute, seconde et fuseau horaire fournis en tant qu'argument. Le fuseau horaire est exprimé en minutes et peut être positif ou négatif. Par exemple, le fuseau horaire "UTC-01:00" est exprimé en tant que "-60". |

#### Méthodes

| Nom | Description |
|---|---|
| `unsigned char Day() const` | Retourne le jour du mois de l'objet `DateTime` actuel. Les valeurs de retour s'étendent de 1 à 31. |
| `int DayOfYear() const` | Retourne le jour de l'année de l'objet `DateTime` actuel. Les valeurs de retour s'étendent de 1 à 366. |
| `bool HasTimezone() const` | Retourne Booléenne **true** si l'objet `DateTime` actuel présente un fuseau horaire défini ; **false** sinon. |
| `unsigned char Hour() const` | Retourne l'heure de l'objet `DateTime` actuel. Les valeurs de retour s'étendent de 0 à 23. |
| `static bool IsLeapYear(int year)` | Retourne Booléenne **true** si l'année de la classe `DateTime` est une année bissextile ; **false** sinon. |
| `unsigned char Minute() const` | Retourne la minute de l'objet `DateTime` actuel. Les valeurs de retour s'étendent de 0 à 59. |

| Nom | Description |
|---|---|
| `unsigned char Month() const` | Retourne le mois de l'objet `DateTime` actuel. Les valeurs de retour s'étendent de 1 à 12. |
| `__int64 NormalizedValue() const` | Retourne la valeur de l'objet `DateTime` exprimé en tant que Coordinated Universal Time (UTC). |
| `double Second() const` | Retourne la seconde de l'objet `DateTime` actuel. Les valeurs de retour s'étendent de 0 à 59. |
| `void SetTimezone(short tz)` | Définit le fuseau horaire de l'objet `DateTime` actuel à la valeur du fuseau horaire fourni en tant qu'argument. L'argument **tz** est exprimé en minutes et peut êtr positif ou négatif. |
| `short Timezone() const` | Retourne le fuseau horaire, en minutes de l'objet `DateTime` actuel. Avant d'utiliser cette méthode, veuillez vous assurer que l'objet comporte réellement un fuseau horaire, en appelan la méthode `HasTimezone()`. |
| `__int64 Value() const` | Retourne la valeur de l'objet `DateTime` object, exprimé dans le nombre de tics (intervalles de 100 nanosecondes) qui se sont écoulés depuis 12:00:00 minuit, le 1er janvier, 0001. |
| `int Weekday() const` | Retourne le jour de la semaine de l'objet `DateTime` actuel, en tant qu'un entier. Les valeurs s'étendent de 0 à 6, 0 étant lundi (ISO-8601). |
| `int Weeknumber() const` | Retourne le nombre de semaines dans l'année de l'objet `DateTime` actuel. Les valeurs de retour sont conformes à ISO-8601. |
| `int WeekOfMonth() const` | Retourne le nombre de semaines dans le mois de l'objet `DateTime` actuel. Les valeurs de retour sont conformes à ISO-8601. |
| `int Year() const` | Retoune l'année de l'objet `DateTime` actuel. |

## Exemple

```
void Example()
{
   // initialize a new DateTime instance to 12:00:00 midnight, January 1st, 0001
   altova::DateTime dt1 = altova::DateTime();

   // initialize a new DateTime instance using the year, month, day, hour, minute, and
second
   altova::DateTime dt2 = altova::DateTime(2015, 11, 10, 9, 8, 7);

   // initialize a new DateTime instance using the year, month, day, hour, minute,
second, and UTC +01:00 timezone
    altova::DateTime dt = altova::DateTime(2015, 11, 22, 13, 53, 7, 60);

   // Get the value of this DateTime object
   std::cout << "The number of ticks of the DateTime object is: " << dt.Value() <<
```

```
std::endl;

   // Get the year
   cout << "The year is: " << dt.Year() << endl;
   // Get the month
   cout << "The month is: " << (int)dt.Month() << endl;
   // Get the day of the month
   cout << "The day of the month is: " << (int) dt.Day() << endl;
   // Get the day of the year
   cout << "The day of the year is: " << dt.DayOfYear() << endl;
   // Get the hour
   cout << "The hour is: " << (int) dt.Hour() << endl;
   // Get the minute
   cout << "The minute is: " << (int) dt.Minute() << endl;
   // Get the second
   cout << "The second is: " << dt.Second() << endl;
   // Get the weekday
   cout << "The weekday is: " << dt.Weekday() << endl;
   // Get the week number
   cout << "The week of year is: " << dt.Weeknumber() << endl;
   // Get the week in month
   cout << "The week of month is: " << dt.WeekOfMonth() << endl;

   // Check whether a DateTime instance has a timezone
   if (dt.HasTimezone() == TRUE)
   {
      // output the value of the Timezone
      cout <<  "The timezone is: " << dt.Timezone() << endl;
   }
   else
   {
      cout <<  "No timezone has been defined." << endl;
   }

   // Construct a DateTime object with a timezone UTC+01:00 (Vienna)
   altova::DateTime vienna_dt = DateTime(2015, 11, 23, 14, 30, 59, +60);
   // Output the result in readable format
   cout << "The Vienna time: "
       << (int) vienna_dt.Month()
       << "-" << (int) vienna_dt.Day()
       << " " << (int) vienna_dt.Hour()
       << ":" << (int) vienna_dt.Minute()
       << ":" << (int) vienna_dt.Second()
       << endl;

   // Convert the value to UTC time
   DateTime utc_dt = DateTime(vienna_dt.NormalizedValue());
   // Output the result in readable format
   cout << "The UTC time:    "
      << (int) utc_dt.Month()
      << "-" << (int) utc_dt.Day()
      << " " << (int) utc_dt.Hour()
      << ":" << (int) utc_dt.Minute()
```

```
      << ":" << (int) utc_dt.Second()
      << endl;

   // Check if a year is a leap year
   int year = 2016;
   if( altova::DateTime::IsLeapYear(year) )
   { cout << year << " is a leap year" << endl; }
   else
   { cout << year << " is not a leap year" << endl; }
}
```

## 15.2.4.2 altova::Duration

Cette classe vous permet de traiter des attributs XML ou des éléments de type `xs: duration` .

### Constructeurs

| Nom | Description |
|---|---|
| `Duration()` | Initialise une nouvelle instance de la classe `Duration` dans une valeur vide. |
| `Duration(const DayTimeDuration& dt)` | Initialise une nouvelle instance de la classe `Duration` dans une durée définie par l'argument `dt` (voir altova::DayTimeDuration 1003 ). |
| `Duration(const YearMonthDuration& ym)` | Initialise une nouvelle instance de la classe `Duration` dans la durée définie par l'argument `ym` (voir altova::YearMonthDuration 1003 ). |
| `Duration(const YearMonthDuration& ym, const DayTimeDuration& dt)` | Initialise une nouvelle instance de la classe `Duration` dans les durées définies par les arguments `dt` et `ym` (voir altova::YearMonthDuration 1003 et altova::DayTimeDuration 1003 ). |

### Méthodes

| Nom | Description |
|---|---|
| `int Days() const` | Retourne le nombre de jours dans l'instance `Duration` actuelle. |
| `DayTimeDuration DayTime() const` | Retourne le jour et la durée de temps dans l'instance `Duration` actuelle, exprimé en tant qu'un objet `DayTimeDuration` (voir altova::DayTimeDuration 1003 ). |
| `int Hours() const` | Retourne le nombre d'heures dans l'instance `Duration` actuelle. |
| `bool IsNegative() const` | Retourne Boléenne **true** si l'instance `Duration` actuelle est négative. |
| `bool IsPositive() const` | Retourne Boléenne **true** si l'instance `Duration` actuelle est positive. |
| `int Minutes() const` | Retourne le nombre de minutes dans l'instance `Duration` actuelle. |

| Nom | Description |
|---|---|
| `int Months() const` | Retourne le nombre de mois dans l'instance `Duration` actuelle. |
| `double Seconds() const` | Retourne le nombre de secondes dans l'instance `Duration` actuelle. |
| `YearMonthDuration YearMonth() const` | Retourne la durée d'année et de mois dans l'instance `Duration` actuelle, exprimée en tant qu'un objet `YearMonthDuration` (voir altova::YearMonthDuration 1003 ). |
| `int Years() const` | Retourne le nombre des années dans l'instance `Duration` actuelle. |

## Exemple

L'extrait de code suivant illustre la création d'un nouvel objet `Duration`, ainsi que la lecture de valeurs en provenant.

```
void ExampleDuration()
{
   // Create an empty Duration object
   altova::Duration empty_duration = altova::Duration();

   // Create a Duration object using an existing duration value
   altova::Duration duration1 = altova::Duration(empty_duration);

   // Create a YearMonth duration of six years and five months
   altova::YearMonthDuration yrduration = altova::YearMonthDuration(6, 5);

   // Create a DayTime duration of four days, three hours, two minutes, and one second
   altova::DayTimeDuration dtduration = altova::DayTimeDuration(4, 3, 2, 1);

   // Create a Duration object by combining the two previously created durations
   altova::Duration duration = altova::Duration(yrduration, dtduration);

   // Get the number of years in this Duration instance
   cout << "Years:  " << duration.Years() << endl;

   // Get the number of months in this Duration instance
   cout << "Months: " << duration.Months() << endl;

   // Get the number of days in this Duration instance
   cout << "Days:   " << duration.Days() << endl;

   // Get the number of hours in this Duration instance
   cout << "Hours:  " << duration.Hours() << endl;

   // Get the number of hours in this Duration instance
   cout << "Minutes: " << duration.Minutes() << endl;

   // Get the number of seconds in this Duration instance
   cout << "Seconds: " << duration.Seconds() << endl;
}
```

## 15.2.4.3 altova::DayTimeDuration

Cette classe vous permet de traiter des types de durée de schéma XML qui consistent en une partie jour et une partie heure.

### Constructeurs

| Nom | Description |
|---|---|
| `DayTimeDuration()` | Initialise une nouvelle instance de la classe `DayTimeDuration` dans une valeur vide. |
| `DayTimeDuration(int days, int hours, int minutes, double seconds)` | Initialise une nouvelle instance de la classe `DayTimeDuration` dans le nombre de jours, heures, minutes, secondes fournis en tant qu'argument. |
| `explicit DayTimeDuration(__int64 value)` | Initialise une nouvelle instance de la classe `DayTimeDuration` dans une durée qui consiste en autant de tics (intervalles de 100 nanosecondes) que fournis dans l'argument **value**. |

### Méthodes

| Nom | Description |
|---|---|
| `int Days() const` | Retourne le nombre de jours dans l'instance `DayTimeDuration` actuelle. |
| `int Hours() const` | Retourne le nombre d'heures dans l'instance `DayTimeDuration` actuelle. |
| `bool IsNegative() const` | Retourne Booléenne **true** si l'instance `DayTimeDuration` actuelle est négative. |
| `bool IsPositive() const` | Retourne Booléenne **true** si l'instance `DayTimeDuration` actuelle est positive. |
| `int Minutes() const` | Retourne le nombre de minutes dans l'instance `DayTimeDuration` actuelle. |
| `double Seconds() const` | Retourne le nombre de secondes dans l'instance `DayTimeDuration` actuelle. |
| `__int64 Value() const` | Retourne la valeur (en tics) de l'instance`DayTimeDuration` actuelle. |

## 15.2.4.4 altova::YearMonthDuration

Cette classe vous permet de traiter des types de durée de schéma XML qui consistent en une partie année et mois.

### Constructeurs

| Nom | Description |
|---|---|
| `YearMonthDuration()` | Initialise une nouvelle instance de la classe `YearMonthDuration` dans une valeur vide. |
| `YearMonthDuration(int years, int months)` | Initialise une nouvelle instance de la classe `YearMonthDuration` dans le nombre des années et des mois fournis dans les arguments **années** et **mois**. |
| `explicit YearMonthDuration(int value)` | Initialise une nouvelle instance de la classe `YearMonthDuration` dans une durée qui consiste en autant de tics (intervalles de 100 nanosecondes) comme fournir dans l'argument **value**. |

### Méthodes

| Nom | Description |
|---|---|
| `bool IsNegative() const` | Retourne Booléennen **true** si l'instance `YearMonthDuration` actuelle est négative. |
| `bool IsPositive() const` | Retourne Booléennen **true** si l'instance `YearMonthDuration` actuelle est positive. |
| `int Months() const` | Retourne le nombre de mois dans l'instance `YearMonthDuration` actuelle. |
| `int Value() const` | Retourne la valeur (en tics) de l'instance `YearMonthDuration` actuelle. |
| `int Years()` | Retourne le nombre d'années dans l'instance `YearMonthDuration` actuelle. |

## 15.2.4.5 altova::meta::Attribute

Cette classe vous permet d'accéder à des informations de schéma concernant des classes générées depuis des attributs. Veuillez noter que cette classe n'est pas destinée à fournir des informations dynamiques à propos des instances particulières d'un attribut dans un document XML. Au lieu, il vous permet d'obtenir des informations par le biais d'un programme concernant un attribut particulier défini dans le schéma XML.

### Méthodes

| Nom | Description |
|---|---|
| `SimpleType GetDataType()` | Retourne le type du contenu d'attribut. |
| `string_type GetLocalName()` | Retourne le nom local de l'attribut. |
| `string_type GetNamespaceURI()` | Retourne l'URI d'espace de noms de l'attribut. |

| Nom | Description |
|---|---|
| `bool IsRequired()` | Retourne true si l'attribut est requis. |

### Opérateurs

| Nom | Description |
|---|---|
| `bool operator()` | Retourne true si ce n'est pas l'attribut NULL. |
| `bool operator!()` | Retourne true s'il s'agit de l'attribut NULL. |

## 15.2.4.6 altova::meta::ComplexType

Cette classe vous permet d'accéder à des informations de schéma concernant des classes générées depuis des types complexes. Veuillez noter que cette classe n'est pas destinée à fournir des informations dynamiques à propos des instances particulières des types complexes dans un document XML. Au lieu, elle vous permet d'obtenir des informations par le biais d'un programme concernant un type complexes particulier défini dans le schéma XML.

### Méthodes

| Nom | Description |
|---|---|
| `Attribute FindAttribute(const char_type* localName, const char_type* namespaceURI)` | Trouve l'attribut avec le nom local spécifié et l'URI d'espace de noms |
| `Element FindElement(const char_type* localName, const char_type* namespaceURI)` | Trouve l'élément avec le nom local spécifié et l'URI d'espace de noms. |
| `std::vector<Attribute> GetAttributes()` | Retourne une liste de tous les attributs. |
| `ComplexType GetBaseType()` | Retourne le type de base de ce type. |
| `SimpleType GetContentType()` | Retourne le type simple du contenu. |
| `std::vector<Element> GetElements()` | Retourne une liste de tous les éléments. |
| `string_type GetLocalName()` | Retourne le nom local du type. |
| `string_type GetNamespaceURI()` | Retourne l'URI d'espace de noms du type. |

### Opérateurs

| Nom | Description |
|---|---|
| `bool operator()` | Retourne true si ce n'est pas le ComplexType NULL. |

| Nom | Description |
|---|---|
| `bool operator!()` | Retourne true s'il s'agit du ComplexType NULL. |

## 15.2.4.7 altova::meta::Element

Cette classe vous permet d'accéder à des informations concernant des classes générées depuis des éléments de schéma. Veuillez noter que cette classe n'est pas destinée à fournir des informations dynamiques à propos des instances particulières d'un élément dans un document XML. Au lieu, il vous permet d'obtenir des informations par le biais d'un programme concernant un élément particulier défini dans le schéma XML.

### Méthodes

| Nom | Description |
|---|---|
| `ComplexType GetDataType()` | Retourne le type de l'élément. Noter qu'il s'agit toujours d'un type de complexe même s'il est déclaré en tant que simple dans le schéma original. Utiliser GetContentType() de l'objet retourné pour obtenir le type de contenu simple. |
| `string_type GetLocalName()` | Retourne le nom local de l'élément. |
| `unsigned int GetMaxOccurs()` | Retourne la valeur maxOccurs définie dans le schéma. |
| `unsigned int GetMinOccurs()` | Retoune la valeur minOccurs définie dans le schéma. |
| `string_type GetNamespaceURI()` | Retourne l'URI d'espace de noms de l'élément . |

### Opérateurs

| Nom | Description |
|---|---|
| `bool operator()` | Retourne true si ce n'est pas l'élément NULL. |
| `bool operator!()` | Retourne true s'il s'agit de l'élément NULL. |

## 15.2.4.8 altova::meta::SimpleType

Cette classe vous permet d'accéder à des informations de schéma concernant des classes générées depuis des types simples. Veuillez noter que cette classe n'est pas destinée à fournir des informations dynamiques à propos des instances particulières des types simples dans un document XML. Au lieu, elle vous permet d'obtenir des informations par le biais d'un programme concernant un type simple particulier défini dans le schéma XML.

## Méthodes

| Nom | Description |
|---|---|
| `SimpleType GetBaseType()` | Retourne le type de base de ce type. |
| `std::vector<string_type> GetEnumerations()` | Retourne une liste de toutes les facettes d'énumération. |
| `unsigned int GetFractionDigits()` | Retourne la valeur de cette facette. |
| `unsigned int GetLength()` | Retourne la valeur de cette facette. |
| `string_type GetLocalName()` | Retourne le nom local du type. |
| `string_type GetMaxExclusive()` | Retourne la valeur de cette facette. |
| `string_type GetMaxInclusive()` | Retourne la valeur de cette facette. |
| `unsigned int GetMaxLength()` | Retourne la valeur de cette facette. |
| `string_type GetMinExclusive()` | Retourne la valeur de cette facette. |
| `string_type GetMinInclusive()` | Retourne la valeur de cette facette. |
| `unsigned int GetMinLength()` | Retourne la valeur de cette facette. |
| `string_type GetNamespaceURI()` | Retourne l'URI d'espace de noms du type. |
| `std::vector<string_type> GetPatterns()` | Retourne une liste de toutes les facettes de motif. |
| `unsigned int GetTotalDigits()` | Retourne la valeur de cette facette. |
| `WhitespaceType GetWhitespace()` | Retourne la valeur de la facette d'espace blanc, qui est un de :<br>• Whitespace_Unknown<br>• Whitespace_Preserve<br>• Whitespace_Replace<br>• Whitespace_Collapse |

## Opérateurs

| Nom | Description |
|---|---|
| `bool operator()` | Retourne true si ce n'est pas de SimpleType NULL. |
| `bool operator!()` | Retourne true s'il s'agit du SimpleType NULL. |

## 15.2.4.9 [YourSchema]::[CDoc]

Lorsque le code est généré depuis un schéma XML, le code généré fournit une classe de document avec le même nom que le schéma. Cette classe contient tous les éléments root possibles en tant que membres, ainsi que les méthodes suivantes. Veuillez noter que, dans les noms de méthode ci-dessous, "CDoc" représente le nom de la classe de document générée elle-même.

### Méthodes

| Nom | Description |
|---|---|
| `static CDoc CreateDocument()` | Crée un nouveau document XML vide. Doit être publié en utilisant DestroyDocument(). |
| `static void DeclareAllNamespacesFromSchema(ElementType& node)` | Déclare tous les espaces de noms depuis le Schéma XML dans l'élément fournit en dans qu'argument (généralement, l'élément racine XML). L'appel de cette méthode est utile si votre schéma a plusieurs déclarations d'espace de noms, chacune mappée dans un préfixe, et que vous souhaitez les déclarer tous dans l'élément fourni en tant qu'argument. |
| `void DestroyDocument()` | Détruit un document. Toutes les références au document et à ses nœuds sont invalidées. Cela doit être appelé lorsque vous avez terminé de travailler avec un document. |
| `static CDoc LoadFromBinary(const std:vector<unsigned char>& xml)` | Charge un document XML depuis un byte array. |
| `static CDoc LoadFromFile(const string_type& fileName)` | Charge un document XML depuis un fichier. |
| `static CDoc LoadFromString(const string_type& xml)` | Charge un document XML depuis un string. |
| `std::vector<unsigned char> SaveToBinary(bool prettyPrint)` | Enregistre un document XML sous un byte array. Lorsque défini sur true, l'argument `prettyPrint` reformate le document XML pour une meilleure lecture. |
| `std::vector<unsigned char> SaveToBinary(bool prettyPrint, const string_type & encoding)` | Enregistre un document XML sous un byte array, avec un formatage "pretty-print" optionnel, avec l'encodage spécifié. |
| `std::vector<unsigned char> SaveToBinary(bool prettyPrint, const string_type & encoding, bool bBigEndian, bool bBOM)` | Enregistre un document XML sous un byte array, avec un formatage "pretty-print" optionnel, avec l'encodage spécifié. La marque Byte order et Unicode byte-order peuvent être spécifiées pour les encodages Unicode. |
| `void SaveToFile(const string_type & fileName, bool prettyPrint)` | Enregistre un document XML sous un fichier, avec un formatage "pretty-print" optionnel. |
| `void SaveToFile(const string_type & fileName, bool omitXmlDecl)` | Enregistre un document XML sous un fichier. Si l'argument `omitXmlDecl` est défini pour être true, la déclaration XML ne sera pas rédigée. |

| Nom | Description |
|---|---|
| `void SaveToFile(const string_type & fileName, bool omitXmlDecl, const string_type & encoding)` | Enregistre un document XML sous un fichier avec l'encodage spécifié. Si l'argument `omitXmlDecl` est défini pour être true, la déclaration XML ne sera pas rédigée. |
| `void SaveToFile(const string_type & fileName, bool prettyPrint, bool omitXmlDecl, const string_type & encoding, bool bBigEndian, bool bBOM)` | Enregistre un document XML sous un fichier, avec un formatage "pretty-print" optionnel, avec l'encodage spécifié. La marque Byte order et Unicode byte-order peuvent être spécifiées pour les encodages Unicode. |
| `void SaveToFile(const string_type & fileName, bool prettyPrint, bool omitXmlDecl, const string_type & encoding, bool bBigEndian, bool bBOM, const string_type & lineend)` | Enregistre un document XML sous un fichier, avec un formatage "pretty-print" optionnel, avec l'encodage spécifié et la fin de ligne spécifiée. La marque Byte order et Unicode byte-order peuvent être spécifiées pour les encodages Unicode.<br><br>Cette méthode est uniquement disponible si vous générez le code pour la bibliothèque Xerces3 XML (voir Génération[1089]). |
| `void SaveToFile(const string_type& fileName, bool prettyPrint, bool omitXmlDecl, const string_type & encoding, const string_type & lineend)` | Enregistre un document XML sous un fichier, avec un formatage "pretty-print" optionnel, avec l'encodage spécifié et la fin de ligne spécifiée.<br><br>Cette méthode est uniquement disponible si vous générez le code pour la bibliothèque Xerces3 XML (voir Génération[1089]). |
| `void SaveToFile(const string_type & fileName, bool prettyPrint, const string_type & encoding)` | Enregistre un document XML sous un fichier, avec un formatage "pretty-print" optionnel, avec l'encodage spécifié. |
| `void SaveToFile(const string_type& fileName, bool prettyPrint, const string_type & encoding, bool bBigEndian, bool bBOM)` | Enregistre un document XML sous un fichier, avec un formatage "pretty-print" optionnel, avec l'encodage spécifié. La marque Byte order et Unicode byte-order peuvent être spécifiées pour les encodages Unicode. |
| `void SaveToFile(const string_type& fileName, bool prettyPrint, const string_type & encoding, bool bBigEndian, bool bBOM, const string_type & lineend)` | Enregistre un document XML sous un fichier, avec l'encodage spécifié et la fin de ligne spécifiée. La marque Byte order et Unicode byte-order peuvent être spécifiées pour les encodages Unicode.<br><br>Cette méthode est uniquement disponible si vous générez le code pour la bibliothèque Xerces3 XML (voir Génération[1089]). |
| `void SaveToFile(const string_type& fileName, bool prettyPrint, const string_type & encoding, const string_type & lineend)` | Enregistre un document XML sous un fichier, avec un formatage "pretty-print" optionnel, avec l'encodage spécifié et la fin de ligne spécifiée.<br><br>Cette méthode est uniquement disponible si vous générez le code pour la bibliothèque Xerces3 XML (voir Génération[1089]). |
| `string_type SaveToString(bool prettyPrint)` | Enregistre un document XML sous un string, avec un formatage "pretty-print" optionnel. |

| Nom | Description |
|---|---|
| `string_type SaveToString(bool prettyPrint, bool omitXmlDecl)` | Enregistre un document XML sous un string, avec un formatage "pretty-print" optionnel. Si l'argument `omitXmlDecl` est défini pour être true, la déclaration XML ne sera pas rédigée. |
| `void SetDTDLocation(const string_type & dtdLocation)` | Ajoute une déclaration DOCTYPE avec l'ID de système spécifiée. Un élément root doit déjà exister. Cette méthode n'est pas prise en charge pour MSXML, puisqu'il n'est pas possible d'ajouter une déclaration DOCTYPE dans un document dans la mémoire. |
| `void SetSchemaLocation(const string_type & schemaLocation)` | Ajoute un attribut xsi:schemaLocation ou xsi:noNamespaceSchemaLocation à l'élément root. Un élément root doit déjà exister. |

## 15.2.4.10 [YourSchema]::[ElementType]

Cette classe propose des méthodes pour manipuler des éléments XML depuis votre schéma. Des méthodes de cette classe peuvent être appelées sur des éléments, pas sur le document XML lui-même. Noter qu'il n'est pas nécessaire d'instancier la classe directement pour appeler des méthodes de cette classe. Tout élément créé en utilisant les méthodes **`append()`** ou **`appendWithPrefix()`** est de type `[ElementType]`.

### Méthodes

| Nom | Description |
|---|---|
| `void DeclareNamespace(const string_type prefix, const string_type nsURI)` | Cette méthode prend deux arguments qui sont tous les deux de type string : le préfixe et l'URI d'espace de noms que vous souhaitez utiliser. Le préfixe fournit en tant qu'argument sera mappé dans la valeur URI d'espace de noms fournie en tant qu'argument. Si le préfixe fourni en tant qu'argument est vide, la méthode crée ou contourne la déclaration d'espace de noms par défaut dans l'élément.<br><br>Par exemple, assumons que le document XML a un élément XML appelé "purchase". Si vous appelez<br><br>`purchase.DeclareNamespace(_T("ord"), _T("http://OrderTypes"));`<br><br>Alors le document XML devient<br><br>`<purchase xmlns:ord="http://OrderTypes" />`<br><br>Un autre exemple ; si vous appelez : |

| Nom | Description |
|---|---|
| | `purchase.DeclareNamespace(_T(""), _T("http://OrderTypes"));`<br><br>Alors le document XML devient<br><br>`<purchase xmlns="http://OrderTypes" />`<br><br>Note : l'espace de noms déclaré est utilisé lorsque les éléments ou les attributs enfants ultérieurs, conformément aux règles suivantes :<br><br>1. si l'espace de noms enfant est le défaut, utiliser le préfixe vide.<br>2. si l'espace de noms enfant est égal à l'espace de noms parent, utiliser le préfixe parent.<br>3. Sinon, chercher le préfixe le plus proche en commençant du parent vers le haut, en utilisant l'algorithme lookup décrit dans la section "B.2: Namespace Prefix Lookup" at https://www.w3.org/TR/2002/WD-DOM-Level-3-Core-20021022/namespaces-algorithms.html.<br>4. Si aucun préfixe n'a été trouvé pour l'espace de noms élément, utiliser un préfixe vide. |

## 15.2.4.11 [YourSchema]::MemberAttribute

Lorsque le code est généré depuis un schéma XML, une classe telle que celle-ci est créée pour chaque attribut de membre d'un type.

### Méthodes

| Nom | Description |
|---|---|
| `bool exists()` | Retourne true si l'attribut existe. |
| `int GetEnumerationValue()` | Généré pour des types d'énumération uniquement. Retourne une des constantes générée pour les valeurs possibles, ou "Invalid" si la valeur ne correspond pas à une des valeurs énumérées dans le schéma. |
| `altova::meta::Attribute info()` | Retourne un objet pour l'information de schéma de requête (voir `altova::meta::Attribute`). |
| `void remove()` | Supprime l'attribut depuis son élément parent. |
| `void SetEnumerationValue(int)` | Généré pour des types d'énumération uniquement. Passer une des constantes générées pour les valeurs possibles dans cette |

| Nom | Description |
|---|---|
| | méthode pour définir la valeur. |

## 15.2.4.12 [YourSchema]::MemberElement

Lorsque le code est généré depuis un schéma XML, une classe telle que celle-ci est créée pour chaque élément de membre d'un type. Dans les descriptions ci-dessous, "MemberType" représente le nom de l'élément de membre lui-même.

### Méthodes

| Nom | Description |
|---|---|
| `Iterator<MemberType> all()` | Retourne un objet pour itérer des instances de l'élément de membre. |
| `MemberType append()` | Crée un nouvel élément et l'ajoute à son parent. |
| `MemberType appendWithPrefix(string_type prefix)` | Crée un nouvel élément ayant le préfixe fournit en tant qu'argument, et l'appose à son parent. Pour consulter un exemple, voir Exemple : Purchase Order 990. |
| `unsigned int count()` | Retourne le décompte des éléments. |
| `int GetEnumerationValue()` | Généré pour des types d'énumération uniquement. Retourne une des constantes générée pour les valeurs possibles, ou **Invalid** si la valeur ne correspond pas à une des valeurs énumérées dans le schéma. |
| `bool exists()` | Retourne true si au moins un élément existe. |
| `MemberType first()` | Retourne la première instance de l'élément de membre. |
| `MemberType operator[](unsigned int index)` | Retourne l'élément de membre spécifié par l'index. |
| `altova::meta::Element info()` | Retourne un objet pour l'information de schéma de requête (voir `altova::meta::Element` 1006). |
| `MemberType last()` | Retourne la dernière instance de l'élément de membre. |
| `void remove()` | Supprime toutes les occurrences de l'élément depuis son parent. |
| `void removeAT(unsigned int index)` | Supprime l'occurrence de l'élément spécifié par l'index. |
| `void SetEnumerationValue(int)` | Généré pour des types d'énumération uniquement. Passer une des constantes générées pour les valeurs possibles dans cette méthode pour définir la valeur. |

## 15.2.5 Référence aux classes générées (C#)

Ce chapitre contient une description des classes C# générées avec MapForce epuis un schéma XML ou DTD (voir Générer du code depuis des Schémas XML ou DTDs 953 ). Vous pouvez intégrer ces classes dans votre code pour lire, modifier et écrire des documentation XML.

**Note:** Le code généré peut inclure d'autres classes de support, qui ne sont pas recensées ici et qui sont sujettes à des modifications.

### 15.2.5.1 Altova.Types.DateTime

Cette classe vous permet de traiter des attributs XML ou des éléments qui ont des types de date et d'heure, comme `xs:dateTime`.

#### Constructeurs

| | Nom | Description |
|---|---|---|
| | `DateTime(DateTime obj)` | Initialise une nouvelle instance de la classe `DateTime` dans l'objet `DateTime` fourni en tant qu'argument. |
| | `DateTime(System.DateTime newvalue)` | Initialise une nouvelle instance de la classe `DateTime` dans l'objet `System.DateTime` fourni en tant qu'argument. |
| | `DateTime(int year, int month, int day, int hour, int minute, double second, int offsetTZ)` | Initialise une nouvelle instance de la classe `DateTime` à l'offset d'année, mois, jour, heure, minute, seconde et fuseau horaire fournis en tant qu'arguments. |
| | `DateTime(int year, int month, int day, int hour, int minute, double second)` | Initialise une nouvelle instance de la classe `DateTime` à l'année, mois, jour, heure, minute et seconde fournis en tant qu'arguments. |
| | `DateTime(int year, int month, int day)` | Initialise une nouvelle instance de la classe `DateTime` à l'année, mois et jour fournis en tant qu'argument. |

#### Propriétés

| | Nom | Description |
|---|---|---|
| | `bool HasTimezone` | Obtient une valeur Booléenne qui indique si `DateTime` a un fuseau horaire. |
| | `static DateTime Now` | Obtient un objet `DateTime` qui est défini sur la date et l'heure actuelles sur cet ordinateur. |
| | `short TimezoneOffset` | Obtient ou définit l'offset de fuseau horaire, en minutes de l'objet `DateTime`. |

| | Nom | Description |
|---|---|---|
| | `System.DateTime Value` | Obtient ou définit l'offset de l'objet `DateTime` en tant qu'une valeur `System.DateTime`. |

## Méthodes

| | Nom | Description |
|---|---|---|
| | `int CompareTo(object obj)` | La classe `DateTime` met en place l'interface `IComparable`. Cette méthode compare l'instance actuelle de `DateTime` avec un autre objet et retourne un entier qui indique si l'instance actuelle précèdes, suit, ou se produit dans la même position dans l'ordre de tri que l'autre objet. Voir aussi https://msdn.microsoft.com/en-us/library/system.icomparable.compareto(v=vs.110).aspx |
| | `override bool Equals(object obj)` | Retourne **true** si l'objet spécifié est égal à l'objet actuel ; **false** sinon. |
| | `System.DateTime GetDateTime(bool correctTZ)` | Retourne un objet `System.DateTime` depuis l'instance `Altova.Types.DateTime`. L'argument booléen `correctTZ` spécifie si le temps de l'objet retourné doit être ajusté conformément au fuseau horaire de l'instance actuel `Altova.Types.DateTime`. |
| | `override int GetHashCode()` | Retourne le code hash de l'instance actuelle. |
| | `int GetWeekOfMonth()` | Retourne le nombre de semaines dans le mois sous forme d'un entier. |
| | `static DateTime Parse( string s )` | Crée un objet `DateTime` depuis le string fournit en tant qu'argument. Par exemple, les valeurs de string d'échantillon suivantes sont converties avec succès dans un objet `DateTime` :<br><br>`2015-01-01T23:23:23`<br>`01/01/2015`<br>`2015-11`<br>`23:23:23`<br><br>Une exception est soulevée si le string ne peut pas être converti en un objet `DateTime`.<br><br>Veuillez noter que cette méthode est statique et qu'elle peut uniquement être appelée dans la classe `Altova.Types.DateTime` elle-même, pas dans une instance de la classe. |
| | `static DateTime Parse(string s, DateTimeFormat format)` | Crée un objet `DateTime` depuis un string, en utilisant le format fourni en tant qu'argument. Pour la liste des formats possibles, voir Altova.Types.DateTimeFormat 1016.<br><br>Une exception est soulevée si le string ne peut pas être converti en un objet `DateTime`. |

| | Nom | Description |
|---|---|---|
| | | Veuillez noter que cette méthode est statique et qu'elle peut uniquement être appelée dans la classe `Altova.Types.DateTime` elle-même, pas dans une instance de la classe. |
|  | `override string ToString()` | Convertit l'objet `DateTime` en un string. |
|  | `string ToString(DateTimeFormat format)` | Convertit l'objet `DateTime` en un string, utiliser le format fournit en tant qu'argument. Pour la liste des formats possibles, voir Altova.Types.DateTimeFormat[1016]. |

## Opérateurs

| Nom | Description |
|---|---|
| `!=` | Détermine si `DateTime` a n'est pas égal à `DateTime` b. |
| `<` | Détermine si `DateTime` a est inférieur à `DateTime` b. |
| `<=` | Détermine si `DateTime` a est inférieur ou égal à `DateTime` b. |
| `==` | Détermine si `DateTime` a est égal à `DateTime` b. |
| `>` | Détermine si `DateTime` a est supérieur à `DateTime` b. |
| `>=` | Détermine si `DateTime` a est supérieur ou égal à `DateTime` b. |

## Exemples

Avant d'utiliser les extraits de code suivants dans votre programme, s'assurer que les types Altova sont importés :

```
utiliser Altova.Types;
```

L'extrait de code suivant illustre plusieurs moyens de créer des objets `DateTime` :

```
protected static void DateTimeExample1()
{
   // Create a DateTime object from the current system time
   Altova.Types.DateTime dt = new Altova.Types.DateTime(System.DateTime.Now);
   Console.WriteLine("The current time is: " + dt.ToString());

   // Create an Altova DateTime object from parts (no timezone)
   Altova.Types.DateTime dt1 = new Altova.Types.DateTime(2015, 10, 12, 10, 50, 33);
   Console.WriteLine("My custom time is : " + dt1.ToString());

   // Create an Altova DateTime object from parts (with UTC+60 minutes timezone)
   Altova.Types.DateTime dt2 = new Altova.Types.DateTime(2015, 10, 12, 10, 50, 33, 60);
```

```
   Console.WriteLine("My custom time with timezone is : " + dt2.ToString());

   // Create an Altova DateTime object by parsing a string
   Altova.Types.DateTime dt3 = Altova.Types.DateTime.Parse("2015-01-01T23:23:23");
   Console.WriteLine("Time created from string: " + dt3.ToString());

   // Create an Altova DateTime object by parsing a string formatted as schema date
   Altova.Types.DateTime dt4 = Altova.Types.DateTime.Parse("2015-01-01",
DateTimeFormat.W3_date);
   Console.WriteLine("Time created from string formatted as schema date: " +
dt4.ToString());
}
```

The following code listing illustrates various ways to format `DateTime` objects:

```
protected static void DateTimeExample2()
{
   // Create a DateTime object from the current system time
   Altova.Types.DateTime dt = new Altova.Types.DateTime(System.DateTime.Now);

   // Output the unformatted DateTime
   Console.WriteLine("Unformatted time: " + dt.ToString());

   // Output this DateTime formatted using various formats
   Console.WriteLine("S_DateTime:       " + dt.ToString(DateTimeFormat.S_DateTime));
   Console.WriteLine("S_Days:           " + dt.ToString(DateTimeFormat.S_Days));
   Console.WriteLine("S_Seconds:        " + dt.ToString(DateTimeFormat.S_Seconds));
   Console.WriteLine("W3_date:          " + dt.ToString(DateTimeFormat.W3_date));
   Console.WriteLine("W3_dateTime:      " + dt.ToString(DateTimeFormat.W3_dateTime));
   Console.WriteLine("W3_gDay:          " + dt.ToString(DateTimeFormat.W3_gDay));
   Console.WriteLine("W3_gMonth:        " + dt.ToString(DateTimeFormat.W3_gMonth));
   Console.WriteLine("W3_gMonthDay:     " + dt.ToString(DateTimeFormat.W3_gMonthDay));
   Console.WriteLine("W3_gYear:         " + dt.ToString(DateTimeFormat.W3_gYear));
   Console.WriteLine("W3_gYearMonth:    " + dt.ToString(DateTimeFormat.W3_gYearMonth));
   Console.WriteLine("W3_time:          " + dt.ToString(DateTimeFormat.W3_time));
}
```

## 15.2.5.2 Altova.Types.DateTimeFormat

Le type `DateTimeFormat` enum comporte les valeurs constantes suivantes :

| Valeur | Description | Exemple |
|---|---|---|
| **S_DateTime** | Formate la valeur en tant que dateTime standard, avec une précision d'un dix millionième de seconde, y compris le fuseau horaire. | `2015-11-12 12:19:03.9019132+01:00` |
| **S_Days** | Formate la valeur en tant que nombre de jours passés depuis l'époque UNIX. | `735913,631897345108796296 2963` |

| Valeur | Description | Exemple |
|---|---|---|
| **S_Seconds** | Formate la valeur en tant que le nombre de seconde écoulées depuis l'époque UNIX, avec une précision d'un dix millionième d'une seconde. | `63582937678,0769062` |
| **W3_date** | Formate la valeur en tant que date de schéma. | `12/11/2015` |
| **W3_dateTime** | Formate la valeur en tant que dateTime de schéma. | `2015-11-12T15:12:14.5194251` |
| **W3_gDay** | Formate la valeur en tant que gDay de schéma. | `---12`<br>(en partant du principe que la date est le 12 du mois) |
| **W3_gMonth** | Formate la valeur en tant que gMonth de schéma. | `--11`<br>(en partant du principe que le mois est novembre) |
| **W3_gMonthDay** | Formate la valeur en tant que gMonthDay de schéma. | `--11-12`<br>(en partant du principe que la date est le 12 novembre) |
| **W3_gYear** | Formate la valeur en tant que gYear de schéma. | `2015`<br>(en partant du principe que l'année est 2015) |
| **W3_gYearMonth** | Formate la valeur en tant que gYearMonth de schéma. | `2015-11`<br>(en partant du principe que l'année est 2015 et que le mois est novembre) |
| **W3_time** | Formate la valeur en tant qu'heure de schéma, avec une précision d'un dix millionième d'une seconde. | `15:19:07.5582719` |

## 15.2.5.3 Altova.Types.Duration

Cette classe vous permet de traiter des attributs XML ou des éléments de type `xs: duration` .

### Constructeurs

| | Nom | Description |
|---|---|---|
|  | `Duration(Duration obj)` | Initialise une nouvelle instance de la classe `Duration` dans l'objet `Duration` fournie en tant qu'argument. |
|  | `Duration(Duration newvalue)` | Initialise une nouvelle instance de la classe `Duration` dans l'objet `System.TimeSpan` fournie en tant qu'argument. |

| | Nom | Description |
|---|---|---|
| | `Duration(long ticks)` | Initialise une nouvelle instance de la classe `Duration` dans la le nombre de tics fournie en tant qu'argument. |
| | `Duration(int newyears, int newmonths, int days, int hours, int minutes, int seconds, double partseconds, bool bnegative)` | Initialise une nouvelle instance de la classe `Duration` dans une durée construite depuis des parties fournies en tant qu'argument. |

## Propriétés

| | Nom | Description |
|---|---|---|
| | `int Months` | Obtient ou définit le nombre des mois dans l'instance actuelle de `Duration`. |
| | `System.TimeSpan Value` | Obtient ou définit la valeur (en tant que `System.TimeSpan`) de l'instance actuelle de `Duration`. |
| | `int Years` | Retourne le nombre des années dans l'instance actuelle de `Duration`. |

## Méthodes

| | Nom | Description |
|---|---|---|
| | `override bool Equals(object other)` | Retourne **true** si l'objet spécifié est égal à l'objet actuel ; **false** sinon. |
| | `override int GetHashCode()` | Retourne le code hash de l'instance actuelle. |
| | `bool IsNegative()` | Retourne **true** si l'instance actuelle de `Duration` représente une durée négative. |
| | `static Duration Parse( string s, ParseType pt )` | Retourne un objet `Altova.Types.Duration` parsé depuis le string fourni en tant qu'argument, en utilisant le type de parsage fourni en tant qu'argument. Valeurs de type de parsage valides :<br>**DURATION** Parse la durée en partant du principe que la partie de l'année, du mois, du jour et de la durée de temps existe.<br>**YEARMONTH** Parse la durée en partant du principe que seules les parties d'année et de mois existent.<br>**DAYTIME** Parse la durée en partant du principe que la partie du jour et d'heure existent.<br>Veuillez noter que cette méthode est statique et peut uniquement être appelée dans la classe elle-même, pas dans une instance de la classe. |

| | Nom | Description |
|---|---|---|
| | `override string ToString()` | Convertit l'instance actuelle `Duration` en un string. Par exemple, une durée de 3 heures, 4 minutes et 5 secondes peut être convertie en "PT3H4M5S". |
| | `string ToYearMonthString()` | Convertit l'instance actuelle `Duration` en un string, en utilisant le type de parsage "Year and Month". |

## Opérateurs

| Nom | Description |
|---|---|
| `!=` | Détermine si `Duration` a n'est pas égal à `Duration` b. |
| `==` | Détermine si `Duration` a est égal à `Duration` b. |

## Exemples

Avant d'utiliser les extraits de code suivants dans votre programme, s'assurer que les types Altova sont importés :

```
using Altova.Types;
```

L'extrait de code suivant illustre plusieurs moyens de créer des objets `Duration` :

```
protected static void DurationExample1()
{
   // Create a new time span of 3 hours, 4 minutes, and 5 seconds
   System.TimeSpan ts = new TimeSpan(3, 4, 5);
   // Create a Duration from the time span
   Duration dr = new Duration(ts);
   // The output is: PT3H4M5S
   Console.WriteLine("Duration created from TimeSpan: " + dr.ToString());

   // Create a negative Altova.Types.Duration from 6 years, 5 months, 4 days, 3 hours,
       // 2 minutes, 1 second, and .33 of a second
   Duration dr1 = new Duration(6, 5, 4, 3, 2, 1, .33, true);
   // The output is: -P6Y5M4DT3H2M1.33S
   Console.WriteLine("Duration created from parts: " + dr1.ToString());

   // Create a Duration from a string using the DAYTIME parse type
   Duration dr2 = Altova.Types.Duration.Parse("-P4DT3H2M1S", Duration.ParseType.DAYTIME);
   // The output is -P4DT3H2M1S
   Console.WriteLine("Duration created from string: " + dr2.ToString());

   // Create a duration from ticks
   Duration dr3 = new Duration(System.DateTime.UtcNow.Ticks);
   // Output the result
```

```
   Console.WriteLine("Duration created from ticks: " + dr3.ToString());
}
```

L'extrait de code suivant illustre comment obtenir des valeurs depuis des objets `Duration` :

```
protected static void DurationExample2()
{
   // Create a negative Altova.Types.Duration from 6 years, 5 months, 4 days, 3 hours,
       // 2 minutes, 1 second, and .33 of a second
   Duration dr = new Duration(6, 5, 4, 3, 2, 1, .33, true);
   // The output is: -P6Y5M4DT3H2M1.33S
   Console.WriteLine("The complete duration is: " + dr.ToString());

   // Get only the year and month part as string
   string dr1 = dr.ToYearMonthString();
   Console.WriteLine("The YEARMONTH part is: " + dr1);

   // Get the number of years in duration
   Console.WriteLine("Years: " + dr.Years);

   // Get the number of months in duration
   Console.WriteLine("Months: " + dr.Months);
}
```

## 15.2.5.4 Altova.Xml.Meta.Attribute

Cette classe vous permet d'accéder à des informations de schéma concernant des classes générées depuis des attributs. Veuillez noter que cette classe n'est pas destinée à fournir des informations dynamiques à propos des instances particulières d'un attribut dans un document XML. Au lieu, il vous permet d'obtenir des informations par le biais d'un programme concernant un attribut particulier défini dans le schéma XML.

### Propriétés

| | Nom | Description |
|---|---|---|
| | `SimpleType DataType` | Retourne le type du contenu d'attribut. |
| | `string LocalName` | Retourne le nom local de l'attribut. |
| | `string NamespaceURI` | Retourne l'URI d'espace de noms de l'attribut. |
| | `XmlQualifiedName QualifiedName` | Retourne le nom qualifié de l'attribut. |
| | `bool Required()` | Retourne true si l'attribut est requis. |

## 15.2.5.5 Altova.Xml.Meta.ComplexType

Cette classe vous permet d'accéder à des informations de schéma concernant des classes générées depuis des types complexes. Veuillez noter que cette classe n'est pas destinée à fournir des informations dynamiques à propos des instances particulières des types complexes dans un document XML. Au lieu, elle vous permet d'obtenir des informations par le biais d'un programme concernant un type complexes particulier défini dans le schéma XML.

### Propriétés

| | Nom | Description |
|---|---|---|
| | `Attribute[] Attributes` | Retourne une liste de tous les attributs. |
| | `ComplexType BaseType` | Retourne le type de base de ce type ou null si aucun type de base type n'existe. |
| | `SimpleType ContentType` | Retourne le type simple du contenu. |
| | `Element[] Elements` | Retourne une liste de tous les éléments. |
| | `string LocalName` | Retourne le nom local du type. |
| | `string NamespaceURI` | Retourne l'URI d'espace de noms du type. |
| | `XmlQualifiedName QualifiedName` | Retourne le nom qualifié de ce type. |

### Méthodes

| | Nom | Description |
|---|---|---|
| | `ComplexType BaseType` | Retourne le type de base de ce type. |
| | `bool Equals(obj)` | Contrôle si deux objets d'info se réfèrent au même type, basé dans la comparaison du nom qualifié. Retourne true si le type a le même nom qualifié. |
| | `Attribute FindAttribute(string localName, string namespaceURI)` | Trouve l'attribut avec le nom local spécifié et l'URI d'espace de noms |
| | `Element FindElement(string localName, string namespaceURI)` | Trouve l'élément avec le nom local spécifié et l'URI d'espace de noms. |

## 15.2.5.6 Altova.Xml.Meta.Element

Cette classe vous permet d'accéder à des informations concernant des classes générées depuis des éléments de schéma. Veuillez noter que cette classe n'est pas destinée à fournir des informations dynamiques à propos

des instances particulières d'un élément dans un document XML. Au lieu, il vous permet d'obtenir des informations par le biais d'un programme concernant un élément particulier défini dans le schéma XML.

### Propriétés

| | Nom | Description |
|---|---|---|
| | `ComplexType DataType` | Retourne le type de l'élément. Noter qu'il s'agit toujours d'un type de complexe même s'il est déclaré en tant que simple dans le schéma original. Utiliser la propriété `ContentType` de l'objet retourné pour obtenir le type de contenu simple. |
| | **`string`** `LocalName` | Retourne le nom local de l'élément. |
| | **`int`** `MaxOccurs` | Retourne la valeur `maxOccurs` définie dans le schéma. |
| | **`int`** `MinOccurs` | Retourne la valeur `minOccurs` définie dans le schéma. |
| | **`string`** `NamespaceURI` | Retourne l'URI d'espace de noms de l'élément . |
| | `XmlQualifiedName QualifiedName` | Retourne le nom qualifié de l'élément. |

## 15.2.5.7 Altova.Xml.Meta.SimpleType

Cette classe vous permet d'accéder à des informations de schéma concernant des classes générées depuis des types simples. Veuillez noter que cette classe n'est pas destinée à fournir des informations dynamiques à propos des instances particulières des types simples dans un document XML. Au lieu, elle vous permet d'obtenir des informations par le biais d'un programme concernant un type simple particulier défini dans le schéma XML.

### Propriétés

| | Nom | Description |
|---|---|---|
| | `SimpleType BaseType` | Retourne le type de base de ce type. |
| | **`string`**`[] Enumerations` | Retourne une liste de toutes les facettes d'énumération. |
| | **`int`** `FractionDigits` | Retourne la valeur de cette facette. |
| | **`int`** `Length` | Retourne la valeur de cette facette. |
| | **`string`** `LocalName` | Retourne le nom local du type. |
| | **`string`** `MaxExclusive` | Retourne la valeur de cette facette. |
| | **`string`** `MaxInclusive` | Retourne la valeur de cette facette. |
| | **`int`** `MaxLength` | Retourne la valeur de cette facette. |
| | **`string`** `MinExclusive` | Retourne la valeur de cette facette. |

| | Nom | Description |
|---|---|---|
| | **string** MinInclusive | Retourne la valeur de cette facette. |
| | **int** MinLength | Retourne la valeur de cette facette. |
| | **string** NamespaceURI | Retourne l'URI d'espace de noms du type. |
| | **string**[] Patterns | Retourne les facettes de motif, ou null si aucun motif n'est spécifié. |
| | XmlQualifiedName QualifiedName | Retourne le nom qualifié de ce type. |
| | **int** TotalDigits | Retourne la valeur de cette facette. |
| | WhitespaceType Whitespace | Retourne la facette de normalisation d'espace blanc. |

## 15.2.5.8 [YourSchema].[Doc]

Lorsque le code est généré depuis un schéma XML, le code généré fournit une classe de document avec le même nom que le schéma. Cette classe contient tous les éléments root possibles en tant que membres, ainsi que les membres recensés ci-dessous. Veuillez noter que, dans les noms de méthode ci-dessous, "Doc" représente le nom de la classe de document générée elle-même.

### Méthodes

| | Nom | Description |
|---|---|---|
| | **static** Doc CreateDocument() | Crée un nouveau document XML vide. |
| | **static** Doc CreateDocument(**string** encoding) | Crée un nouveau document XML vide, avec l'encodage de type "encoding". |
| | **static** void DeclareAllNamespacesFromSchema(nœud **Altova.Xml.TypeBase** node) | Déclare tous les espaces de noms depuis le Schéma XML dans l'élément fournit en dans qu'argument (généralement, l'élément racine XML). L'appel de cette méthode est utile si votre schéma a plusieurs déclarations d'espace de noms, chacune mappée dans un préfixe, et que vous souhaitez les déclarer tous dans l'élément fourni en tant qu'argument. |
| | **static** Doc LoadFromBinary(**byte**[] binary) | Charge un document XML depuis un byte array. |
| | **static** Doc LoadFromFile(**string** filename) | Charge un document XML depuis un fichier. |
| | **static** Doc LoadFromString(**string** xmlstring) | Charge un document XML depuis un string. |

| | Nom | Description |
|---|---|---|
| | `byte[] SaveToBinary(bool prettyPrint)` | Enregistre un document XML sous un de byte, avec un formatage "pretty-print" optionnel. |
| | `byte[] SaveToBinary(bool prettyPrint, string encoding)` | Enregistre un document XML sous un byte array, avec un formatage "pretty-print" optionnel, avec l'encodage spécifié. |
| | `byte[] SaveToBinary(bool prettyPrint, string encoding, bool bBigEndian, bool bBOM)` | Enregistre un document XML sous un byte array, avec un formatage "pretty-print" optionnel, avec l'encodage spécifié, l'ordre de byte et BOM (Byte Order Mark). |
| | `void SaveToFile(string fileName, bool prettyPrint)` | Enregistre un document XML sous un fichier, avec un formatage "pretty-print" optionnel. |
| | `void SaveToFile(string fileName, bool prettyPrint, bool omitXmlDecl)` | Enregistre un document XML sous un fichier, avec un formatage "pretty-print" optionnel. Si `omitXmlDecl` est true, la déclaration XML ne sera pas rédigée. |
| | `void SaveToFile(string fileName, bool prettyPrint, bool omitXmlDecl, string encoding)` | Enregistre un document XML sous un fichier, avec un formatage "pretty-print" optionnel, avec l'encodage spécifié. Si `omitXmlDecl` est true, la déclaration XML ne sera pas rédigée. |
| | `void SaveToFile(string fileName, bool prettyPrint, string encoding, string lineend)` | Enregistre un document XML sous un fichier, avec un formatage "pretty-print" optionnel, avec l'encodage spécifié et le(s) caractère(s) de fin de ligne. |
| | `void SaveToFile(string fileName, bool prettyPrint, bool omitXmlDecl, string encoding, string lineend)` | Enregistre un document XML sous un fichier, avec un formatage "pretty-print" optionnel, avec l'encodage spécifié et le(s) caractère(s) de fin de ligne. Si `omitXmlDecl` est true, la déclaration XML ne sera pas rédigée. |
| | `void SaveToFile(string fileName, bool prettyPrint, bool omitXmlDecl, string encoding, bool bBigEndian, bool bBOM, string lineend)` | Enregistre un document XML sous un fichier, avec un formatage "pretty-print" optionnel, avec l'encodage spécifié, l'ordre de byte, BOM (Byte Order Mark) et le(s) caractère(s) de fin de ligne. Si `omitXmlDecl` est true, la déclaration XML ne sera pas rédigée. |
| | `void SaveToFileWithLineEnd(string fileName, bool prettyPrint, bool omitXmlDecl, string lineend)` | Enregistre un document XML sous un fichier, avec un formatage "pretty-print" optionnel et le(s) caractère(s) de fin de ligne. Si `omitXmlDecl` est true, la déclaration XML ne sera pas rédigée. |
| | `string SaveToString(bool prettyPrint)` | Enregistre un document XML sous un fichier, avec un formatage "pretty-print" optionnel. |
| | `string SaveToString(bool prettyPrint, bool omitXmlDecl)` | Enregistre un document XML sous un fichier, avec un formatage "pretty-print" optionnel. Si `omitXmlDecl` est true, la déclaration XML ne sera pas rédigée. |

| | Nom | Description |
|---|---|---|
| | **void** SetDTDLocation(**string** dtdLocation) | Ajoute une déclaration DOCTYPE avec l'ID de système spécifiée. Un élément root doit déjà exister. |
| | **void** SetSchemaLocation(**string** schemaLocation) | Ajoute un attribut xsi:schemaLocation ou xsi:noNamespaceSchemaLocation à l'élément root. Un élément root doit déjà exister. |

## 15.2.5.9 [YourSchema].[ElementType]

Cette classe propose des méthodes pour manipuler des éléments XML depuis votre schéma. Des méthodes de cette classe peuvent être appelées sur des éléments, pas sur le document XML lui-même. Noter qu'il n'est pas nécessaire d'instancier la classe directement pour appeler des méthodes de cette classe. Tout élément créé en utilisant les méthodes **Append()** ou **AppendWithPrefix()** est de type [ElementType].

### Méthodes

| | Nom | Description |
|---|---|---|
| | void DeclareNamespace(string prefix, string nsURI) | Cette méthode prend deux arguments qui sont tous les deux de type string : le préfixe et l'URI d'espace de noms que vous souhaitez utiliser. Le préfixe fournit en tant qu'argument sera mappé dans la valeur URI d'espace de noms fournie en tant qu'argument. Si le préfixe fourni en tant qu'argument est vide, la méthode crée ou contourne la déclaration d'espace de noms par défaut dans l'élément.<br><br>Par exemple, assumons que le document XML a un élément XML appelé "purchase". Si vous appelez<br><br>`purchase.DeclareNamespace("ord", "http://OrderTypes");`<br><br>Alors le document XML devient<br><br>`<purchase xmlns:ord="http://OrderTypes" />`<br><br>Un autre exemple ; si vous appelez :<br><br>`purchase.DeclareNamespace("", "http://OrderTypes");`<br><br>Alors le document XML devient<br><br>`<purchase xmlns="http://OrderTypes" />`<br><br>Note : l'espace de noms déclaré est utilisé lorsque les |

| | Nom | Description |
|---|---|---|
| | | éléments ou les attributs enfants ultérieurs, conformément aux règles suivantes :<br><br>1. si l'espace de noms enfant est le défaut, utiliser le préfixe vide.<br>2. si l'espace de noms enfant est égal à l'espace de noms parent, utiliser le préfixe parent.<br>3. Sinon, chercher le préfixe le plus proche en commençant du parent vers le haut, en utilisant l'algorithme lookup décrit dans la section "B.2: Namespace Prefix Lookup" at https://www.w3.org/TR/2002/WD-DOM-Level-3-Core-20021022/namespaces-algorithms.html.<br>4. Si aucun préfixe n'a été trouvé pour l'espace de noms élément, utiliser un préfixe vide. |

## 15.2.5.10 [YourSchemaType].MemberAttribute

Lorsque le code est généré depuis un schéma XML, une classe est créée pour chaque attribut membre d'un type. Dans les descriptions ci-dessous, "AttributeType" représente le type de l'attribut de membre lui-même.

### Méthodes

| | Nom | Description |
|---|---|---|
| | `bool Exists()` | Retourne true si l'attribut existe. |
| | `void Remove()` | Supprime l'attribut depuis son élément parent. |

### Propriétés

| | Nom | Description |
|---|---|---|
| | **`int`** `EnumerationValue` | Généré pour des types d'énumération uniquement. Définit ou obtient la valeur d'attribut en utilisant une des constantes générée pour les valeurs possibles. Retourne **Invalid** si les valeurs ne correspondent pas à une des valeurs énumérées dans le schéma. |
| | `Altova.Xml.Meta.Attribute Info` | Retourne un objet pour l'information de schéma de requête (voir `Altova.Xml.Meta.Attribute` 1020 ). |
| | `AttributeType Value` | Définit ou obtient la valeur d'attribut. |

## 15.2.5.11 [YourSchemaType].MemberElement

Lorsque le code est généré depuis un schéma XML, une classe avec les membres suivants est créée pour chaque élément de membre d'un type. La classe met en place l'interface standard `System.Collections.IEnumerable`, donc il peut être utilisé avec l'instruction `foreach`.

Dans les descriptions ci-dessous, "MemberType" représente le type de l'élément de membre lui-même.

### Méthodes

| | Nom | Description |
|---|---|---|
| | `MemberType Append()` | Crée un nouvel élément et l'ajoute à son parent. |
| | `MemberType AppendWithPrefix(string prefix)` | Crée un nouvel élément ayant le préfixe fournit en tant qu'argument, et l'appose à son parent. Pour consulter un exemple, voir Exemple : Purchase Order 990. |
| | `MemberType At(int index)` | Retourne l'élément de membre spécifié par l'index. |
| | `System.Collections.IEnumerator GetEnumerator()` | Retourne un objet pour itérer des instances de l'élément de membre. |
| | `void Remove()` | Supprime toutes les occurrences de l'élément depuis son parent. |
| | `void RemoveAt(int index)` | Supprime l'occurrence de l'élément spécifié par l'index. |

### Propriétés

| | Nom | Description |
|---|---|---|
| | `int Count` | Retourne le décompte des éléments. |
| | `int EnumerationValue` | Généré pour des types d'énumération uniquement. Définit ou obtient la valeur d'élément en utilisant une des constantes générée pour les valeurs possibles. Retourne **Invalid** si les valeurs ne correspondent pas à une des valeurs énumérées dans le schéma. |
| | `bool Exists` | Retourne true si au moins un élément existe. |
| | `MemberType First` | Retourne la première instance de l'élément de membre. |
| | `Altova.Xml.Meta.Element Info` | Retourne un objet pour l'information de schéma de requête (voir `Altova.Xml.Meta.Element` 1021 ). |
| | `MemberType Last` | Retourne la dernière instance de l'élément de membre. |
| | `MemberType this[int index]` | Retourne l'élément de membre spécifié par l'index. |

| | Nom | Description |
|---|---|---|
| | `MemberType Value` | Définit ou obtient le contenu d'élément (uniquement généré si l'élément peut avoir un contenu simple ou mixte). |

## 15.2.6 Référence aux classes générées (Java)

Ce chapitre contient une description des classes Java générées avec MapForce depuis un schéma XML ou DTD (voir Générer du code depuis des Schémas XML ou DTDs 953 ). Vous pouvez intégrer ces classes dans votre code pour lire, modifier et écrire des documentation XML.

**Note:** Le code généré peut inclure d'autres classes de support, qui ne sont pas recensées ici et qui sont sujettes à des modifications.

### 15.2.6.1 com.altova.types.DateTime

Cette classe vous permet de traiter des attributs XML ou des éléments qui ont des types de date et d'heure, comme `xs:dateTime`.

#### Constructeurs

| | Nom | Description |
|---|---|---|
| C | **public** `DateTime()` | Initialise une nouvelle instance de la classe `DateTime` dans une valeur vide. |
| C | **public** `DateTime(DateTime newvalue)` | Initialise une nouvelle instance de la classe `DateTime` dans l'objet `DateTime` fourni en tant qu'argument. |
| C | **public** `DateTime(`**int** `newyear,` **int** `newmonth,` **int** `newday,` **int** `newhour,` **int** `newminute,` **int** `newsecond,` **double** `newpartsecond,` **int** `newoffsetTZ)` | Initialise une nouvelle instance de la classe `DateTime` à l'année, mois, jour, heure, minute, seconde, la partie fractionnaire de la seconde et fuseau horaire fournis en tant qu'argument. La partie fractionnaire de la seconde newpartsecond doit se situer entre 0 et 1. L'offset de fuseau horaire newoffsetTZ peut être soit positif soit négatif et s'exprime en minutes. |
| C | **public** `DateTime(`**int** `newyear,` **int** `newmonth,` **int** `newday,` **int** `newhour,` **int** `newminute,` **int** `newsecond,` **double** `newpartsecond)` | Initialise une nouvelle instance de la classe `DateTime` dans l'année, le mois, le jour, l'heure, la minute, la seconde et la partie fractionnelle d'une seconde fournie en tant qu'arguments. |
| C | **public** `DateTime(`**int** `newyear,` **int** `newmonth,` **int** `newday)` | Initialise une nouvelle instance de la classe `DateTime` pour l'année, le mois et le jour fournis en tant qu'arguments. |
| C | **public** `DateTime(Calendar newvalue)` | Initialise une nouvelle instance de la classe `DateTime` dans la valeur `java.util.Calendar` fournie en tant qu'argument. |

## Méthodes

| | Nom | Description |
|---|---|---|
| S | **static** DateTime now() | Retourne l'heure actuelle en tant qu'un objet DateTime. |
| S | **static** DateTime parse(String s) | Retourne un objet DateTime parsé depuis la valeur de string fournie en tant qu'argument. Par exemple, les valeurs de string d'échantillon suivantes sont converties avec succès dans un objet DateTime :<br><br>2015-11-24T12:54:47.969+01:00<br>2015-11-24T12:54:47<br>24/11/2015 |
| | **int** getDay() | Retourne le jour de l'instance actuelle de DateTime. |
| | **int** getHour() | Retourne l'heure de l'instance actuelle de DateTime. |
| | **int** getMillisecond() | Retourne la milliseconde de l'instance actuelle de DateTime, en tant qu'une valeur d'entier. |
| | **int** getMinute() | Retourne la minute de l'instance actuelle DateTime. |
| | **int** getMonth() | Retourne le mois de l'instance actuelle DateTime. |
| | **double** getPartSecond() | Retourne la partie fractionnaire de la seconde de l'instance actuelle de DateTime, en tant que valeur **double**. La valeur de retour est supérieure à zéro et inférieure à un, par exemple :<br><br>0.313 |
| | **int** getSecond() | Retourne la seconde de l'instance actuelle DateTime. |
| | **int** getTimezoneOffset() | Retourne l'offset de fuseau horaire, en minutes de l'instance actuelleDateTime. Par exemple, le fuseau horaire "UTC-01:00" serait retourné en tant que :<br><br>-60 |
| | Calendar getValue() | Retourne l'instance actuelle de DateTime en tant qu'une valeur java.util.Calendar. |
| | **int** getWeekday() | Retourne le jour dans la semaine de l'instance actuelle DateTime. Les valeurs s'étendent de 0 à 6, 0 étant lundi (ISO-8601). |
| | **int** getYear() | Retoune l'année de l'instance actuelle DateTime. |
| | **int** hasTimezone() | Retourne l'information concernant le fuseau horaire de l'instance actuelle DateTime. Les valeurs de retour possibles sont :<br>CalendarBase.TZ_MISSING — Un offset de fuseau horaire n'est pas défini. |

| | Nom | Description |
|---|---|---|
| | | `CalendarBase.TZ_UTC` Le fuseau horaire est UTC.<br>`CalendarBase.TZ_OFFSET` Un offset de fuseau d'horaire a été défini. |
| ● | **void** `setDay( int nDay )` | Définit le jour de l'instance actuelle `DateTime` à la valeur fournie en tant qu'argument. |
| ● | **void** `setHasTimezone( int nHasTZ )` | Définit l'information de fuseau horaire de l'instance actuelle `DateTime` à la valeur fournie en tant qu'argument. Cette méthode peut être utilisée pour supprimer l'information de fuseau horaire ou définir le fuseau horaire sur UTC (Coordinated Universal Time). Les valeurs valides pour l'argument nHasTZ :<br>`CalendarBase.TZ_MISSING` Configurer l'offset de fuseau horaire sur non-défini.<br>`CalendarBase.TZ_UTC` Configurer le fuseau horaire sur UTC.<br>`CalendarBase.TZ_OFFSET` Si l'objet actuel a un offset de fuseau horaire, ne pas le modifier. |
| ● | **void** `setHour( int nHour )` | Définit l'heure de l'instance actuelle `DateTime` à la valeur fournie en tant qu'argument. |
| ● | **void** `setMinute( int nMinute )` | Définit la minute de l'instance actuelle `DateTime` à la valeur fournie en tant qu'argument. |
| ● | **void** `setMonth( int nMonth )` | Définit le mois de l'instance actuelle `DateTime` à la valeur fournie en tant qu'argument. |
| ● | **void** `setPartSecond( double nPartSecond )` | Définit la partie fractionnaire de la seconde de l'instance actuelle `DateTime` à la valeur fournie en tant qu'argument. |
| ● | **void** `setSecond( int nSecond )` | Définit la seconde de l'instance actuelle `DateTime` à la valeur fournie en tant qu'argument. |
| ● | **void** `setTimezoneOffset( int nOffsetTZ )` | Définit l'offset de fuseau horaire de l'instance actuelle `DateTime` à la valeur fournie en tant qu'argument. La valeur nOffsetTZ doit être un entier (positif ou négatif) et doit être exprimée en minutes. |
| ● | **void** `setYear( int nYear )` | Définit l'année de l'instance actuelle `DateTime` à la valeur fournie en tant qu'argument. |
| ● | `String toString()` | Retourne la représentation de string de l'instance actuelle `DateTime`, par exemple :<br>`2015-11-24T15:50:56.968+01:00` |

## Exemples

Avant d'utiliser les extraits de code suivants dans votre programme, s'assurer que les types Altova sont importés :

```
import com.altova.types.*;
```

L'extrait de code suivant illustre plusieurs moyens de créer des objets `DateTime` :

```
protected static void DateTimeExample1()
{
    // Initialize a new instance of the DateTime class to the current time
   DateTime dt = new DateTime(DateTime.now());
   System.out.println("DateTime created from current date and time: " + dt.toString());

   // Initialize a new instance of the DateTime class by supplying the parts
   DateTime dt1 = new DateTime(2015, 11, 23, 14, 30, 24, .459);
   System.out.println("DateTime from parts (no timezone): " + dt1.toString());

   // Initialize a new instance of the DateTime class by supplying the parts
   DateTime dt2 = new DateTime(2015, 11, 24, 14, 30, 24, .459, -60);
   System.out.println("DateTime from parts (with negative timezone): " + dt2.toString());

   // Initialize a new instance of the DateTime class by parsing a string value
   DateTime dt3 = DateTime.parse("2015-11-24T12:54:47.969+01:00");
   System.out.println("DateTime parsed from string: " + dt3.toString());
}
```

The following code listing illustrates getting values from `DateTime` objects:

```
protected static void DateTimeExample2()
   {
      // Initialize a new instance of the DateTime class to the current time
      DateTime dt = new DateTime(DateTime.now());

      // Output the formatted year, month, and day of this DateTime instance
      String str1 = String.format("Year: %d; Month: %d; Day: %d;", dt.getYear(),
dt.getMonth(), dt.getDay());
      System.out.println(str1);

      // Output the formatted hour, minute, and second of this DateTime instance
      String str2 = String.format("Hour: %d; Minute: %d; Second: %d;", dt.getHour(),
dt.getMinute(), dt.getSecond());
      System.out.println(str2);

      // Return the timezone (in minutes) of this DateTime instance
      System.out.println("Timezone: " + dt.getTimezoneOffset());

      // Get the DateTime as a java.util.Calendar value
      java.util.Calendar dt_java = dt.getValue();
```

```
      System.out.println("" + dt_java.toString());

      // Return the day of week of this DateTime instance
      System.out.println("Weekday: " + dt.getWeekday());

      // Check whether the DateTime instance has a timezone defined
      switch(dt.hasTimezone())
      {
         case CalendarBase.TZ_MISSING:
            System.out.println("No timezone.");
            break;
         case CalendarBase.TZ_UTC:
            System.out.println("The timezone is UTC.");
            break;
         case CalendarBase.TZ_OFFSET:
            System.out.println("This object has a timezone.");
            break;
         default:
            System.out.println("Unable to determine whether a timezone is defined.");
            break;
      }
   }
```

The following code listing illustrates changing the timezone offset of a `DateTime` object:

```
protected static void DateTimeExample3()
{
   // Create a new DateTime object with timezone -0100 UTC
   DateTime dt = new DateTime(2015, 11, 24, 14, 30, 24, .459, -60);
   // Output the value before the change
   System.out.println("Before: " + dt.toString());
   // Change the offset to +0100 UTC
   dt.setTimezoneOffset(60);
   // Output the value after the change
   System.out.println("After:  " + dt.toString());
}
```

## 15.2.6.2 com.altova.types.Duration

Cette classe vous permet de traiter des attributs XML ou des éléments de type `xs: duration` .

### Constructeurs

| | Nom | Description |
|---|---|---|
| C | `Duration(Duration newvalue)` | Initialise une nouvelle instance de la classe `Duration` dans l'objet `Duration` fournie en tant qu'argument. |
| C | `Duration(int newyear, int newmonth, int newday, int` | Initialise une nouvelle instance de la classe `Duration` dans une durée construite depuis des parties fournies en tant qu'argument. |

| | Nom | Description |
|---|---|---|
| | `newhour, int newminute, int newsecond, double newpartsecond, boolean newisnegative)` | |

## Méthodes

| | Nom | Description |
|---|---|---|
| S | `static Duration getFromDayTime( int newday, int newhour, int newminute, int newsecond, double newpartsecond )` | Retourne un objet `Duration` créé depuis le nombre de jours, heures, minutes, secondes et partie de seconde fractionnelle fourni en tant qu'argument. |
| S | `static Duration getFromYearMonth( int newyear, int newmonth )` | Retourne un objet `Duration` créé depuis le nombre d'années et de mois fourni en tant qu'argument. |
| S | `static Duration parse( String s )` | Retourne un objet `Duration` créé depuis le string fourni en tant qu'argument. Par exemple, le string -P1Y1M1DT1H1M1.333S peut être utilisé pour créer une durée négative d'une année, d'un mois, d'un jour, d'une heure, d'une minute, d'une seconde et des parties fractionnelles 0.333 d'une seconde. Pour créer une durée négative, apposer le signe moins ( - ) au string. |
| S | `static Duration parse( String s, ParseType pt )` | Retourne un objet `Duration` créé depuis le string fourni en tant qu'argument, en utilisant un format de parsage spécifique. Le format de parsage peut être un des suivants :<br>**ParseType.DAYTIME** : Peut être utilisé lorsque le string **s** consiste en un des éléments suivants : jours, heures, minutes, secondes, parties fractionnelles de la seconde, par exemple -P4DT4H4M4.774S.<br>**ParseType.DURATION** : Peut être utilisé lorsque le string **s** consiste en un des éléments suivants : années, mois, jours, heures, minutes, secondes, parties fractionnelles de la seconde, par exemple P1Y1M1DT1H1M1.333S.<br>**ParseType.YEARMONTH** : Peut être utilisé lorsque le string **s** consiste en un des éléments suivants : années, mois. Par exemple : P3Y2M. |
| | `int getDay()` | Retourne le nombre de jours dans l'instance `Duration` actuelle. |

| | Nom | Description |
|---|---|---|
| | `long getDayTimeValue()` | Retourne la valeur de jour et d'heure (en millisecondes) de l'instance `Duration` actuelle. Les années et le mois sont ignorés. |
| | `int getHour()` | Retourne le nombre d'heures dans l'instance `Duration` actuelle. |
| | `int getMillisecond()` | Retourne le nombre de millisecondes dans l'instance `Duration` actuelle. |
| | `int getMinute()` | Retourne le nombre de minutes dans l'instance `Duration` actuelle. |
| | `int getMonth()` | Retourne le nombre de mois dans l'instance `Duration` actuelle. |
| | `double getPartSecond()` | Retourne le nombre de parties de secondes fractionnelles dans l'instance `Duration` actuelle. |
| | `int getSecond()` | Retourne le nombre de secondes dans l'instance `Duration` actuelle. |
| | `int getYear()` | Retourne le nombre des années dans l'instance `Duration` actuelle. |
| | `int getYearMonthValue()` | Retourne la valeur (en mois) de l'année et du mois de l'instance `Duration` actuelle. Les jours, heures, secondes et millisecondes sont ignorés. |
| | `boolean isNegative()` | Retourne Boléenne **true** si l'instance `Duration` actuelle est négative. |
| | `void setDayTimeValue(long l)` | Définit la durée dans le nombre de millisecondes fournies en tant qu'argument, ne touchant que la partie jour et heure de la durée. |
| | `void setNegative( boolean isnegative )` | Convertit l'instance actuelle `Duration` en une durée négative. |
| | `void setYearMonthValue(int l)` | Définit la durée dans le nombre de mois fournis en tant qu'argument. Seule la partie années et mois de la durée est touchée. |
| | `String toString()` | Retourne la représentation de string de l'instance actuelle `Duration`, par exemple :<br><br>`-P4DT4H4M4.774S` |
| | `String toYearMonthString()` | Retourne la représentation de string de la partie YearMonth de l'instance actuelle `Duration`, par exemple :<br><br>`P1Y2M` |

## Exemples

Avant d'utiliser les extraits de code suivants dans votre programme, s'assurer que les types Altova sont importés :

```
import com.altova.types.*;
import com.altova.types.Duration.ParseType;
```

L'extrait de code suivant illustre plusieurs moyens de créer des objets `Duration` :

```
protected static void ExampleDuration()
{
   // Create a negative duration of 1 year, 1 month, 1 day, 1 hour, 1 minute, 1 second,
   // and 0.333 fractional second parts
   Duration dr = new Duration(1, 1, 1, 1, 1, 1, .333, true);

   // Create a duration from an existing Duration object
   Duration dr1 = new Duration(dr);

   // Create a duration of 4 days, 4 hours, 4 minutes, 4 seconds, .774 fractional second
parts
   Duration dr2 = Duration.getFromDayTime(4, 4, 4, 4, .774);

   // Create a duration of 3 years and 2 months
   Duration dr3 = Duration.getFromYearMonth(3, 2);

   // Create a duration from a string
   Duration dr4 = Duration.parse("-P4DT4H4M4.774S");

   // Create a duration from a string, using specific parse formats
   Duration dr5 = Duration.parse("-P1Y1M1DT1H1M1.333S", ParseType.DURATION);
   Duration dr6 = Duration.parse("P3Y2M", ParseType.YEARMONTH);
   Duration dr7 = Duration.parse("-P4DT4H4M4.774S", ParseType.DAYTIME);
}
```

L'extrait de code suivant illustre l'obtention et la configuration d'objets `Duration` :

```
protected static void DurationExample2()
{
   // Create a duration of 1 year, 2 month, 3 days, 4 hours, 5 minutes, 6 seconds,
       // and 333 milliseconds
   Duration dr = new Duration(1, 2, 3, 4, 5, 6, .333, false);
   // Output the number of days in this duration
   System.out.println(dr.getDay());

   // Create a positive duration of one year and 333 milliseconds
   Duration dr1 = new Duration(1, 0, 0, 0, 0, 0, .333, false);
   // Output the day and time value in milliseconds
   System.out.println(dr1.getDayTimeValue());
```

```
   // Create a positive duration of 1 year, 1 month, 1 day, 1 hour, 1 minute, 1 second,
       // and 333 milliseconds
   Duration dr2 = new Duration(1, 1, 1, 1, 1, 1, .333, false);
   // Output the year and month value in months
   System.out.println(dr2.getYearMonthValue());

   // Create a positive duration of 1 year and 1 month
   Duration dr3 = new Duration(1, 1, 0, 0, 0, 0, 0, false);
   // Output the value
   System.out.println("The duration is now: " + dr3.toString());
   // Set the DayTime part of duration to 1000 milliseconds
   dr3.setDayTimeValue(1000);
   // Output the value
   System.out.println("The duration is now: " + dr3.toString());
   // Set the YearMonth part of duration to 1 month
   dr3.setYearMonthValue(1);
   // Output the value
   System.out.println("The duration is now: " + dr3.toString());
   // Output the year and month part of the duration
   System.out.println("The YearMonth part of the duration is: " +
dr3.toYearMonthString());
}
```

## 15.2.6.3 com.altova.xml.meta.Attribute

Cette classe vous permet d'accéder à des informations de schéma concernant des classes générées depuis des attributs. Veuillez noter que cette classe n'est pas destinée à fournir des informations dynamiques à propos des instances particulières d'un attribut dans un document XML. Au lieu, il vous permet d'obtenir des informations par le biais d'un programme concernant un attribut particulier défini dans le schéma XML.

### Méthodes

| | Nom | Description |
|---|---|---|
| | `SimpleType getDataType()` | Retourne le type du contenu d'attribut. |
| | `String getLocalName()` | Retourne le nom local de l'attribut. |
| | `String getNamespaceURI()` | Retourne l'URI d'espace de noms de l'attribut. |
| | **`boolean`** `isRequired()` | Retourne true si l'attribut est requis. |

## 15.2.6.4 com.altova.xml.meta.ComplexType

Cette classe vous permet d'accéder à des informations de schéma concernant des classes générées depuis des types complexes. Veuillez noter que cette classe n'est pas destinée à fournir des informations dynamiques à propos des instances particulières des types complexes dans un document XML. Au lieu, elle vous permet d'obtenir des informations par le biais d'un programme concernant un type complexes particulier défini dans le schéma XML.

### Méthodes

| | Nom | Description |
|---|---|---|
| | `Attribute findAttribute(String localName, String namespaceURI)` | Trouve l'attribut avec le nom local spécifié et l'URI d'espace de noms |
| | `Element findElement(String localName, String namespaceURI)` | Trouve l'élément avec le nom local spécifié et l'URI d'espace de noms. |
| | `Attribute[] GetAttributes()` | Retourne une liste de tous les attributs. |
| | `ComplexType getBaseType()` | Retourne le type de base de ce type. |
| | `SimpleType getContentType()` | Retourne le type simple du contenu. |
| | `Element[] GetElements()` | Retourne une liste de tous les éléments. |
| | `String getLocalName()` | Retourne le nom local du type. |
| | `String getNamespaceURI()` | Retourne l'URI d'espace de noms du type. |

## 15.2.6.5 com.altova.xml.meta.Element

Cette classe vous permet d'accéder à des informations concernant des classes générées depuis des éléments de schéma. Veuillez noter que cette classe n'est pas destinée à fournir des informations dynamiques à propos des instances particulières d'un élément dans un document XML. Au lieu, il vous permet d'obtenir des informations par le biais d'un programme concernant un élément particulier défini dans le schéma XML.

### Méthodes

| | Nom | Description |
|---|---|---|
| | `ComplexType getDataType()` | Retourne le type de l'élément. Noter qu'il s'agit toujours d'un type de complexe même s'il est déclaré en tant que simple dans le schéma original. Utiliser `getContentType()` de l'objet retourné pour obtenir le type de contenu simple. |
| | `String getLocalName()` | Retourne le nom local de l'élément. |
| | **`int`** `getMaxOccurs()` | Retourne la valeur maxOccurs définie dans le schéma. |
| | **`int`** `getMinOccurs()` | Retoune la valeur minOccurs définie dans le schéma. |
| | `String getNamespaceURI()` | Retoune l'URI d'espace de noms de l'élément . |

## 15.2.6.6 com.altova.xml.meta.SimpleType

Cette classe vous permet d'accéder à des informations de schéma concernant des classes générées depuis des types simples. Veuillez noter que cette classe n'est pas destinée à fournir des informations dynamiques à propos des instances particulières des types simples dans un document XML. Au lieu, elle vous permet d'obtenir des informations par le biais d'un programme concernant un type simple particulier défini dans le schéma XML.

### Méthodes

| | Nom | Description |
|---|---|---|
| | `SimpleType getBaseType()` | Retourne le type de base de ce type. |
| | `String[] getEnumerations()` | Retourne un array de toutes les facettes d'énumération. |
| | `int getFractionDigits()` | Retourne la valeur de cette facette. |
| | `int getLength()` | Retourne la valeur de cette facette. |
| | `String getLocalName()` | Retourne le nom local du type. |
| | `String getMaxExclusive()` | Retourne la valeur de cette facette. |
| | `String getMaxInclusive()` | Retourne la valeur de cette facette. |
| | `int getMaxLength()` | Retourne la valeur de cette facette. |
| | `String getMinExclusive()` | Retourne la valeur de cette facette. |
| | `String getMinInclusive()` | Retourne la valeur de cette facette. |
| | `int getMinLength()` | Retourne la valeur de cette facette. |
| | `String getNamespaceURI()` | Retourne l'URI d'espace de noms du type. |
| | `String[] getPatterns()` | Retoune un array de toutes les facettes de motif. |
| | `int getTotalDigits()` | Retourne la valeur de cette facette. |
| | `int getWhitespace()` | Retourne la valeur de la facette d'espace blanc, qui est un de : `com.altova.typeinfo.WhitespaceType.Whitespace_Unknown` `com.altova.typeinfo.WhitespaceType.Whitespace_Preserve` `com.altova.typeinfo.WhitespaceType.Whitespace_Replace` `com.altova.typeinfo.WhitespaceType.Whitespace_Collapse` |

## 15.2.6.7 com.[YourSchema].[Doc]

Lorsque le code est généré depuis un schéma XML, le code généré fournit une classe de document avec le même nom que le schéma. Cette classe contient tous les éléments root possibles en tant que membres, ainsi

que les membres recensés ci-dessous. Veuillez noter que, dans les noms de méthode ci-dessous, "Doc" représente le nom de la classe de document générée elle-même.

## Méthodes

| | Nom | Description |
|---|---|---|
| S | `static Doc createDocument()` | Crée un nouveau document XML vide. |
| S | `static void declareAllNamespacesFromSchema(com.altova.xml.ElementType node)` | Déclare tous les espaces de noms depuis le Schéma XML dans l'élément fournit en dans qu'argument (généralement, l'élément racine XML). L'appel de cette méthode est utile si votre schéma a plusieurs déclarations d'espace de noms, chacune mappée dans un préfixe, et que vous souhaitez les déclarer tous dans l'élément fourni en tant qu'argument. |
| S | `static Doc loadFromBinary(byte[] xml)` | Charge un document XML depuis un byte array. |
| S | `static Doc loadFromFile(String fileName)` | Charge un document XML depuis un fichier. |
| S | `static Doc loadFromString(String xml)` | Charge un document XML depuis un string. |
| | `byte[] saveToBinary(boolean prettyPrint)` | Enregistre un document XML sous un de byte, avec un formatage "pretty-print" optionnel. |
| | `byte[] saveToBinary(boolean prettyPrint, String encoding)` | Enregistre un document XML sous un byte array, avec un formatage "pretty-print" optionnel, avec l'encodage spécifié. |
| | `byte[] saveToBinary(boolean prettyPrint, String encoding, boolean bigEndian, boolean writeBOM)` | Enregistre un document XML sous un byte array, avec un formatage "pretty-print" optionnel, avec l'encodage spécifié. La marque Byte order et Unicode byte-order peuvent être spécifiées pour les encodages Unicode. |
| | `void saveToFile(String fileName, boolean prettyPrint)` | Enregistre un document XML sous un fichier, avec un formatage "pretty-print" optionnel. |
| | `void saveToFile(String fileName, boolean prettyPrint, boolean omitXmlDecl)` | Enregistre un document XML sous un fichier, avec un formatage "pretty-print" optionnel, avec l'encodage UTF-8. Si `omitXmlDecl` est true, la déclaration XML ne sera pas rédigée. |
| | `void saveToFile(String fileName, boolean prettyPrint, boolean omitXmlDecl, String encoding)` | Enregistre un document XML sous un fichier, avec un formatage "pretty-print" optionnel, avec l'encodage spécifié. Si `omitXmlDecl` est true, la déclaration XML ne sera pas rédigée. |
| | `void saveToFile(String fileName, boolean prettyPrint, boolean omitXmlDecl, String encoding,` | Enregistre un document XML sous un fichier, avec un formatage "pretty-print" optionnel, avec l'encodage spécifié. Si `omitXmlDecl` est true, la déclaration XML ne sera pas rédigée. La marque Byte order et Unicode byte-order peuvent être spécifiées pour les encodages Unicode. |

| | Nom | Description |
|---|---|---|
| | `boolean bBigEndian, boolean bBOM)` | |
| | `void saveToFile(String fileName, boolean prettyPrint, String encoding)` | Enregistre un document XML sous un fichier, avec un formatage "pretty-print" optionnel, avec l'encodage spécifié. |
| | `void saveToFile(String fileName, boolean prettyPrint, String encoding, boolean bBigEndian, boolean bBOM)` | Enregistre un document XML sous un fichier, avec un formatage "pretty-print" optionnel, avec l'encodage spécifié. La marque Byte order et Unicode byte-order peuvent être spécifiées pour les encodages Unicode. |
| | `String saveToString(boolean prettyPrint)` | Enregistre un document XML sous un string, avec un formatage "pretty-print" optionnel. |
| | `String saveToString(boolean prettyPrint, boolean omitXmlDecl)` | Enregistre un document XML sous un string, avec un formatage "pretty-print" optionnel. Si `omitXmlDecl` est true, la déclaration XML ne sera pas rédigée. |
| | `void setSchemaLocation(String schemaLocation)` | Ajoute un attribut `xsi:schemaLocation` ou `xsi:noNamespaceSchemaLocation` à l'élément root. Un élément root doit déjà exister. |

## 15.2.6.8 com.[YourSchema].[ElementType]

Cette classe propose des méthodes pour manipuler des éléments XML depuis votre schéma. Des méthodes de cette classe peuvent être appelées sur des éléments, pas sur le document XML lui-même. Noter qu'il n'est pas nécessaire d'instancier la classe directement pour appeler des méthodes de cette classe. Tout élément créé en utilisant les méthodes **`append()`** ou **`appendWithPrefix()`** est de type `ElementType`.

### Méthodes

| | Nom | Description |
|---|---|---|
| | `void declareNamespace(String prefix, String nsURI)` | Cette méthode prend deux arguments qui sont tous les deux de type string : le préfixe et l'URI d'espace de noms que vous souhaitez utiliser. Le préfixe fournit en tant qu'argument sera mappé dans la valeur URI d'espace de noms fournie en tant qu'argument. Si le préfixe fourni en tant qu'argument est vide, la méthode crée ou contourne la déclaration d'espace de noms par défaut dans l'élément.<br><br>Par exemple, assumons que le document XML a un élément XML appelé "purchase". Si vous appelez<br><br>`purchase.declareNamespace("ord", "http://OrderTypes");` |

| | Nom | Description |
|---|---|---|
| | | Alors le document XML devient<br><br>`<purchase xmlns:ord="http://OrderTypes" />`<br><br>Un autre exemple ; si vous appelez :<br><br>`purchase.declareNamespace("", "http://OrderTypes");`<br><br>Alors le document XML devient<br><br>`<purchase xmlns="http://OrderTypes" />`<br><br>Note : l'espace de noms déclaré est utilisé lorsque les éléments ou les attributs enfants ultérieurs, conformément aux règles suivantes :<br><br>1. si l'espace de noms enfant est le défaut, utiliser le préfixe vide.<br>2. si l'espace de noms enfant est égal à l'espace de noms parent, utiliser le préfixe parent.<br>3. Sinon, chercher le préfixe le plus proche en commençant du parent vers le haut, en utilisant l'algorithme lookup décrit dans la section "B.2: Namespace Prefix Lookup" at https://www.w3.org/TR/2002/WD-DOM-Level-3-Core-20021022/namespaces-algorithms.html.<br>4. Si aucun préfixe n'a été trouvé pour l'espace de noms élément, utiliser un préfixe vide. |

## 15.2.6.9 com.[YourSchema].[YourSchemaType].MemberAttribute

Lorsque le code est généré depuis un schéma XML, une classe est créée pour chaque attribut membre d'un type. Dans les descriptions ci-dessous, "AttributeType" représente le type de l'attribut de membre lui-même.

### Méthodes

| | Nom | Description |
|---|---|---|
| | `boolean exists()` | Retourne true si l'attribut existe. |
| | `int getEnumerationValue()` | Généré pour des types d'énumération uniquement. Retourne une des constantes générée pour les valeurs possibles, ou **Invalid** si la valeur ne correspond pas à une des valeurs énumérées dans le schéma. |

| | Nom | Description |
|---|---|---|
| | `com.altova.xml.meta.Attribute getInfo()` | Retourne un objet pour l'information de schéma de requête (voir `com.altova.xml.meta.Attribute`[1036] ). |
| | `AttributeType getValue()` | Définit ou obtient la valeur d'attribut. |
| | **`void`** `remove()` | Supprime l'attribut depuis son élément parent. |
| | **`void`** `setEnumerationValue(`**`int`**`)` | Généré pour des types d'énumération uniquement. Passer une des constantes générées pour les valeurs possibles dans cette méthode pour définir la valeur. |
| | **`void`** `setValue(AttributeType value)` | Définit la valeur d'attribut. |

## 15.2.6.10 com.[YourSchema].[YourSchemaType].MemberElement

Lorsque le code est généré depuis un schéma XML, une classe avec les membres suivants est créée pour chaque élément de membre d'un type. Dans les descriptions ci-dessous, "MemberType" représente le type de l'élément de membre lui-même.

### Méthodes

| | Nom | Description |
|---|---|---|
| | `MemberType append()` | Crée un nouvel élément et l'ajoute à son parent. |
| | `MemberType appendWithPrefix(String prefix)` | Crée un nouvel élément ayant le préfixe fournit en tant qu'argument, et l'appose à son parent. Pour consulter un exemple, voir Exemple : Purchase Order[990]. |
| | `MemberType at(`**`int`**` index)` | Retourne l'instance de l'élément membre en tant que l'index spécifié. |
| | **`int`** `count()` | Retourne le décompte des éléments. |
| | **`boolean`** `exists()` | Retourne true si au moins un élément existe. |
| | `MemberType first()` | Retourne la première instance de l'élément de membre. |
| | **`int`** `getEnumerationValue()` | Généré pour des types d'énumération uniquement. Retourne une des constantes générée pour les valeurs possibles, ou **Invalid** si la valeur ne correspond pas à une des valeurs énumérées dans le schéma. |
| | `com.altova.xml.meta.Element getInfo()` | Retourne un objet pour l'information de schéma de requête (voir `com.altova.xml.meta.Element`[1037] ). |
| | `MemberType getValue()` | Obtient le contenu de l'élément (uniquement généré si l'élément peut avoir un contenu simple ou mixte). |

| | Nom | Description |
|---|---|---|
| | `java.util.Iterator iterator()` | Retourne un objet pour itérer des instances de l'élément de membre. |
| | `MemberType last()` | Retourne la dernière instance de l'élément de membre. |
| | **`void`** `remove()` | Supprime toutes les occurrences de l'élément depuis son parent. |
| | **`void`** `removeAt(`**`int`** `index)` | Supprime l'occurrence de l'élément spécifié par l'index. |
| | **`void`** `setEnumerationValue(`**`int`** `index)` | Généré pour des types d'énumération uniquement. Passer une des constantes générées pour les valeurs possibles dans cette méthode pour définir la valeur. |
| | **`void`** `setValue(MemberType value)` | Définit le contenu d'élément (uniquement généré si l'élément peut avoir un contenu simple ou mixte). |

## 15.2.7 Référence SPL

Cette section donne un aperçu du SPL (Spy Programming Language), langage de modèle du générateur de code. Nous vous recommandons de disposer d'expérience de programmation, et de maîtriser les opérateurs, les fonctions, les variables et les classes, ainsi que les bases de la programmation orientée vers l'objet - très largement utilisé dans SPL.

Les modèles utilisés par MapForce sont fournis dans le dossier de l'application **`spl`**. Vous pouvez utiliser ces fichiers en tant qu'orientation pour vous aider à développer vos propres modèles.

### Comment fonctionne le générateur de code

Le code est généré sur la base des fichiers modèle (**`.spl`** files) et le modèle d'objet fourni par MapForce. Les fichiers modèle contiennent le code du langage de programmation cible combiné avec les instructions SPL pour créer des fichiers, lire des informations depuis un modèle d'objet, et réaliser des calculs.

Le fichier modèle est interprété par le générateur de code et sort les fichiers du source-code du/des langage/s cible (à avoir, les fichiers de code non compilés) et tout autre fichier de projet pertinent ou fichier dépendant du modèle. Le code de source peut ensuite être compilé dans un fichier exécutable qui accède aux données La section XML décrites par le fichier de schéma.

Les fichiers SPL ont accès à une grande variété d'informations recueillie depuis les schémas de source. Veuillez noter qu'un fichier SPL n'est pas lié à un schéma spécifique, mais permet l'accès à tous les schémas. Veuillez vous assurer d'écrire vos fichiers SPL génériquement et d'éviter les structures, qui s'appliquent aux schémas spécifiques.

*Notes relatives aux noms de méthodes*
Lors de la personnalisation de la génération de code en utilisant les fichiers SPL fournis, il peut être nécessaire de réserver des noms pour éviter toute collision avec d'autres symboles. Suivez les instructions ci-dessous :

1. Naviguer dans le répertoire d'installation du programme, par exemple, `C:\Program Files\Altova\MapForce2024`.

2. Dans le sous-répertoire **spl**, trouver le répertoire correspondant au langage de programme, par exemple, **..\spl\java**.
3. Ouvrez le fichier **settings.spl** et insérez une nouvelle ligne dans la section `reserve`, par exemple, **reserve "myReservedWord"**.
4. Régénérer le code de programme.

### Exemple : Créer un nouveau fichier dans SPL

Voici un fichier SPL de base. Il crée un fichier nommé **test.cpp**, et place l'instruction incluse à l'intérieur. La commande close achève le modèle.

```
[create "test.cpp"]
#include "stdafx.h"
[close]
```

## 15.2.7.1 Structure SPL de base

Un fichier SPL contient du texte littéral de sortie, mêlé d'instructions de générateur de code.

Les instructions du générateur de code sont contenues dans des crochets '**[**' et '**]**'. Plusieurs instructions peuvent être contenues dans une paire de crochets. Des instructions supplémentaires doivent être séparées par une nouvelle ligne ou un double point '**:**'.

Des exemples valides sont :

```
[$x = 42
$x = $x + 1]
```

ou

```
[$x = 42: $x = $x + 1]
```

### Ajouter du texte à des fichiers

Le texte non contenu par **[** et **]**, est écrit directement dans le fichier de sortie actuel. S'il n'y a pas de fichier de sortie actuel, le texte est ignoré (voir Utiliser les fichiers 1049 comment créer un fichier de sortie).

Pour produire des crochets littéraux, les échapper avec une barre oblique inversée : **\[** et **\]**; pour produire une barre oblique inversée utiliser ****.

### Commentaires

Les commentaires se trouvant dans un bloc d'instructions commencent toujours avec un caractère **'** , et se terminent sur la ligne suivante, ou à un caractère de bloc de fermeture **]**.

### 15.2.7.2 Déclarations

Les instructions suivantes sont évaluées tout en parsant le fichier de modèle SPL. Elles **ne sont pas** touchées par les instructions de contrôle du flux comme des conditions, des boucles ou des sous-routines, et sont toujours évaluées exactement une seule fois.

Ces mots-clé, comme tous les mots-clé dans SPL, ne sont pas sensibles à la casse.

Veuillez noter que toutes ces déclarations doivent se trouver dans un bloc délimité par des crochets.

#### map ... to ...

```
map mapname key to value [, key to value ]...
```

Cette instruction ajoute des informations dans un mappage. Voir ci-dessous pour des utilisations spécifiques.

```
map schemanativetype schematype to typespec
```

Le type de Schéma XML built-in spécifié sera mappé dans le type natif spécifié ou la classe, en utilisant le formateur spécifié. Ce paramètre s'applique uniquement à la génération de code pour la version 2007r3 et plus. Typespec est un type natif ou un nom de classe, suivi par une virgule, suivi par l'instance de classe de formateur.

Exemple :

```
map schemanativetype "double" en "double,Altova::DoubleFormatter"
```

#### map type ... to ...

```
map type schematype to classname
```

Le type de Schéma XML built-in spécifié sera mappé dans la classe spécifiée. Ce paramètre s'applique uniquement à la génération de code pour la version 2007 ou en-dessous.

Exemple :

```
map type "float" to "CSchemaFloat"
```

#### default ... is ...

```
default setting is value
```

Cette instruction vous permet de toucher la manière dont les noms de classe et de membre sont dérivés depuis le Schéma XML. Veuillez noter que les noms de paramètre sont sensibles à la casse.

Exemple :

```
default "InvalidCharReplacement" is "_"
```

| Nom du paramètre | Explication |
|---|---|
| ValidFirstCharSet | Les caractères autorisés pour lancer un identifiant |
| ValidCharSet | Les caractères autorisés pour d'autres caractères dans un identifiant |
| InvalidCharReplacement | Le caractère qui remplacera tous les caractères dans les noms qui ne se trouvent pas dans ValidCharSet |
| AnonTypePrefix | Préfixe pour les noms des types anonymes* |
| AnonTypeSuffix | Suffixe pour les noms des types anonymes* |
| ClassNamePrefix | Préfixe pour des noms de classe générés |
| ClassNameSuffix | Suffixe pour les noms de classe générés |
| EnumerationPrefix | Préfixe pour les constantes symboliques déclarées pour les valeurs d'énumération |
| EnumerationUpperCase | "on" pour convertir les noms de constante d'énumération pour la casse majuscule |
| FallbackName | Si un nom consiste uniquement en caractères qui ne sont pas dans ValidCharSet, utiliser celui-là |

* Les noms des types anonymes sont construits à partir de AnonTypePrefix + nom d'élément + AnonTypeSuffix

## reserve

```
reserve word
```

Ajoute le mot spécifié pour la liste des mots réservés. Cela garantit qu'il ne sera jamais généré en tant que classe ou de nom de membre.

Exemple :

```
reserve "while"
```

### include

Inclut le fichier spécifié en tant que source SPL. Cela vous permet de partager votre modèle dans plusieurs fichiers pour une édition et une gestion plus simple.

```
include filename
```

Exemple :

```
include "Module.cpp"
```

## 15.2.7.3 Variables

Tout fichier SPL non-trivial nécessitera des variables. Certaines variables sont prédéfinies[1048] par le générateur de code, et des nouvelles variables peuvent être créées simplement en y attribuant des valeurs.

Le caractère **$** est utilisé lors d'une **déclaration** ou **utilisation** d'une variable, un nom de variable est toujours préfixé par **$**. Les noms de variable sont **sensibles à la casse**.

Types de variables :

- entier - aussi utilisé en tant que booléenne, où 0 est faux et tout le reste est vrai
- string
- objet - fourni par MapForce
- iterator - voir instruction foreach[1053]

Les types de variables sont déclarés par la première affectation :

```
[$x = 0]
```

x est maintenant un entier.

```
[$x = "teststring"]
```

x est maintenant traité en tant qu'un string.

### Strings

Les constantes de string sont toujours entourées dans des guillemets doubles, comme dans l'exemple ci-dessus. **\n** et **\t** dans des guillemets doubles sont interprétés en tant que newline et tab, **\"** est un guillemet double littéral, et **** est une barre oblique inversée. Les constantes de string peuvent aussi s'étendre sur plusieurs lignes.

La concaténation de string utilise le caractère **&**

```
[$BasePath = $outputpath & "/" & $JavaPackageDir]
```

### Objets

Les objets représentent l'information contenue dans le schéma XML, les structures de base de données, les fichiers de texte et les mappages. Les objets ont des **propriétés**, qui peuvent être accédées en utilisant l'opérateur **.**. Il n'est pas possible de créer de nouveaux objets dans SPL (ils sont prédéfinis par le générateur de code, dérivés depuis le mapping) d'entrée, mais il est possible d'attribuer des objets aux variables.

Exemple:

```
class [=$class.Name]
```

Cet exemple produit le mot "class", suivi par une espace et la valeur de la propriété **Name** property de l'objet **$class**.

## 15.2.7.4 Variables prédéfinies

Une fois que le fichier de schéma a été analysé par le générateur de code, les objets dans la table ci-dessous existent dans le Template Engine.

| Nom | Type | Description |
|---|---|---|
| $schematype | entier | 1 pour DTD, 2 pour Schema XML |
| $TheLibrary | Bibliothèque 1057 | La bibliothèque dérivée depuis le Schéma XML ou le DTD |
| $module | string | Nom du Schéma de source sans extension |
| $outputpath | string | Le chemin de sortie spécifié par l'utilisateur ou le chemin de sortie par défaut |

Uniquement pour la génération C++ :

| Nom | Type | Description |
|---|---|---|
| $domtype | entier | 1 pour MSXML, 2 pour Xerces |
| $libtype | entier | 1 pour static LIB, 2 pour DLL |
| $mfc | booléenne | True si la prise en charge MFC est activée |
| $VSVersion | entier | Spécifie la version Visual Studio. Valeurs valides :<br>**0** Aucun projet Visual Studio<br>**2010** Visual Studio 2010<br>**2013** Visual Studio 2013 |

| Nom | Type | Description |
|---|---|---|
| | | **2015** Visual Studio 2015<br>**2017** Visual Studio 2017<br>**2019** Visual Studio 2019 |

Uniquement pour la génération C# :

| Nom | Type | Description |
|---|---|---|
| $VSVersion | entier | Spécifie la version Visual Studio. Valeurs valides :<br>**0** Aucun projet Visual Studio<br>**2010** Visual Studio 2010<br>**2013** Visual Studio 2013<br>**2015** Visual Studio 2015<br>**2017** Visual Studio 2017<br>**2019** Visual Studio 2019 |

## 15.2.7.5 Créer des fichiers de sortie

Ces instructions sont utilisées pour créer des fichiers de sortie depuis une génération de code. Veuillez noter que toutes ces instructions doivent se trouver dans un bloc délimité par des crochets.

### create

**create** *filename*

crée un nouveau fichier. Le fichier doit être fermé avec l'instruction **close**. Toutes les sorties suivantes sont écrites dans le fichier spécifié.

Exemple :

```
[create $outputpath & "/" & $JavaPackageDir & "/" & $application.Name & ".java"]
package [=$JavaPackageName];

public class [=$application.Name]Application {
...
}
[close]
```

### close

ferme le fichier de sortie actuel.

**=$***variable*

Writes écrit la valeur de la variable spécifiée dans le fichier de sortie actuel.

Exemple :

```
[$x = 20+3]
The result of your calculation is [=$x] - so have a nice day!
```

La fichier de sortie sera :

```
The result of your calculation is 23 - so have a nice day!
```

### write

**write** *string*

écrit le string dans le fichier de sortie actuel.

Exemple :

```
[write "C" & $name]
```

Il peut aussi s'écrire :

```
C[=$name]
```

### filecopy ... to ...

**filecopy** *source* **to** *target*

cope le fichier de source dans le fichier cible, sans interprétation.

Exemple :

```
filecopy "java/mapforce/mapforce.png" to $outputpath & "/" & $JavaPackageDir &
"/mapforce.png"
```

## 15.2.7.6 Opérateurs

Les opérateurs dans SPL fonctionnent comme dans la plupart des langages de programmation.

Liste des opérateurs SPL dans un ordre de précédence décroissant :

| | |
|---|---|
| . | Accéder à la propriété d'objet |
| ( ) | Regroupement d'expression |
| true | constante booléenne "true" |
| false | constante booléenne "false" |
| & | Concaténation de string |
| - | Signe pour le nombre négatif |
| not | Négation logique |
| * | Multiplier |
| / | Diviser |
| % | Modulo |
| + | Ajouter |
| - | Soustraire |
| <= | Inférieur à ou égal |
| < | Inférieur à |
| >= | Supérieur à ou égal |
| > | Supérieur à |
| = | Égal |
| <> | Non égal |
| and | Conjonction logique (avec une évaluation de court circuit) |
| or | Disjonction logique (avec une évaluation de court circuit) |
| = | Attribution |

## 15.2.7.7 Conditions

SPL vous permet d'utiliser les instructions "if" standard. La syntaxe est comme suit :

```
if condition
   statements
else
   statements
endif
```

ou, sans else :

```
if condition
   statements
endif
```

**Note:** La condition n'est pas contenue dans des parenthèses.

Comme dans tout autre langage de programmation, les conditions sont construites avec des opérateurs[1051] de logique et comparaison.

Exemple :

```
[if $namespace.ContainsPublicClasses and $namespace.Prefix <> ""]
   whatever you want ['inserts whatever you want, in the resulting file]
[endif]
```

## Switch

SPL contient aussi une instruction à choix multiple.

Syntaxe :

```
switch $variable
   case X:
      statements
   case Y:
   case Z:
      statements
   default:
      statements
endswitch
```

Les libellés de cas doivent être des constantes ou des variables.

L'instruction switch dans SPL ne tombe pas à travers les cas (comme dans C), il n'y a donc pas de nécessité pour une instruction de "break".

## 15.2.7.8 Collections et foreach

### Collections et itérateurs

Une collection contient plusieurs objets - comme un array ordinaire. Les itérateurs résolvent le problème du stockage et l'incrémentation des indexes array lorsque vous accédez aux objets.

Syntaxe :

```
foreach iterator in collection
   statements
next
```

Exemple :

```
[foreach $class in $classes
   if not $class.IsInternal
      ]  class [=$class.Name];
[  endif
next]
```

Exemple 2 :

```
[foreach $i in 1 To 3
 Write "// Step " & $i & "\n"
 ' Do some work
next]
```

Dans la première ligne :

**$classes** est l' **objet global** 1048 de tous les types générés. Il s'agit d'une collection d'objets de classe uniques.

**Foreach** passe à travers tous les items dans $classes, et exécute le code suivant l'instruction, jusqu'à l'instruction **next**, pour chacun d'entre eux.

Dans chaque itération, **$class** est attribué à l'objet de classe suivant. Il vous suffit de travailler avec l'objet de classe au lieu d'utiliser classes[i]->Name() comme pour un travail en C++.

Tous les itérateurs de collection possèdent les propriétés supplémentaires suivantes:

| | |
|---|---|
| Index | L'index suivant, en commençant avec 0 |
| IsFirst | true si l'objet actuel est le premier de la collection (index est 0) |
| IsLast | true si l'objet actuel est le derner de la collection |
| Current | L'objet actuel (cela est implicite s'il n'est pas spécifié et peut être exclu) |

Exemple :

```
[foreach $enum in $facet.Enumeration
   if not $enum.IsFirst
      ], [
   endif
   ]"[=$enum.Value]"[
next]
```

## 15.2.7.9 Sous-routines

Le générateur de code prend en charge des sous-routines sous la forme de procédures ou de fonctions.

Fonctions :

- Passage de valeurs par valeur et par référence
- Paramètres locaux/globaux (local dans le cadre des sous-routines)
- Variables locales
- Invocation récursive (les sous-routines peuvent s'appeler elles-même)

### 15.2.7.9.1 Déclaration de sous-routine

#### Sous-routine

Exemple de syntaxe :

```
Sub SimpleSub()
       ... lines of code
 EndSub
```

- **Sub** est le mot-clé qui dénote la procédure.
- **SimpleSub** est le nom attribué à la sous-routine.
- Λα **parenthèse**Ronde peut contenir une liste de paramètre.
- Le bloc de code d'une sous-routine est lancé immédiatement après la parenthèse de fermeture de paramètre.
- **EndSub** dénote la fin du code de bloc.

**Note:** La **déclaration** récursive ou en cascade de la sous-routine n'est pas permise, donc une sous-routine peut ne pas contenir une autre sous-routine.

#### Paramètres

Les paramètres peuvent aussi être passés par des procédures utilisant la syntaxe suivante :

- Tous les paramètres doivent être des variables
- Les variables doivent être préfixées par le caractère **$**
- Les variables locales sont définies dans une sous-routine
- Les variables globales sont déclarées explicitement, en-dehors des sous-routines

- Plusieurs paramètres sont séparés par le caractère virgule "**,**" dans des parenthèses arrondies
- Des paramètres peuvent passer des valeurs

### Paramètres - des valeurs passées

Des paramètres peuvent être passées de deux manières, par des valeurs et par des références, en utilisant les mots-clés **ByVal** et **ByRef** respectivement.

Syntaxe :

```
' define sub CompleteSub()
[Sub CompleteSub( $param, ByVal $paramByValue, ByRef $paramByRef )
] ...
```

- **ByVal** spécifie que le paramètre est passé par valeur. Veuillez noter que la plupart des objets peut uniquement être passé par la référence.
- **ByRef** spécifie que le paramètre est passé par la référence. Il s'agit du défaut si ni ByVal ou ByRef est spécifié.

### La fonction retourne des valeurs

Pour retourner une valeur depuis une sous-routine, utiliser l'instruction **return** . Une telle fonction peut être appelée depuis l'intérieur d'une expression.

Exemple :

```
' define a function
[Sub MakeQualifiedName( ByVal $namespacePrefix, ByVal $localName )
if $namespacePrefix = ""
 return $localName
else
 return $namespacePrefix & ":" & $localName
endif
EndSub
]
```

## 15.2.7.9.2 Invocation de sous-routine

Utiliser**call** pour invoquer une sous-routine, suivi par le nom de procédure et les paramètres, le cas échéant.

```
Call SimpleSub()
```

ou

```
Call CompleteSub( "FirstParameter", $ParamByValue, $ParamByRef )
```

### Invocation de fonction

Pour invoquer une fonction (toute sous-routine qui contient une instruction **return**), il suffit d'utiliser son nom dans une expression. Ne pas utiliser l'instruction **call** pour appeler les fonctions. Exemple :

```
$QName = MakeQualifiedName($namespace, "entry")
```

## 15.2.7.9.3 Exemple de sous-routine

L'exemple suivant montre la déclaration de sous-routine et l'invocation.

```
[create $outputpath & $module & "output.txt"

' define sub SimpleSub()
Sub SimpleSub()
]SimpleSub() called
[endsub

' execute sub SimpleSub()
Call SimpleSub()

$ParamByValue       = "Original Value"
]ParamByValue       = [=$ParamByValue]
[$ParamByRef  = "Original Value"
]ParamByRef   = [=$ParamByRef]

' define sub CompleteSub()
[Sub CompleteSub( $param, ByVal $paramByValue, ByRef $paramByRef )
]CompleteSub called.
       param = [=$param]
       paramByValue = [=$paramByValue]
       paramByRef = [=$paramByRef]
[$ParamByRef = "Local Variable"
$paramByValue = "new value"
$paramByRef = "new value"
]      Set values inside Sub
[$ParamByRef = "Local Variable"
$paramByValue = "new value"
$paramByRef = "new value"
]CompleteSub finished.
[endsub

' run sub CompleteSub()
Call CompleteSub( "FirstParameter", $ParamByValue, $ParamByRef )
]
ParamByValue=[=$ParamByValue]
ParamByRef=[=$ParamByRef]
[
```

```
Fermer
]
```

## 15.2.7.10 Types intégrés

La section décrit les propriétés des types intégrés utilisés dans les variables prédéfinies[1048] qui décrivent le schéma parsé.

### 15.2.7.10.1 Bibliothèque

Cet objet représente la bibliothèque toute entière générée depuis le Schéma XML ou le DTD.

| Propriétés | Type | Description |
|---|---|---|
| SchemaNamespaces | Espace de noms[1057] collection | Espaces de noms dans cette bibliothèque |
| SchemaFilename | string | Nom du fichier XSD ou DTD à partir duquel cette bibliothèque est dérivée |
| SchemaType | entier | 1 pour DTD, 2 pour Schéma XML |
| Guid | string | Une ID unique globale |
| CodeName | string | Nom de bibliothèque généré (dérivé du nom de fichier de schéma) |

### 15.2.7.10.2 Espace de noms

Un objet d'espace de noms par espace de noms de Schéma XML est généré. Les composants de schéma qui ne se trouvent pas dans un espace de noms sont contenus dans un objet d'espace de noms spécial avec un NamespaceURI vide. Veuillez noter que pour DTD, les espaces de noms sont aussi dérivés des attributs dont les noms commencent avec "xmlns".

| Propriétés | Type | Description |
|---|---|---|
| CodeName | string | Nom pour le code généré (dérivé du préfixe) |
| LocalName | string | Préfixe d'espace de noms |
| NamespaceURI | string | URI d'espace de noms |
| Types | Type[1058] collection | Tous les types contenus dans cet espace de noms |
| Library | Bibliothèque[1057] | Bibliothèque contenant cet espace de noms |

### 15.2.7.10.3 Type

Cet objet représente un type simple ou complexe. Il est utilisé pour générer une classe dans le langage cible. Il existe un type supplémentaire par bibliothèque qui représente le document, qui a tous les éléments racine possibles en tant que membres.

Des types anonymes ont un LocalName vide.

| Propriétés | Type | Description |
|---|---|---|
| CodeName | string | Nom pour le code généré (dérivé depuis le nom local ou la déclaration de parent) |
| LocalName | string | Nom original dans le schéma |
| Namespace | Namespace 1057 | Espace de noms contenant ce type |
| Attributes | Membre 1059 collection | Attributs contenus dans ce type* |
| Elements | Membre 1059 collection | Élément enfant contenu dans ce type |
| IsSimpleType | booléen | True pour les types ssimples, false pour les types complexes |
| IsDerived | booléen | True si ce type est dérivé d'un autre type, qui est aussi représenté par un objet Type |
| IsDerivedByExtension | booléen | True si ce type est dérivé par extension |
| IsDerivedByRestriction | booléen | True si ce type est dérivé par restriction |
| IsDerivedByUnion | booléen | True si ce type est dérivé par union |
| IsDerivedByList | booléen | True si ce type est dérivé par list |
| BaseType | Type | Le type de base de ce type (si IsDerived est true) |
| IsDocumentRootType | booléen | True si ce type représente le document lui-même |
| Library | Library 1057 | Bibliothèque contenant ce type |
| IsFinal | booléen | True si déclaré en tant que final dans le schéma |
| IsMixed | booléen | True si ce type peut avoir du contenu mixte |
| IsAbstract | booléen | True si ce type est déclaré en tant qu'abstrait |
| IsGlobal | booléen | True si ce type est déclaré globalement dans le schéma |
| IsAnonymous | booléen | True si ce type est déclaré localement dans un élément |

Pour les types simples uniquement :

| Propriétés | Type | Description |
|---|---|---|
| IsNativeBound | booléen | True si la liaison de type native existe |
| NativeBinding | NativeBinding 1060 | Liaison native pour ce type |
| Facettes | Facettes 1060 | Facettes de ce type |
| Whitespace | string | Raccourci vers la facette Whitespace |

* Les types complexes avec du contenu de texte (il s'agit des types avec du contenu mixte et des complexType avec simpleContent) ont un membre d'attribut non nommé supplémentaire qui représente le contenu de texte.

### 15.2.7.10.4 Membre

Cet objet représente un attribut ou un élément dans le Schéma XML. Il est utilisé pour créer des memnres de classe de types.

| Propriétés | Type | Description |
|---|---|---|
| CodeName | string | Nom pour le code généré (dérivé depuis le nom local ou la déclaration de parent) |
| LocalName | string | Nom original dans le schéma. Vide pour le membre spécial représentant le contenu de texte des types complexes. |
| NamespaceURI | string | L'URI d'espace de noms de cet Élément/Attribut dans le cadre du documents/streams de l'instance XML. |
| DeclaringType | Type 1058 | Type déclarant le membre à l'origine (égal au ContainingType pour les membres non hérités) |
| ContainingType | Type 1058 | Type où cela est un membre de |
| DataType | Type 1058 | Type de données du contenu de ce membre |
| Bibliothèque | Bibliothèque 1057 | Bibliothèque contenant le DataType de ce membre |
| IsAttribute | booléenne | Vrai pour les attributs, faux pour les éléments |
| IsOptional | booléenne | Vrai si minOccurs = 0 ou attribut optionnel |
| IsRequired | booléenne | Vrai si minOccurs > 0 ou attribut requis |

| Propriétés | Type | Description |
|---|---|---|
| IsFixed | booléenne | Vrai pour des attributs fixes, la valeur se trouve dans la propriété Default |
| IsDefault | booléenne | Vrai pour des attributs par défaut, la valeur se trouve dans la propriété Default |
| IsNillable | booléenne | Vrai pour des éléments nillables |
| IsUseQualified | booléenne | Vrai si NamespaceURI n'est pas vide |
| MinOccurs | entier | minOccurs, comme dans le schéma. 1 pour des attributs requis |
| MaxOccurs | entier | maxOccurs, comme dans le schéma. 0 pour les attributs interdits, -1 pour non lié |
| Default | string | Valeur par défaut |

### 15.2.7.10.5 NativeBinding

Cet objet représente la liaison d'un type simple vers un type natif dans le langage de programmation cible, comme spécifié par le mappage "schemanativetype".

| Propriétés | Type | Description |
|---|---|---|
| ValueType | string | Type natif |
| ValueHandler | string | Instance de classe de formateur |

### 15.2.7.10.6 Facettes

Cet objet représente toutes les facettes d'un type simple. Les facettes héritées sont fusionnées avec les facettes déclarées explicitement. Si une facette Length est en effet, MinLength et MaxLength sont définies pour la même valeur.

| Propriétés | Type | Description |
|---|---|---|
| DeclaringType | Type | Les facettes de type sont déclarées sur |
| Whitespace | string | "preserve", "collapse" ou "replace" |
| MinLength | entier | Valeur de Facette |
| MaxLength | entier | Valeur de Facette |
| MinInclusive | entier | Valeur de Facette |
| MinExclusive | entier | Valeur de Facette |

| Propriétés | Type | Description |
|---|---|---|
| MaxInclusive | entier | Valeur de Facette |
| MaxExclusive | entier | Valeur de Facette |
| TotalDigits | entier | Valeur de Facette |
| FractionDigits | entier | Valeur de Facette |
| List | Collection de facette | Toutes les facettes en tant que liste |

## Facette

Cet objet représente une seule facette avec sa valeur calculée effective pour un type spécifique.

| Propriétés | Type | Description |
|---|---|---|
| LocalName | string | Nom de facette |
| NamespaceURI | string | Espace de noms de facette |
| FacetType | string | Une de "normalization", "lexicalspace", "valuespace-length", "valuespace-enum" ou "valuespace-range" |
| DeclaringType | Type 1058 | Type sur lequel cette facette est déclarée |
| FacetCheckerName | string | Nom du vérificateur de facette (depuis le map schemafacet) |
| FacetValue | string ou entier | Valeur réelle de cette facette |

# 16 Commandes de menu

Cette section décrit les commandes de menu de MapForce. Les commandes de menu suivantes sont disponibles :

# 16.1 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`). Pour plus de détails, voir Projets 82..

- Ouvrir

  Ouvre le design de mappage précédemment ouvert (`.mfd`). Dans les éditions Professional et Enterprise, vous pouvez aussi ouvrir un projet de mappage (`.mfp`). Pour plus de détails, voir Projets 82..

- 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

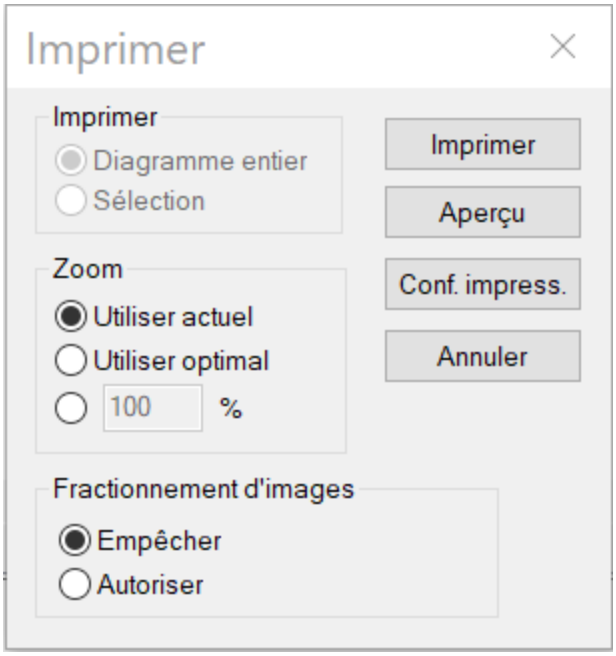
  Recharger le mappage actuel actif rétablit vos derniers changements.

- 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[68].

- Paramètres de mappage

  Ouvre la boîte de dialogue des paramètres de mappage[79] 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[79].

Pour plus d'information sur les langages de transformation disponible, voir Langages de transformation[22]. Pour plus d'information sur le code généré, voir le Générateur de Code[70].

- 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. Pour les détails, voir Compiler un mappage de MapForce[864].

- Déployer sur FlowForce Server (*éditions Professional et Enterprise*)

  Déployez le mappage actuellement actif sur FlowForce Server. Pour les détails, voir Déployer un mappage de MapForce[867].

- Générer documentation (*éditions Professional et Enterprise*)

  Génère la documentation de vos projets de mappage de manière très détaillée dans plusieurs formats de sortie. Pour plus d'informations, voir Générer de la documentation de mappage[823].

- 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.

# 16.2 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**.

# 16.3 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[123]. 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. voir Bases de données[158]. 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. Pour plus d'information, voir Fichiers CSV et Texte[373]. 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 398 et Paramètres dans des UDF 497. 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 409 et Paramètres dans des UDF 497. 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 413, 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  . Pour des détails, voir Joindre des données 426.

- Trier: Nœuds/Lignes

  Insère un composant qui vous permet de trier des nœuds (voir Trier Nœuds/Lignes 438). 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 444. 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. Pour plus d'information, voir (voir Composant SQL WHERE / ORDER) 365. 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 451. 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 444. 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. Pour plus d'information sur les composants d'Exception, voir Exceptions 463.

- Commenter

  Cette commande de menu vous permet d'insérer des commentaires style pense-bête comme composants amovibles. Pour les détails, voir Commentaires 39. Vous pouvez également ajouter des commentaires en cliquant la commande de barre d'outils  .

## 16.4 Projet

MapForce vous permet de grouper vos mappages dans les projets de mappage[82]. Cette rubrique recense toutes les commandes de menu disponibles dans le menu **Projet**.

- Recharger projet

  Recharge le projet actuellement actif et passe à la fenêtre **Projet**.

- Fermer projet

  Ferme le projet actuellement actif.

- Enregistrer projet

  Enregistre le projet actuellement actif.

- Ajouter des fichiers au projet

  Vous permet d'ajouter des mappages au projet actuel.

- Ajouter fichier actif au projet

  Ajoute le fichier actuellement actif au projet actuellement ouvert.

- Créer dossier

  Cette option ajoute un nouveau dossier au projet actuel. Voir Dossiers de projet[86].

- Convertir liens Web

  Cette commande vous permet de créer un lien vers une ressource Web externe et d'ajouter ce lien dans votre projet. Pour plus de détails, voir Projets[82].

- Ouvrir le mappage (*Enterprise Edition*)

  Ouvre le mappage actuellement marqué/sélectionné dans l'onglet **Projet**

- Créer le mappage pour l'opération (*Enterprise Edition*)

  Crée un fichier de mappage pour l'opération actuellement sélectionnée du projet WSDL.

- Ajouter un fichier de mappage pour l'opération (*Enterprise Edition*)

  Vous permet d'ajouter un fichier de mappage précédemment enregistré à l'opération WSDL actuellement active.

- Insérer Service Web (*Enterprise Edition*)

  Vous permet d'insérer un service Web sur la base d'un fichier WSDL existant.

- Ouvrir dans XMLSpy

  Ouvre le fichier WSDL sélectionné dans Altova XMLSpy.

- Générer le Code pour tous le projet

  Génère le code pour tout le projet actuellement visible dans la fenêtre de **Projet**. Le code est généré dans le langage sélectionné pour tous les fichiers de mappage **.mfd** dans chacun des dossiers.

- Générer code dans

  Génère le code de projet dans le langage que vous avez sélectionné depuis le menu contextuel.

- Propriétés

  Ouvre un dialogue où vous pouvez définir les paramètres de projet [85].

- Projets récents

  Affiche une liste des projets ouverts récemment.

# 16.5 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. Voir Bases de données 158.

- 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 45.

- Supprimer doublon

  Supprime un item dupliqué.

- Instructions de traitement/Commentaire

  Cette option vous permet d'insérer des commentaires et instructions de traitement 134 dans les composants XML.

- Écrire du contenu en tant que section CDATA

Cette commande crée une section CDATA[135] 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. Voir Paramètres d'actions de table de base de données[280] pour plus d'information.

- 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[39].

- Propriétés

  Affiche les paramètres d'un composant actuellement sélectionné. Voir Changer les paramètres de composant[44].

# 16.6 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 51.

- Paramètres pour connecter des enfants correspondants

  Vous aide à définir des connexions d'enfants correspondants. Pour les détails, voir Connexions d'enfants correspondants 57.

- 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 57.

- 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 55.

- 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 60.

- 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 55.

- 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 61.

# 16.7 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[491]. 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[494]. 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[492].

- 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[398] et Paramètres dans des UDF[497]. 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[409] et Paramètres dans des UDF[497]. Vous pouvez également Insérer un simple composant de sortie utilisant le bouton de la barre d'outils  .

## 16.8 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 22 . 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 68.

- 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 787. Voir le Tutoriel 4 113.

- 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. Pour en savoir plus sur toutes les bases de données prises en charge, voir Bases de données 158.

- 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 72 (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[72].

# 16.9 Déboguer

Cette rubrique recense toutes les commandes de menu disponibles dans le menu **Déboguer**.

- Commencer le débogage

  Lance ou continue le débogage jusqu'à la survenue d'un point d'arrêt ou la fin d'un mappage. Vous pouvez également lancer le débogage en utilisant la commande de la barre d'outils . Pour plus d'information sur le mode de débogage, voir Débogueur[833].

- Arrêter le débogage

  Stoppe le débogage. Cette commande permet de quitter le mode de débogage et fait repasser MapForce dans le mode standard. Vous pouvez également arrêter le débogage en utilisant la commande de la barre d'outils . Pour plus d'information sur le mode de débogage, voir Débogueur[833].

- Pas à pas détaillé

  Exécute le mappage jusqu'à ce qu'une étape unique soit terminée dans un endroit quelconque du mappage. Dans le débogueur du mappage, un pas est un groupe logique de calculs dépendants qui produisent généralement un seul item de séquence. Dépendant du contexte de mappage, cette commande vous dit environ ceci : *aller à gauche/aller à enfant cible/aller à parent source*. Vous pouvez également accéder à cette commande en utilisant le bouton de la barre d'outils . Pour plus d'information sur le mode de débogage, voir Débogueur[833].

- Pas à pas sortant

  Poursuit l'exécution jusqu'à la fin de l'étape actuelle (ou soit terminée à nouveau pour un autre item de la séquence), ou si une étape non liée s'achève. Cette commande passe par-dessus les calculs qui sont des entrées de l'étape actuelle. Vous pouvez également accéder à cette commande en utilisant le bouton de la barre d'outils . Pour plus d'information sur le mode de débogage, voir Débogueur[833].

- Pas à pas principal

  Poursuit l'exécution jusqu'à ce que le résultat de l'étape actuelle soit consommée ou qu'une étape soit exécutée qui n'est pas une entrée ou un enfant de la consommation. Cette commande sort du calcul actuel. Dépendant du contexte de mappage, cette commande vous dit environ ceci : *aller à droite/aller à parent cible/aller à enfant source*. Vous pouvez également accéder à cette commande en utilisant le bouton de la barre d'outils . Pour plus d'information sur le mode de débogage, voir Débogueur[833].

- Étape minimale

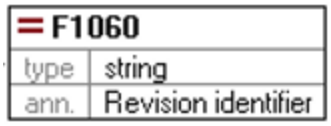
  Poursuit l'exécution jusqu'à ce qu'une valeur soit produite ou consommée. Cette commande subdivise une étape et s'arrêtera typiquement deux fois pour chaque connexion : (i) une fois lorsque sa source produit une valeur et (ii) une fois lorsque sa cible la consomme. MapForce ne calcule pas forcément toujours des valeurs dans l'ordre suggéré par le mappage, les événements de production et de consommation ne se suivent donc pas toujours. Vous pouvez également accéder à cette commande en utilisant le bouton de la barre d'outils . Pour plus d'information sur le mode de débogage, voir Débogueur[833].

# 16.10 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 63.

- 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 53.

- 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[30]. Lorsque le code est généré, la fenêtre **Messages** est automatiquement activée pour montrer le résultat de validation.

- Aperçu

  Active/désactive la fenêtre d'aperçu[29]. Faites glisser le rectangle pour naviguer à travers le mappage.

- Fenêtre de projet (*éditions Professional et Enterprise*)

  Active/désactive la fenêtre **Projet**. Pour en savoir plus sur les projets, voir Projets[82].

- 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**. Pour plus d'information, voir À propos Mode Déboguer[838].

# 16.11 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[881].

- 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[881].

- 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[54] : 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[141].

- 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[1082] et les raccourcis de clavier[1083].

- Restaurer les Barres d'outils et Fenêtres

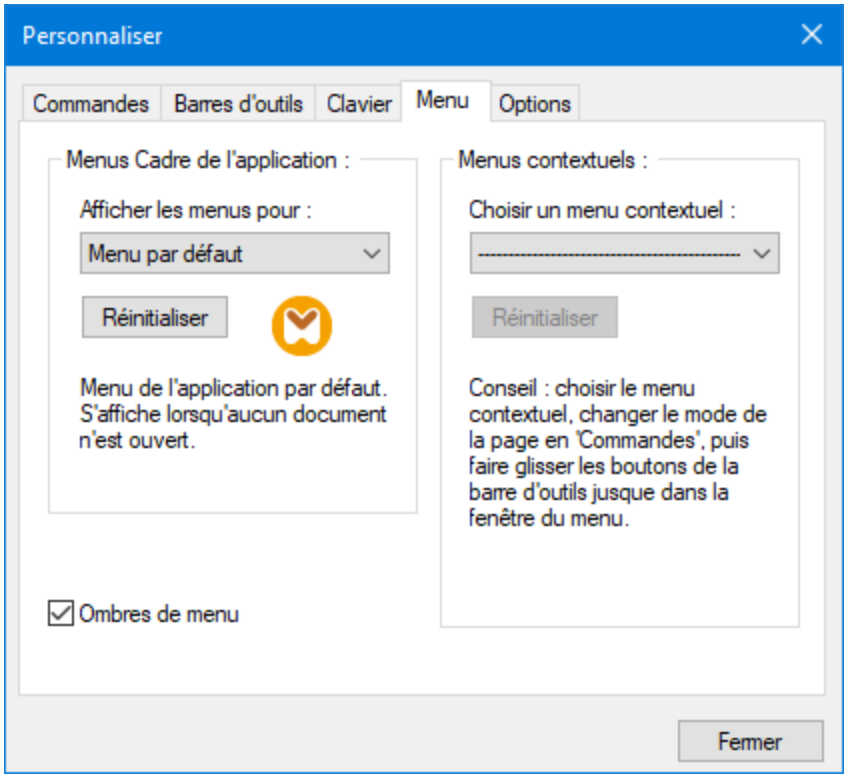
  Restaure les barres d'outils, les fenêtres d'assistants à la saisie, les fenêtres ancrées, etc. à leur affichage par défaut. Vous devez redémarrer MapForce pour que les changements prennent effet.

- Options

  Ouvre la boîte de dialogue **Options** qui vous permet de changer les paramètres par défaut de MapForce. Pour plus d'informations, voir Options 1086.

## 16.11.1 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.

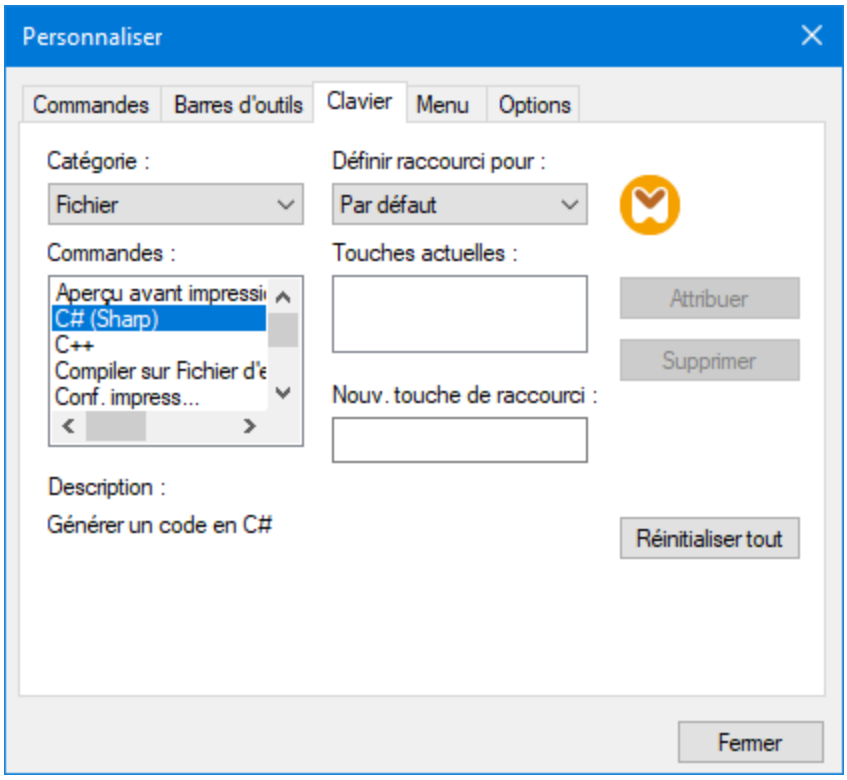
Vous pouvez réinitialiser un menu contextuel dans son état par défaut en le choisissant dans la liste déroulante *Sélectionner le menu contextuel* cliquer sur la touche **Réinitialiser**.

*Ombres de menu*
Sélectionnez la case à cocher *Ombres de menu* pour ajouter des ombres à tous les menus.

## 16.11.2 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 |

# 16.12 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[37]) 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[39].

- *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[44]).

  *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. Voir aussi Préparer des mappages pour l'exécution du serveur[858].

*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[68].

### 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[45].

- *Suppression de composant intelligent :* MapForce vous permet de garder les connexions même après avoir supprimé quelques composants de transformation[37]. 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[66].

### 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. Pour plus d'informations, voir Génération de code[70], Génération[1089], et Compiler des mappages dans les fichiers d'exécution de MapForce Server[864].

### 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[1091].

### 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. Pour les détails, voir Paramètres de requête de base de données 1092.

Proxy de réseau

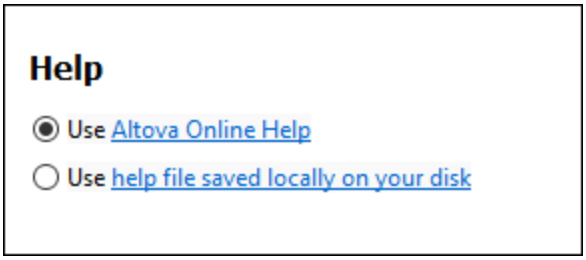
La section *Proxy de réseau* vous permet de configurer des paramètres de proxy personnalisés. Ces paramètres affectent la manière dont l'application se connecte à Internet. Par défaut, l'application utilise les paramètres proxy du système, dans la plupart des cas, vous n'aurez donc pas à changer les paramètres de proxy. Pour plus de détails, voir Proxy de réseau 1098.

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.

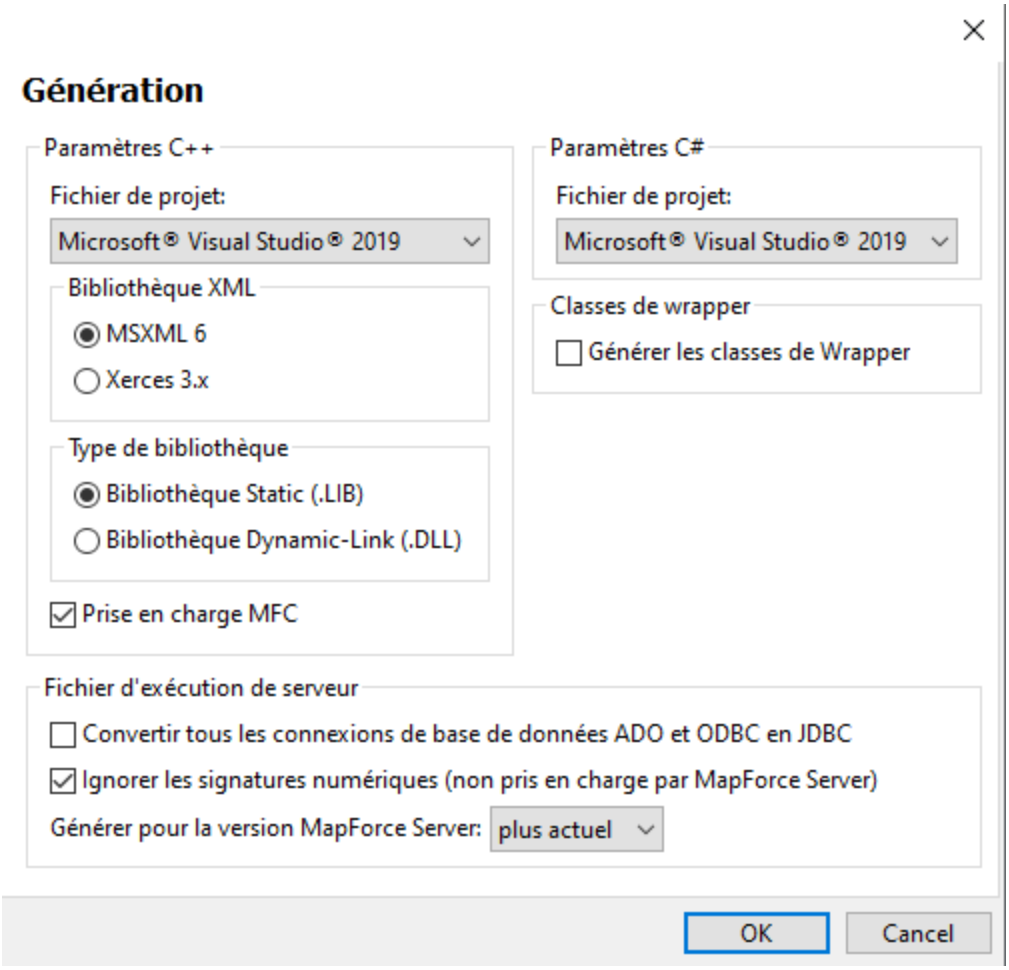
L'option Aide (*capture d'écran ci-dessous*) vous permet de sélectionner lequel des deux formats s'ouvre quand vous cliquez sur la commande **Aide (F1)** dans le menu **Aide**.




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.

## 16.12.1 Génération

La section *Génération* du dialogue des **Options** vous permet de configurer divers paramètres de génération de code et d'exécution du serveur (*voir ci-dessous*).




Les paramètres disponibles sont décrits ci-dessous.

- Paramètres C++

  Cette section définit les paramètres du compilateur suivants pour l'environnement C++,

  - La version Visual Studio (2013, 2015, 2017, 2019, 2022)
  - La bibliothèque XML (MSXML, Xerces 3.x)
  - Si les bibliothèques statiques ou dynamiques doivent être générées
  - Si le code doit être généré avec ou sans la prise en charge MFC

- Paramètres C#

  Si vous devez cibler la plateforme spécifique, sélectionnez l'option pertinente *Microsoft .NET <version>* depuis la liste déroulante. Si vous devez cibler la plateforme .NET Framework pour une version spécifique de Visual Studio, sélectionnez l'option pertinente *Microsoft Visual Studio <version>* .

- Classes Wrapper

  Vous permet de générer des classes wrapper pour les schémas XML. Pour des détails, voir Générer du code depuis des Schémas XML ou des DTD 953.

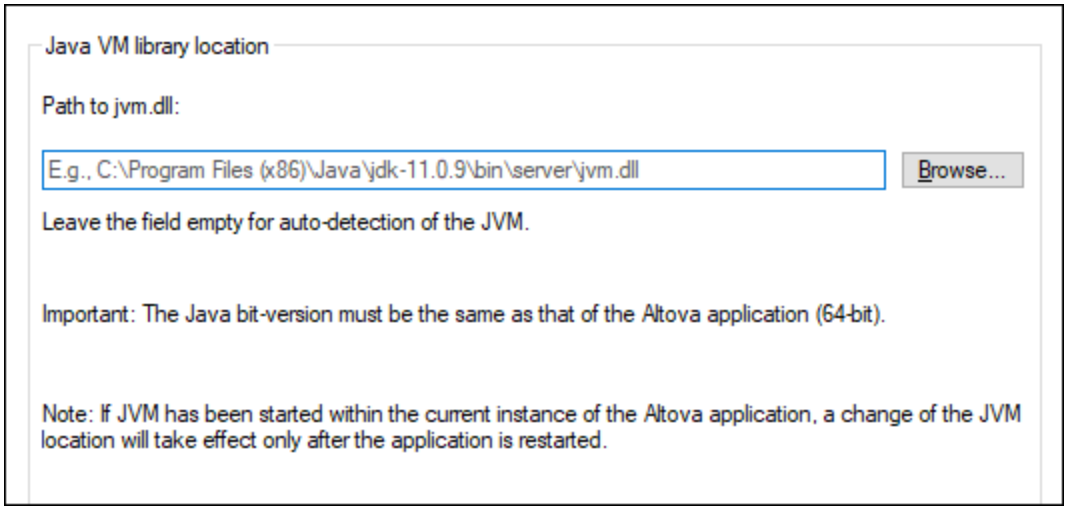
- Fichier d'exécution serveur

  Ces options sont pertinentes lorsque vous compilez des mappages dans les fichiers d'exécution MapForce Server. Pour plus d'information, voir Compiler des Mappages sur les fichiers d'exécution MapForce Server[864].

## 16.12.2 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.

La modification du chemin Java VM touche les zones suivantes :

- Connectivité JDBC
- Fonctions d'extension Java pour XSLT/XPath

Ce paramètre ne touche pas la génération du code Java.

## 16.12.3 Base de données

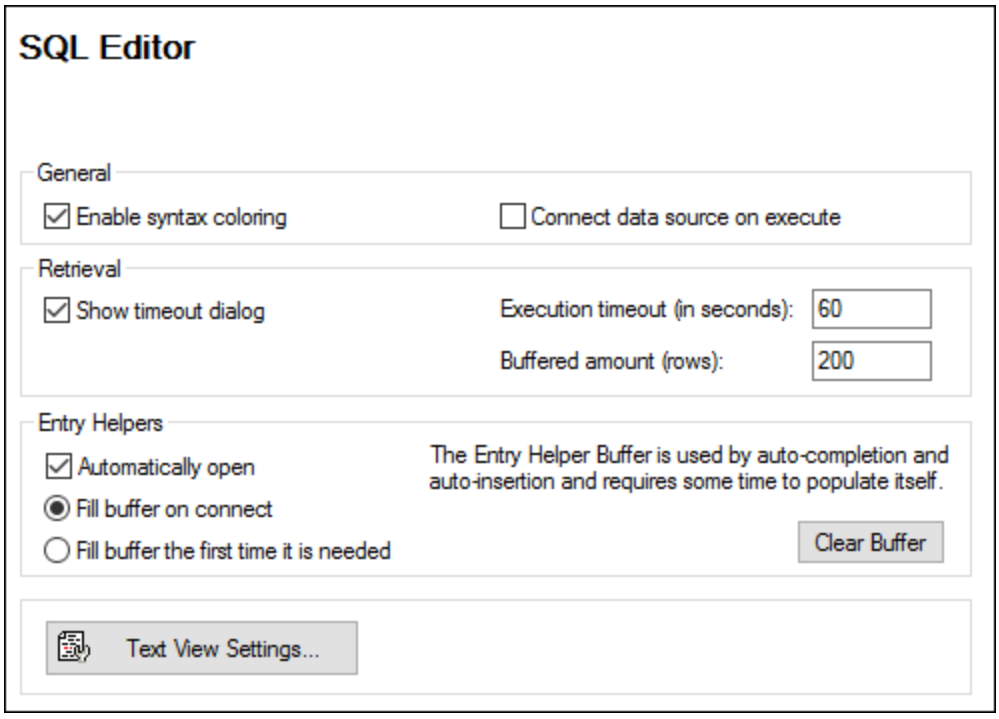
Cette section explique comment configurer les différents paramètres d'édition SQL. Vous pouvez accéder aux paramètres d'une des deux manières suivantes :

- En ouvrant le volet **DB Query** et en cliquant sur  (**Options**).
- En sélectionnant **Outils | Options | base de données**, puis la section pertinente.

Dans la section *Base de données*, vous pouvez définir les paramètres généraux d'édition SQL, les options du mode et de l'encodage des résultats, les paramètres de génération SQL et les polices. Pour les détails, voir les sous-sections ci-dessous.

### Éditeur SQL

La section *Éditeur SQL* vous permet de changer les paramètres généraux d'édition SQL (*voir la capture d'écran ci-dessous*). Les paramètres disponibles sont les suivants :




- *Général* Pour voir les différents éléments de la syntaxe SQL dans des couleurs distinctes, activez la coloration de la syntaxe. <anchor id="conndatasource"></anchor>Sélectionnez la case à cocher

*Connecter la source de données sur exécution* pour se connecter automatiquement à la source de données pertinente dès qu'une instruction SQL est exécutée.

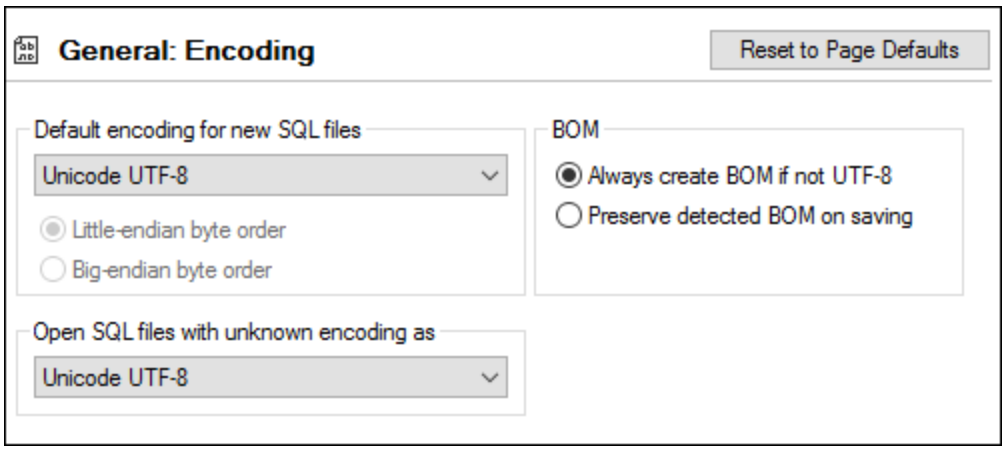
- *Extraction* Activer la case à cocher *Afficher dialogue délai d'expiration* pour vous permettre de changer les paramètres du délai d'expiration lorsque la période d'exécution autorisée est dépassée. Vous pouvez spécifier le temps maximum autorisé pour l'exécution SQL (*délai d'expiration*) en secondes. Vous pouvez aussi définir le nombre de lignes qui seront mises dans un buffer.

- *Assistants à la saisie* Pour permettre des suggestions de saisie semi-automatique dès que vous commencez à taper les instructions SQL, sélectionnez la case à cocher *ouvrir automatiquement* (voir aussi Saisie semi-automatique[256]). Vous pouvez choisir quand vous remplissez le buffer d'assistant à la saisie : quand vous vous connectez à la source de données ou quand le buffer est utilisé pour la première fois. Veuillez noter que le remplissage du tampon peut prendre un certain temps. Utiliser la touche **Supprimer la mémoire tampon** pour réinitialiser la mémoire tampon.

- *Paramètres du Mode Texte :* Vous permet de définir différents paramètres du mode Texte tels que les marges, onglets, aides visuelles, options de surbrillance automatique et raccourcis clavier de navigation en mode Texte. Pour plus d'informations, voir Fonctions de Mode Texte[72].

## Encodage

Dans la section *Encodage*, vous pouvez spécifier les options d'encodage pour les fichiers de l'Éditeur SQL créés ou ouverts dans l'Éditeur SQL (*voir la cpature d'écran ci-dessous*).




- *Encodage par défaut pour les nouveaux fichiers SQL :* Définir l'encodage par défaut pour les nouveaux fichiers de manière à ce que chaque nouveau document inclut la spécification d'encodage. 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 pour les fichiers SQL. L'encodage de fichiers existants n'est pas affecté par ce paramètre.

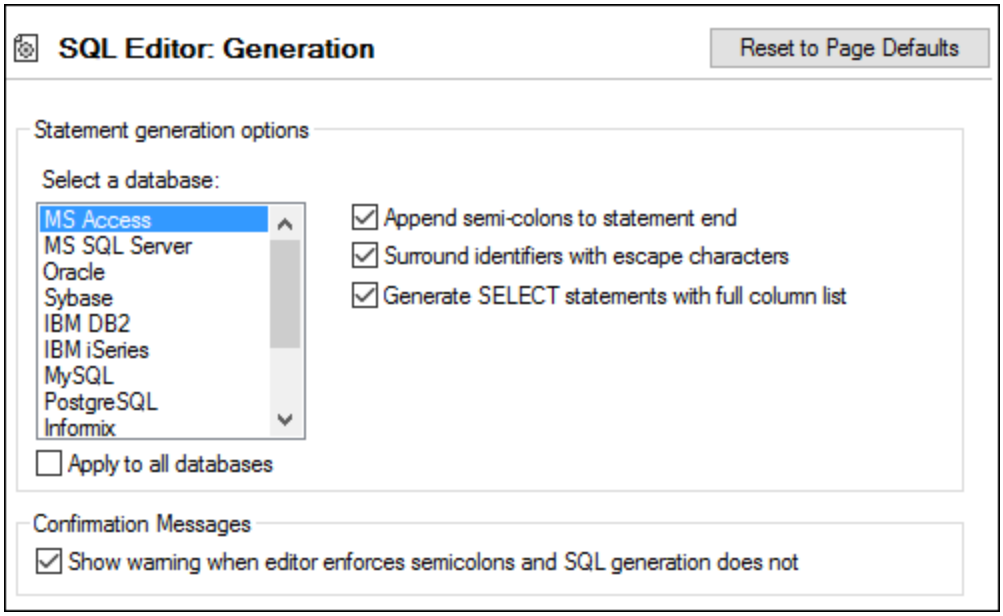
- *Ouvrir les fichiers SQL avec les encodages inconnus comme :* Vous pouvez choisir l'encodage avec lequel ouvrir un fichier SQL sans spécification d'encodage.

- *BOM:* Choisir si vous voulez créer le caractère byte order mark (BOM) pour l'encodage autre que UTF-8 ou s'il faut garder le BOM détecté.

**Note** : Les fichiers SQL qui n'ont pas de spécifications d'encodage sont enregistrés avec un encodage UTF-8.

## Géneration SQL

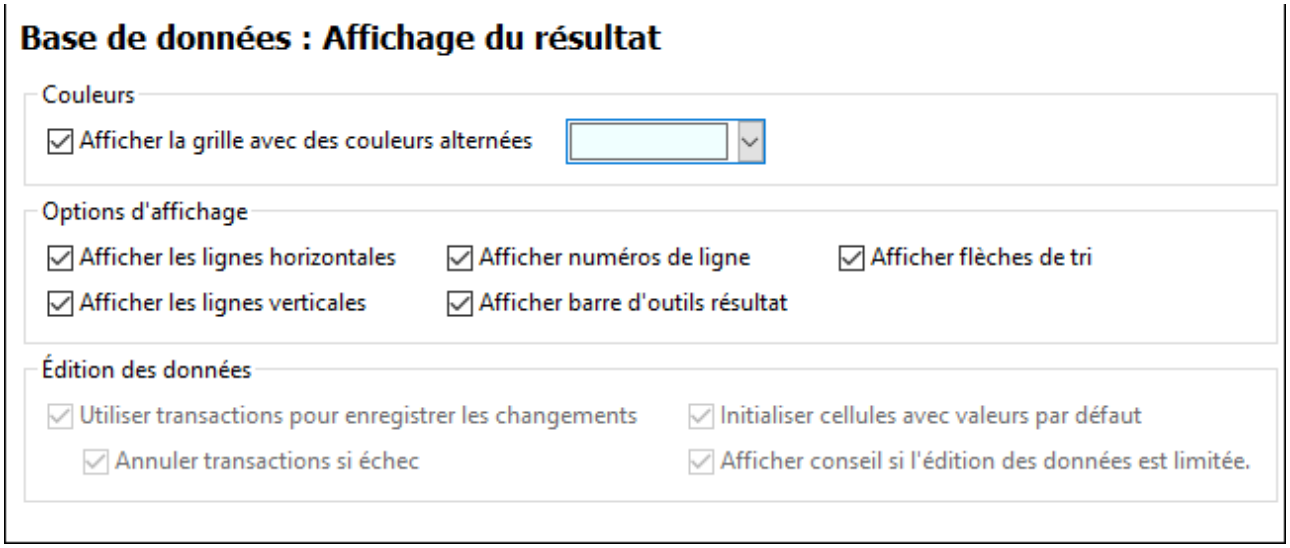
La section *Génération* vous permet de spécifier la syntaxe de génération d'instruction SQL pour les différents genres de base de données (*voir la capture d'écran ci-dessous*).




Afin de définir des préférences de syntaxe pour une base de données spécifique, en choisir une depuis la liste, puis activez ou désactivez les trois options à droite. Vous pouvez aussi choisir d'appliquer le même ensemble de paramètre à toutes les bases de données (la case à cocher *Appliquer à toutes les bases de données*). Veuillez noter que l'utilisation des paramètres communs pour toutes les bases de données peut empêcher l'édition des données dans les bases de données Oracle et IBM DB2 et iSeries par le biais d'une connexion JDBC.

## Affichage du Résultat

Vous pouvez configurer l'apparence de l'onglet **Résultats** du volet **DB Query** dans la section *Mode des Résultats* (*voir la capture d'écran ci-dessous*).

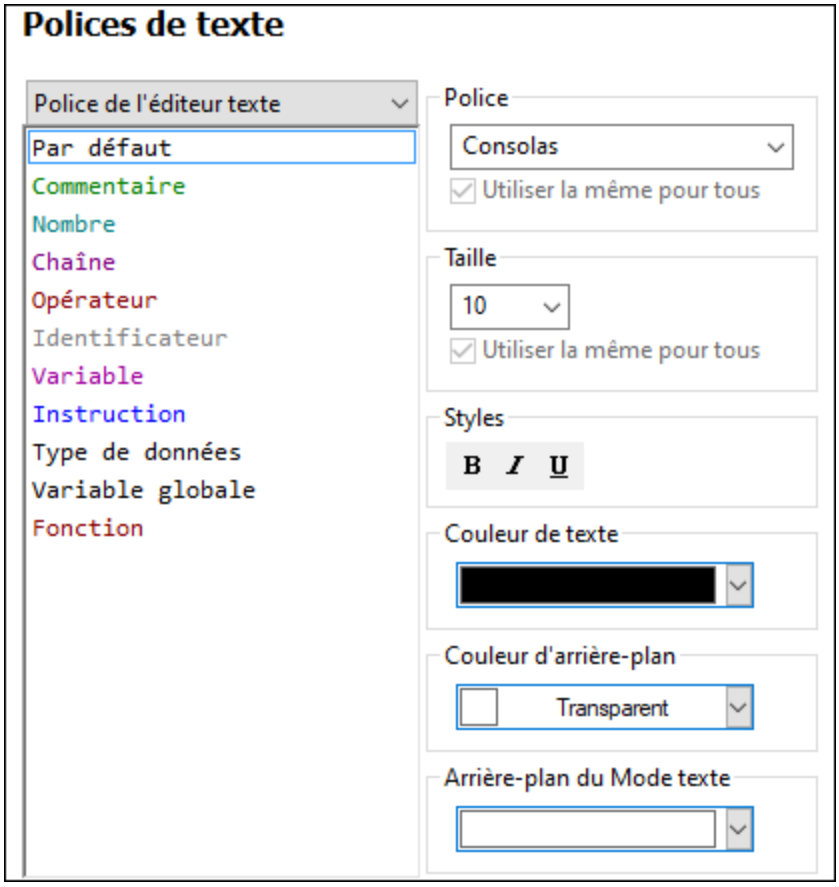



Sélectionnez la case à cocher *Afficher la grille avec des couleurs alternées* dans Afficher les lignes dans les onglets **Résultat** comme grille simple ou une grille avec des lignes alternantes blanches et colorées. Les couleurs alternatives sont configurables. Le groupe *Options d'affichage* vous permet de définir comment seront affichés des lignes de grille horizontales à verticales, les numéros de ligne et la barre d'outils **Résultat**.

Le groupe *Édition des données* vous permet de définir paramètres de transaction si les cellules doivent être remplies avec des valeurs par défaut, et si une indication doit être affichée lorsque l'édition des données est limitée.

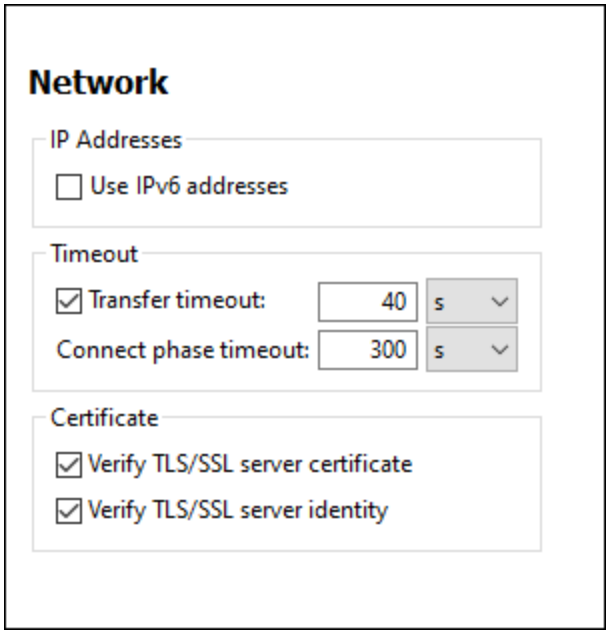
## Polices de texte

Dans la section *Polices de texte*, vous pouvez configurer les couleurs et paramètres de polices d'instructions SQL (*voir la capture d'écran ci-dessous*). Vous pouvez choisir le type de police, le style et la coloration syntaxique de travail les textes types qui apparaissent dans l'Éditeur SQL. Veuillez noter que la même police et la même taille sont utilisées pour tous les types de texte. Seul le style peut être modifié pour des types de texte individuels. Si vous accédez à la boîte de dialogue **Polices de texte** depuis le volet **DB Query**, vous pouvez cliquer sur le bouton **Reset to Page Defaults** pour restaurer les paramètres d'origine.




## 16.12.4 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.

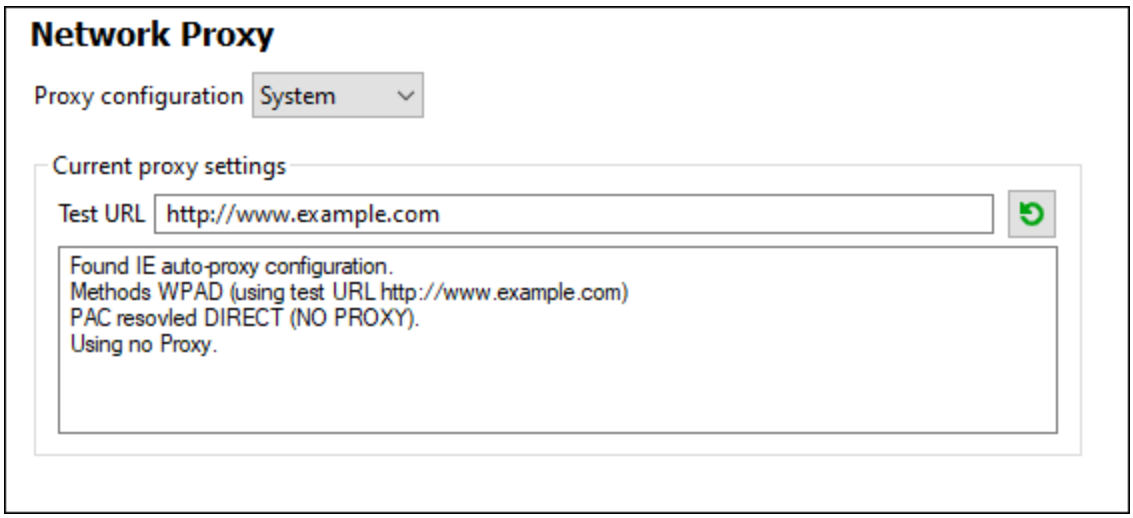
*Délai d'expiration*
- *Délai d'expiration du transfert :* Si cette limite est atteinte pour deux packs de données consécutifs d'un transfert (envoyés ou reçus), alors le transfert entier est abandonné. Les valeurs peuvent être spécifiées en secondes [s] ou millisecondes [ms], avec le défaut étant 40 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*).

## 16.12.5 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.

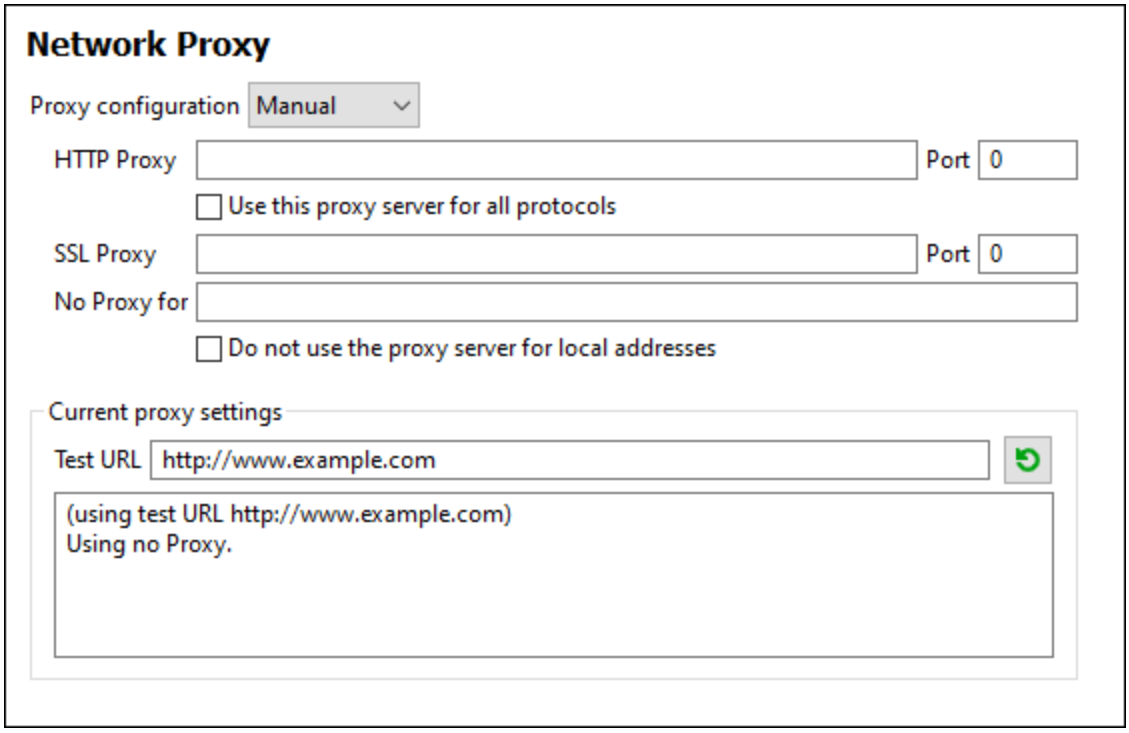



*Utiliser les paramètres de proxy de système*
Utilise les paramètres Internet Explorer (IE) configurables par le biais des paramètres de proxy du système. Effectue également une requête des paramètres configurés avec **`netsh.exe winhttp`**.

*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*).

# 16.13 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

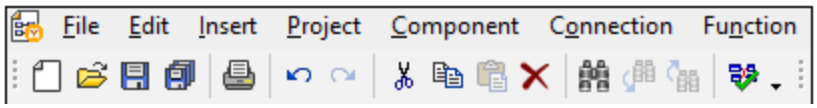
  Cette commande réarrange toutes les fenêtres de document ouvertes sous forme de mosaïque horizontale, les rendant toutes visibles en même temps.

- Mosaïque verticale

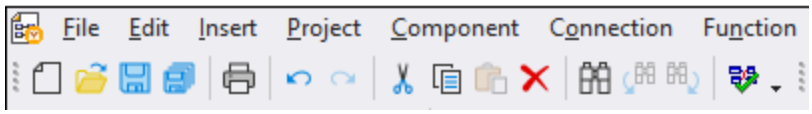
  Cette commande réarrange toutes les fenêtres de document ouvertes mosaïque verticale, les rendant toutes visibles en même temps.

- Thème classique/clair/sombre

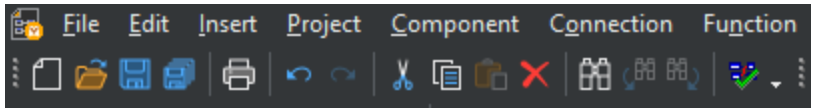
  MapForce vous permet de choisir des thèmes différents : *Classique*, *Clair* et *Sombre*. Les exemples de ces thèmes sont illustrés dans la capture d'écran ci-dessous. L'option par défaut est le thème classique.




*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.

# 16.14 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 :

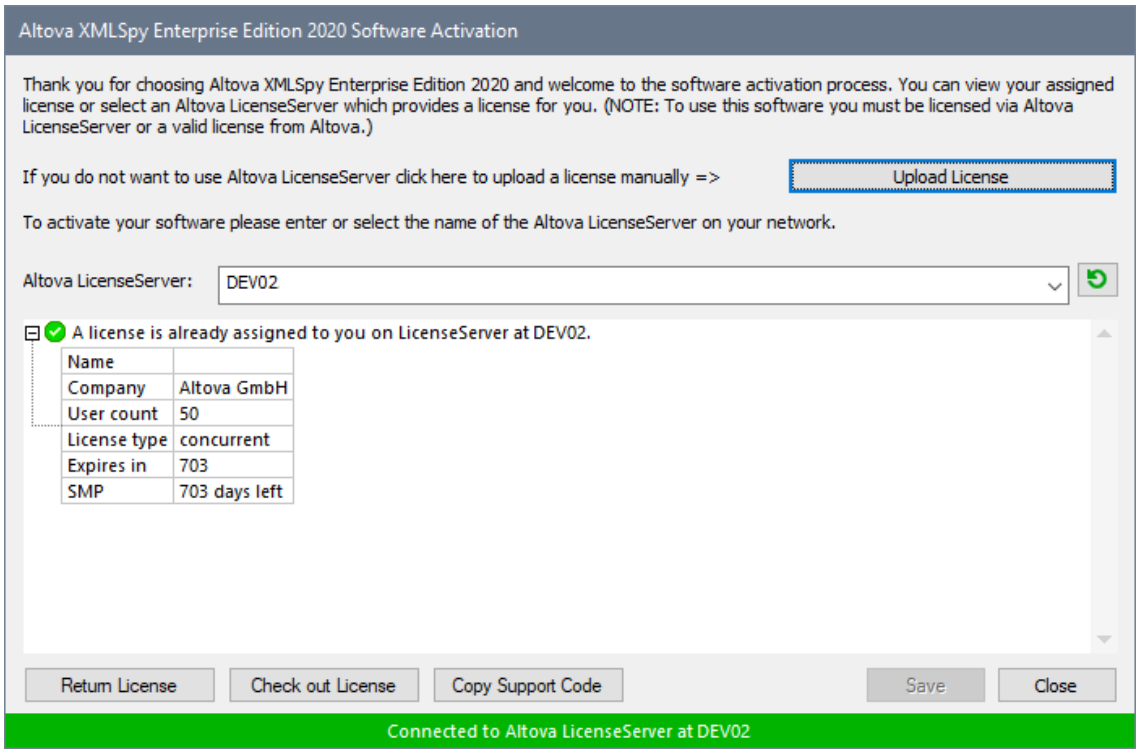
- Enregistrez le fchier de licence (**.altova_licenses**) vers un emplacement approprié, double-cliquez sur le fichier de licence, saisissez tout détail requis dans le dialogue qui apparaît, et terminez en cliquant sur **Appliquer clés**.
- Enregistrez le fichier de licence (**.altova_licenses**) vers 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.

# 17 L'API de MapForce

L'API basée sur COM permet aux clients d'accéder à la fonctionnalité de MapForce depuis un code personnalisé ou une application, et d'automatiser un large éventail de tâches.

L'API COM de MapForce suit les spécifications communes pour les serveurs Automation telles que définies par Microsoft. MapForce est automatiquement enregistré en tant qu'objet de serveur COM pendant l'installation. Une fois que l'objet de serveur COM est enregistré, vous pouvez l'invoquer depuis des applications et des langages de script qui prennent en charge des appels COM par programme. Ceci permet l'accès à l'API de MapForce non seulement depuis des environnements de développement utilisant .NET, C++ et Visual Basic, mais aussi depuis des langages de script comme JScript et VBScript.

Veuillez noter les points suivants :

- Si vous utilisez l'API de MapForce pour créer une application que vous voulez envoyer à d'autres clients, MapForce doit être installé sur chaque ordinateur client. Aussi, votre code d'intégration personnalisé doit être déployé vers (ou votre application installée sur) chaque ordinateur client.
- Certaines méthodes API, telles que `Document.GenerateOutput`, requièrent que la fenêtre principale de MapForce soit visible ou que MapForce (exécuté en tant que serveur COM) soit intégré à l'intérieur de l'interface utilisateur graphique. Si vous devez exécuter des mappages sans aucune assistance à travers divers plate-formes, envisagez l'utilisation de MapForce Server (https://www.altova.com/mapforce-server).

# 17.1 Accéder à l'API

Pour accéder à l'API COM de MapForce, une nouvelle instance de l'objet `Application` doit être créée dans votre application (ou script). Une fois que cet objet est créé, vous pouvez interagir avec MapForce en invoquant ses méthodes et propriétés telles que requises (par exemple, créer un nouveau document, ouvrir un document existant, générer un code de mappage, etc.).

## Exigences préalables

Pour rendre l'objet MapForce COM disponible dans votre projet de Visual Studio, ajoutez une référence au fichier MapForce type library (.tlb). Les instructions suivantes s'appliquent à Visual Studio 2013, mais sont semblables dans d'autres versions de Visual Studio :

1. Sur le menu **Projet**, cliquez sur **Ajouter référence**.
2. Cliquez sur **Parcourir** et sélectionnez le fichier **MapForce.tlb** situé dans le dossier d'installation MapForce.

Un exemple de client API de MapForce dans C# est disponible à : **C:\Users\<username>\Documents\Altova\MapForce2024\MapForceExamples\API\C#**.

Dans Java, l'API de MapForce est disponible par les bibliothèques pont Java-COM. Ces bibliothèques sont disponibles dans le dossier d'installation de MapForce : **C:\Program Files (x86)\Altova\MapForce2024\JavaAPI** (veuillez noter que ce chemin est valide quand MapForce 32-bit est exécuté sur Windows 64-bit, autrement, ajustez le chemin en conséquence).

- `AltovaAutomation.dll`: un wrapper JNI pour les serveurs d'automatisation d'Altova
- `AltovaAutomation.jar` : classes de Java pour accéder aux serveurs d'automatisation d'Altova
- `MapForceAPI.jar` : classes de Java qui wrappent l'interface d'automatisation de MapForce
- `MapForceAPI_JavaDoc.zip` : un fichier `Javadoc` contenant une documentation d'aide pour l'API de Java

Pour permettre l'accès au serveur d'automatisation de MapForce directement depuis le code Java, les bibliothèques ci-dessus doivent être dans la `classpath` de Java.

Un exemple de client API de MapForce dans Java est disponible sous : **C:\Users\<username>\Documents\Altova\MapForce2024\MapForceExamples\API\Java**.

Dans les langages de script tels que JScript ou VBScript, l'objet COM de MapForce est accessible par le Microsoft Windows Script Host (voir https://msdn.microsoft.com/en-us/library/9bbdkx3k.aspx). De tels scripts peuvent être écrits avec un éditeur de texte et n'ont pas besoin de compilation, puisqu'ils sont exécutés par Windows Script Host empaqueté avec Windows. (Pour vérifier si Windows Script Host est exécuté, saisissez `wscript.exe /?` à l'invite de commande). Plusieurs fichiers d'exemple JScript qui appellent l'API de MapForce sont disponibles sous : **C:\Users\<username>\Documents\Altova\MapForce2024\MapForceExamples\API\JScript**.

**Note :** Pour MapForce 32-bit, le nom enregistré ou l'identificateur programmatique (ProgId) de l'objet COM est `MapForce.Application`. Pour MapForce 64-bit, le nom est `MapForce_x64.Application`. Néanmoins, n'oubliez pas que le programme d'appel accédera aux entrée de registre CLASSES dans sa propre ruche de registre, ou groupe (32-bit ou 64-bit). C'est pourquoi, si vous exécutez des scripts en utilisant l'invite de commande standard et Windows Explorer sur Windows 64-bit, les entrées de registre 64-bit seront accédées, et pointeront vers le MapForce 64-bit. Pour cette raison, si les deux MapForce 32-bit

et 64-bit sont installés, une gestion spéciale est nécessaire afin de pouvoir appeler le MapForce 32-bit. Par exemple, si on part du principe que Windows Scripting Host est le programme d'appel, procéder comme suit :

1. Changer le répertoire actuel à **C:\Windows\SysWOW64**.
2. Dans la ligne de commande, saisir **wscript.exe** suivi du chemin vers le script que vous souhaitez exécuter, par exemple :

```
wscript.exe "C:\Users\...
\Documents\Altova\MapForce2024\MapForceExamples\API\JScript\start.js"
```

## Lignes directrices

Les lignes directrices suivantes devraient être prises en compte dans votre code client :

- Ne gardez pas de références aux objets en mémoire plus longtemps que vous en avez besoin. Si un utilisateur interagit entre deux appels de votre client, il n'y a pas de garantie que ces références sont toujours valides.
- Ayez conscience que si votre code client a un incident, les instances de MapForce demeurent tout de même dans le système. Pour les détails comment éviter les messages d'erreur, voir Gestion d'erreur 1112.
- Libérez les références explicitement si vous utilisez les langages tels que C++.

## Créer l'objet de l'application

La syntaxe pour créer l'objet d'`Application` de démarrage dépend du langage de programmation, tel qu'expliqué dans l'exemple ci-dessous :

*C#*

```
// Créez une nouvelle instance de MapForce par l'interface d'automatisation.
MapForceLib.Application objMapForce = new MapForceLib.Application();
```

*Java*

```
// Démarrez MapForce en tant que serveur COM.
com.altova.automation.MapForce.Application objMapForce = new Application();
// les serveurs COM démarrent de manière invisible, donc nous les rendons visibles
objMapForce.setVisible(true);
```

*JScript*

```
// Accédez à une instance en cours d'exécution, ou créez une nouvelle instance de
MapForce.
try
{
	objMapForce = WScript.GetObject ("", "MapForce.Application");
	// afficher l'application s'il s'agit d'une nouvelle instance
	objMapForce.Visible = true;
}
catch(err) { WScript.Echo ("Ne peut pas accéder à ou créer MapForce.Application"); }
```

*VBA*

```
' Créez une nouvelle instance de MapForce.
Dim objMapForce As Application
Set objMapForce = CreateObject("MapForce.Application")
```

*VBScript*

```
' Accédez à une instance en cours d'exécution, ou créez une nouvelle instance de
MapForce.
Set objMapForce = GetObject("MapForce.Application");
```

*Visual Basic*

```
Dim objMapForce As MapForceLib.Application = New MapForceLib.Application
```

## 17.2 Le modèle d'objet

Le point de départ de chaque application qui utilise l'API de MapForce est l'objet d' `Application`[1137] . Toutes les autres interfaces sont accédées à travers l'objet `d'application` en tant que point de départ.

Le modèle d'objet de l'API de MapForce peut être représenté comme suit (chaque niveau d'indentation indique une relation enfant-parent avec le niveau juste au-dessus) :

```
Application[1137]
   Options[1244]
   Projet[1252]
      ProjectItem[1268]
   Documents[1208]
      Document[1189]
         MapForceView[1227]
         Mappage[1234]
            Composant[1169]
               Datapoint[1187]
            Composants[1183]
            Composant[1185]
         Mappings[1243]
   ErrorMarkers[1216]
      ErrorMarker[1212]
AppOutputLines[1164]
   AppOutputLine[1154]
      AppOutputLines
         ...
      AppOutputLineSymbol[1166]
```

Pour toute information sur la création d'une instance d'objet d'`Application`, voir Accéder à l'API[1108]. Pour toute référence aux objets exposés par l'API, voir Référence d'objet[1137].

## 17.3 Gestion d'erreur

L'API de MapForce renvoie les erreurs de deux manières différentes. Chaque méthode API renvoie un HRESULT. Cette valeur de renvoi informe l'appelant sur les défaillances pendant l'exécution de la méthode. Si l'appel a réussi, la valeur de renvoi est égale à S_OK. Les programmeurs C/C++ utilisent généralement HRESULT pour détecter des erreurs.

Visual Basic, les langages de scripts, et d'autres environnements de développement de haut niveau ne donnent pas accès au HRESULT de renvoi d'un appel COM. Ils utilisent le deuxième mécanisme de communication de l'erreur pris en charge par l'API de MapForce, l'interface IErrorInfo. Si une erreur se produit, l'API crée un nouvel objet qui met en œuvre l'interface IErrorInfo. L'environnement de développement prend cette interface et remplit son propre mécanisme de communication d'erreur à base des informations fournies.

Le texte suivant décrit comment gérer les erreurs survenues depuis l'API de MapForce dans les différents environnements de développement.

### Visual Basic

Une manière commune de gérer des erreurs dans Visual Basic est de définir un gestionnaire d'erreurs. Ce gestionnaire d'erreur peut être défini avec la déclaration Sur Erreur. Normalement, le gestionnaire affiche un message d'erreur et nettoie un peu pour éviter des références détachées et tout genre de fuites de ressources. Visual Basic remplit son propre objet Err avec des informations de l'interface IErrorInfo.

```
Sub Validate()
'placer les déclarations de variable ici

'définir le gestionnaire d'erreurs
On Error GoTo ErrorHandler

'si la génération échoue, l'exécution de programme continue à ErrorHandler :
 objMapForce.ActiveDocument.GenerateXSLT()

'le code supplémentaire est mis ici

'quitter
Exit Sub

 ErrorHandler :
MsgBox("Error: " & (Err.Number - vbObjectError) & Chr(13) &
 "Description: " & Err.Description)
End Sub
```

### JavaScript

La mise en œuvre de Microsoft de JavaScript (JScript) fournit un mécanisme « try-catch » pour gérer les erreurs soulevées par les appels COM. Il est très semblable à l'approche de VisualBasic dans le sens que vous déclarez également un objet d'erreur contenant les informations nécessaires.

```
function Generate() {
// veuillez insérer les déclarations de variable ici

try {
 objMapForce.ActiveDocument.GenerateXSLT();
 }
catch (Error) {
 sError = Error.description;
 nErrorCode = Error.number & 0xffff;
return false;
 }

return true;
}
```

## C/C++

C/C++ vous donne accès facilement à `HRESULT` de l'appel COM et à `IErrorInterface`.

```
HRESULT hr;

// Call GenerateXSLT() de MapForce API
if(FAILED(hr = ipDocument->GenerateXSLT()))
{
IErrorInfo *ipErrorInfo = Null;

if(SUCCEEDED(::GetErrorInfo(0, &ipErrorInfo)))
{
BSTR bstrDescr;
ipErrorInfo->GetDescription(&bstrDescr);

// gérer l'information d'erreur
wprintf(L"Error message:\t%s\n",bstrDescr);
::SysFreeString(bstrDescr);

// publier info d'erreur
ipErrorInfo->Release();
}
}
```

## 17.4 Exemple Projet C#

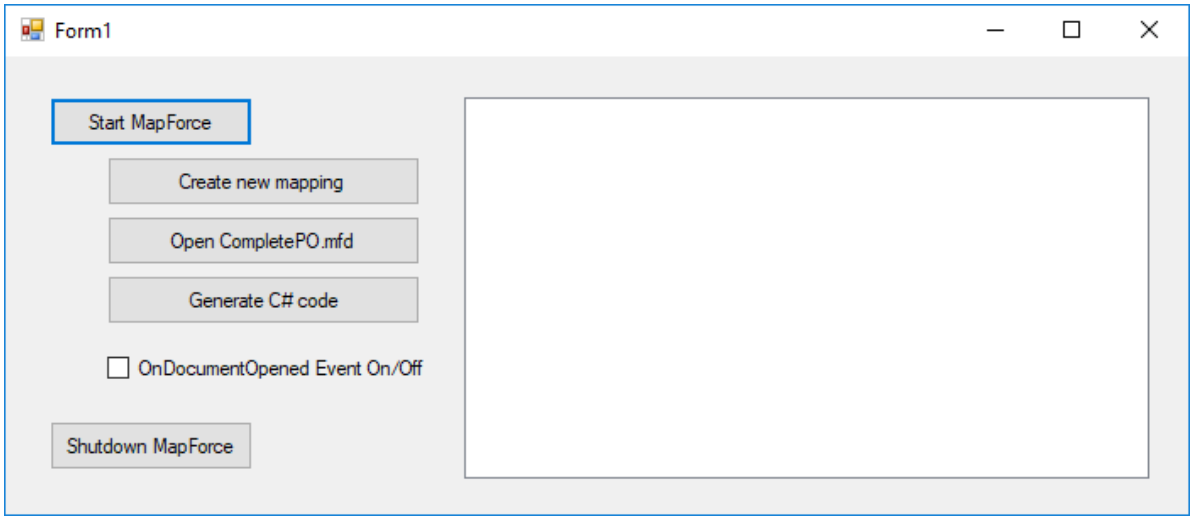
Après avoir installé MapForce, un exemple de projet client de MapForce API pour C# est disponible dans le répertoire **C:\Users\<username>\Documents\Altova\MapForce2024\MapForceExamples\API**.

Pour compiler et exécuter l'exemple, ouvrez la solution .sln file dans Visual Studio et exécutez **Déboguer | Démarrer débogage** ou **Déboguer | Démarrer sans débogage**.

**Note :** Si vous avez un système d'exploitation 64-bit et que vous utilisez une installation 32-bit installation de MapForce, ajoutez la plate-forme **x86** dans le Manager de configuration de la solution et générez l'exemple en utilisant cette configuration. Une nouvelle plateforme x86 (pour la solution active dans Visual Studio) peut être créée dans le nouveau dialogue de la plateforme de solution (**Build | Manager de configuration |plateforme de solution active | <New...>**).

Lorsque vous exécutez un exemple, un formulaire Windows est affiché contenant des boutons invoquant des opérations de base de MapForce :

- Démarrer MapForce
- Créer un nouveau design de mappage
- Ouvrez le fichier CompletePO.mfd file du dossier **...\MapForceExamples** (veuillez noter que vous allez éventuellement devoir ajuster le chemin pour indiquer le dossier **\MapForceExamples** sur votre appareil)
- Générez le code C# dans un répertoire temp
- Arrêter MapForce




### Liste de codes

Cette liste est dotée de commentaires pour faciliter la compréhension. Le code consiste essentiellement en une série de gestionnaires pour les boutons dans l'interface utilisateur affichée ci-dessus.

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
```

```
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace WindowsFormsApplication2
{
public partial class Form1 : Form
{
public Form1()
{
InitializeComponent();
}

// Une instance de MapForce accédée par le biais de son interface d'automatisation.
MapForceLib.Application MapForce;

// Emplacement d'exemples installés avec MapForce
String strExamplesFolder;

private void Form1_Load(object sender, EventArgs e)
{
}

// gestionnaire pour le bouton « Démarrer MapForce »
private void StartMapForce_Click(object sender, EventArgs e)
{
if (MapForce == null)
{
Cursor.Current = Cursors.WaitCursor;

// si nous n'avons pas d'instance de MapForce, nous en créons une et la rendons visible.
MapForce = new MapForceLib.Application();
MapForce.Visible = true;

// localiser des exemples installés avec MapForce.
int majorVersionYear = MapForce.MajorVersion + 1998;
strExamplesFolder = Environment.GetEnvironmentVariable("USERPROFILE") + "\\My Documents\
\Altova\\MapForce" + Convert.ToString(majorVersionYear) + "\\MapForceExamples\\";

Cursor.Current = Cursors.Default;
}
else
{
// si nous avons déjà une instance de MapForce en cours d'exécution, nous changeons son
indicateur de visibilité.
MapForce.Visible = !MapForce.Visible;
}
}

// gestionnaire pour le bouton « Open CompletePO.mfd »
private void openCompletePO_Click(object sender, EventArgs e)
{
```

```
if (MapForce == null)
StartMapForce_Click(null, null);

// Ouvrir un des exemples de fichiers installés avec le produit.
MapForce.OpenDocument(strExamplesFolder + "CompletePO.mfd");
}

// gestionnaire pour le bouton « Créer un nouveau mappage »
private void newMapping_Click(object sender, EventArgs e)
{
if (MapForce == null)
StartMapForce_Click(null, null);

// Créer un nouveau mappage
MapForce.NewMapping();
}

// gestionnaire pour le bouton « Arrêter MapForce »
// arrêter l'instance d'application en libérant l'objet COM.
private void shutdownMapForce_Click(object sender, EventArgs e)
{
if (MapForce != null)
{
// permettre l'arrêt de MapForce en libérant l'IU
MapForce.Visible = false;

// explicitement libérer l'objet COM
try
{
while (System.Runtime.InteropServices.Marshal.ReleaseComObject(MapForce) > 0) ;
}
finally
{
// éviter un accès plus tardif de cet objet.
MapForce = null;
}
}
}

// gestionnaire pour le bouton « Générer le code C# »
private void generateCppCode_Click(object sender, EventArgs e)
{
if (MapForce == null)
listBoxMessages.Items.Add("start MapForce first.");
// les erreurs COM sont renvoyées à C# comme exceptions. Nous utilisons un bloc try/catch
pour les gérer.
try
{
MapForceLib.Document doc = MapForce.ActiveDocument;

listBoxMessages.Items.Add("Active document " + doc.Name);
doc.GenerateCHashCode();
```

```
}
catch (Exception ex)
{
// L'appel COM n'a pas été couronné de succès.
// Probablement qu'aucune instance d'application n'a été lancée ou aucun document n'est
ouvert.
MessageBox.Show("COM error: " + ex.Message);
}
}

delegate void addListBoxItem_delegate(string sText);
// appeler du thread d'interface utilisateur
private void addListBoxItem(string sText)
{
listBoxMessages.Items.Add(sText);
}
// méthode wrapper pour permettre d'appeler les méthodes de contrôle IU depuis un thread
de travail
void syncWithUIthread(Control ctrl, addListBoxItem_delegate methodToInvoke, String sText)
{
// Control.Invoke: Exécute sur le thread IU, mais le thread appelant attend la saisie
avant de poursuivre.
// Control.BeginInvoke: Exécute sur le thread IU et le thread appelant n'attend pas
l'achèvement.
if (ctrl.InvokeRequired)
ctrl.BeginInvoke(methodToInvoke, new Object[] { sText });
}

// gestionnaire d'événement pour l'événement OnDocumentOpened
private void handleOnDocumentOpened(MapForceLib.Document i_ipDocument)
{
String sText = "";

if (i_ipDocument.Name.Length > 0)
sText = "Document " + i_ipDocument.Name + " was opened!";
else
sText = "A new mapping was created.";

// nous devons synchroniser le thread appelant avec le thread IU car
// les événements COM sont déclenchés depuis un thread de travail
addListBoxItem_delegate methodToInvoke = new addListBoxItem_delegate(addListBoxItem);
// appeler syncWithUIthread avec les arguments suivants :
// 1 - listBoxMessages - zone de liste pour afficher les messages depuis les événements
COM
// 2 - methodToInvoke - un délégué C# qui indique la méthode qui sera appelée depuis le
thread IU
// 3 - sText - le texte à afficher dans la zone de liste
syncWithUIthread(listBoxMessages, methodToInvoke, sText);
}

private void checkBoxEventOnOff_CheckedChanged(object sender, EventArgs e)
{
if (MapForce != null)
```

```
{
if (checkBoxEventOnOff.Checked)
MapForce.OnDocumentOpened += new
MapForceLib._IApplicationEvents_OnDocumentOpenedEventHandler(handleOnDocumentOpened);
else
MapForce.OnDocumentOpened -= new
MapForceLib._IApplicationEvents_OnDocumentOpenedEventHandler(handleOnDocumentOpened);
}
}
}
}
```

# 17.5 Exemple Projet Java

Une fois que vous avez installé MapForce, un exemple de projet client de MapForce API pour Java est disponible dans le répertoire **C:\Users\<username>\Documents\Altova\MapForce2024\MapForceExamples\API**.

Vous pouvez tester l'exemple de Java directement depuis la ligne de commande, utilisant le fichier de commandes `BuildAndRun.bat`, ou vous pouvez compiler et exécuter l'exemple de projet depuis l'intérieur d'Eclipse. Voir plus bas pour des instructions sur l'utilisation de ces procédures.

## Liste de fichier

Le dossier d'exemples de Java contient toutes les fichiers requis pour exécuter l'exemple de projet. Ces fichiers sont recensés ci-dessous :

| | |
|---|---|
| `AltovaAutomation.dll` | Java-COM bridge : DLL part |
| `AltovaAutomation.jar` | Java-COM bridge : Java library part |
| `MapForceAPI.jar` | Java classes of the MapForce API |
| `RunMapForce.java` | Code source de l'exemple Java |
| `BuildAndRun.bat` | Fichier de commandes pour compiler et exécuter un exemple de code depuis l'invite de la ligne de commande. Attend le dossier où Java Virtual Machine réside en tant que paramètre. |
| `.classpath` | Fichier d'assistance pour le projet Eclipse |
| `.project` | Fichier de projet Eclipse |
| `MapForceAPI_JavaDoc.zip` | Le fichier `Javadoc` contenant une documentation d'aide pour Java API |

## Ce que fait l'exemple

L'exemple démarre MapForce et effectue quelques opérations, y compris ouvrir et fermer les documents. Quand il a terminé, MapForce reste ouvert. Vous devez le fermer manuellement.

## Exécuter l'exemple depuis la ligne de commande

Pour exécuter l'exemple depuis la ligne de commande, ouvrez une fenêtre d'invite de commande, allez au dossier Java du dossier d'exemples API (*voir ci-dessus pour l'emplacement*), puis tapez :

```
buildAndRun.bat "<Path-to-the-Java-bin-folder>"
```

Le dossier binaire de Java doit être celui d'un kit de développement de Java (JDK) 7 ou d'une installation ultérieure sur votre ordinateur.

Appuyez sur la clé **Retour**. La source de Java dans `RunMapForce.java` sera compilée, puis exécutée.

## Charger l'exemple dans Eclipse

Ouvrez Eclipse et utilisez la commande **Fichier | Importer... Commande | Général | Projets existants dans flux de travail** pour ajouter un fichier de projet Eclipse (`.project`) situé dans le dossier Java du dossier d'exemples API (*voir au-dessus pour l'emplacement*). Le projet `RunMapForce` apparaîtra ensuite dans votre Package Explorer ou Navigator.

Sélectionnez le projet, puis la commande **Run as | Java Application** pour exécuter l'exemple.

**Note :** Vous pouvez sélectionner un nom de classe ou de méthode de l'API de Java et appuyez sur F1 pour recevoir de l'aide pour cette classe ou méthode.

## Liste de code source de Java

Le code source de Java dans l'exemple de fichier `RunMapForce.java` iest recensé ci-dessous avec des commentaires.

```
// access general JAVA-COM bridge classes
import java.util.Iterator;

import com.altova.automation.libs.*;

// access XMLSpy Java-COM bridge
import com.altova.automation.MapForce.*;
import com.altova.automation.MapForce.Enums.ENUMProgrammingLanguage;

/**
 * Un exemple simple qui démarre le serveur COM et réalise quelques opérations sur celui-
ci.
 * N'hésitez pas à élargir.
 */
public class RunMapForce
{
public static void main(String[] args)
{
// une instance de l'application.
Application mapforce = null;

// au lieu d'utiliser la gestion d'erreur COM, utilisez le mécanisme d'exception de Java.
try
{
// Démarrez MapForce en tant que serveur COM.
mapforce = new Application();
// les serveurs COM démarrent de manière invisible, donc nous les rendons visibles
mapforce.setVisible(true);

// Les lignes suivantes se joignent aux événements de l'application en utilisant une mise
en œuvre par défaut
// pour les événements et écrase l'une de ses méthodes.
// Si vous souhaitez écraser tous les événements de document, il est plus judicieux de
dériver votre classe d'écoute
// de DocumentEvents et met en œuvre toutes les méthodes de cette interface.
mapforce.addListener(new ApplicationEventsDefaultHandler()
{
@Override
public void onDocumentOpened(Document i_ipDoc) throws AutomationException
{
String name = i_ipDoc.getName();
```

```
if (name.length() > 0)
System.out.println("Document " + name + " was opened.");
else
System.out.println("A new mapping was created.");
}
});

// Localisez des exemples installés avec le produit.
int majorVersionYear = mapforce.getMajorVersion() + 1998;
String strExamplesFolder = System.getenv("USERPROFILE") + "\\Documents\\Altova\\MapForce"
 + Integer.toString(majorVersionYear) + "\\MapForceExamples\\";
// créez un nouveau mappage MapForce et générez le code c++
Document newDoc = mapforce.newMapping();
ErrorMarkers err1 = newDoc.generateCodeEx(ENUMProgrammingLanguage.eCpp);
afficher(err1);
// ouvrez CompletePO.mfd et générez le code c++
Document doc = mapforce.openDocument(strExamplesFolder + "CompletePO.mfd");
ErrorMarkers err2 = doc.generateCodeEx(ENUMProgrammingLanguage.eCpp);
afficher(err2);

doc.close();
doc = null;

System.out.println("Watch MapForce!");
}
catch (AutomationException e)
{
// e.printStackTrace();
}
finally
{
// Assurez-vous que MapForce peut être arrêté correctement.
if (mapforce != null)
mapforce.dispose();

// Puisque le serveur COM a été rendu visible et l'est encore, il poursuivra son
exécution
// et devra être arrêté manuellement.
System.out.println("Now close MapForce!");
}
}

public static void display(ErrorMarkers err) throws AutomationException
{
Iterator<ErrorMarker> itr = err.iterator();

if (err.getCount() == 0)
System.out.print("Code generation completed successfully.\n");

while (itr.hasNext())
{
String sError = "";
Object element = itr.next();
if (element instanceof ErrorMarker)
```

```
sError = ((ErrorMarker)element).getText();
System.out.print("Error text: " + sError + "\n");
}
}
}
```

## 17.6 Exemples JScript

Une fois que vous avez installé MapForce, plusieurs fichiers d'exemple JScript sont disponibles dans le répertoire **C:\Users\<username>\Documents\Altova\MapForce<%PRODYEAR%\MapForceExamples\API**.

Les exemples de fichiers peuvent être exécutés d'une des deux manières :

- *Depuis la ligne de commande :*
  Ouvrez une fenêtre d'invite de commandes et saisissez le nom d'un des exemples de scripts (par exemple, `Start.js`). L'Hôte de script Windows qui est empaqueté avec Windows exécutera le script.

- *Depuis Windows Explorer :*
  Dans Windows Explorer, parcourez pour le fichier JScript et double-cliquez sur celui-ci. L'Hôte de script Windows qui est empaqueté avec Windows exécutera le script. Après l'exécution du script, la console de commande se ferme automatiquement.

Les exemples de fichiers suivants sont inclus :

| | |
|---|---|
| `Start.js` | Démarrez Mapforce enregistré comme serveur d'automatisation ou connectez-vous à une instance d'exécution (voir Start Application 1123). |
| `DocumentAccess.js` | Montre comment ouvrir, itérer et fermer les documents (voir Accès simple aux documents 1124). |
| `GenerateCode.js` | Montre comment appeler la génération de code utilisant JScript (voir Générer le code 1125). |
| `Readme.txt` | Fournit l'aide basique pour exécuter les scripts. |

Cette documentation comprend en plus quelques listes de code JScript supplémentaires :

- Exemple : Génération de code 1127
- Exemple : Exécution de mappage 1129
- Exemple : Project Support 1133

### 17.6.1 Démarrer Application

La liste de code JScript ci-dessous démarre l'application et l'arrête. Si une instance de l'application est déjà en cours d'exécution, l'instance en cours sera renvoyée. Pour exécuter un script, démarrez-le depuis la fenêtre d'invite de commande ou depuis Windows Explorer, voir également Accéder à l'API 1108.

```
// Initialisez l'objet COM de l'application. Ceci démarrera une nouvelle instance de
l'application et
// renvoie son objet COM principal. Dépendant des paramètres COM, l'objet COM principal
d'une
// application déjà en cours peut être renvoyé.

try { objMapForce = WScript.GetObject("", "MapForce.Application"); }
```

```
catch(err) {}

if( typeof( objMapForce ) == "undefined" )
{
try { objMapForce = WScript.GetObject("", "MapForce_x64.Application") }
catch(err)
 {
 WScript.Echo( "Ne peut accéder à ou créer MapForce.Application" );
 WScript.Quit();
 }
}

// si nouvellement lancée, l'application démarrera sans son IU visible. Mettez-la à
visible.
objMapForce.Visible = true;

WScript.Echo(objMapForce.Edition + " a démarré avec succès. ");

objMapForce.Visible = false; // arrêtera l'application si elle n'a plus de connexions COM
//objMapForce.Visible = true; // maintiendra l'application en cours d'exécution avec l'IU
visible
```

## 17.6.2 Accès simple aux documents

La liste JScript ci-dessous montre comment ouvrir les documents, définir un document en tant que document actif, itérer à travers des documents ouverts et fermer des documents.

```
// Initialisez l'objet COM de l'application. Ceci démarrera une nouvelle instance de
l'application et
// renvoie son objet COM principal. Dépendant des paramètres COM, l'objet COM principal
d'une
// application déjà en cours peut être renvoyé.
try { objMapForce = WScript.GetObject("", "MapForce.Application"); }
catch(err) {}

if( typeof( objMapForce ) == "undefined" )
{
try { objMapForce = WScript.GetObject("", "MapForce_x64.Application") }
catch(err)
 {
 WScript.Echo( "Ne peut accéder à ou créer MapForce.Application" );
 WScript.Quit();
 }
}

// si nouvellement lancée, l'application démarrera sans son IU visible. Mettez-la à
visible.
objMapForce.Visible = true;

// ************************** code snippet pour "Simple Document Access"
**********************
```

```
// Localisez des exemples via la variable d'environnement USERPROFILE. Le chemin doit
être adapté aux versions majeures de release.
objWshShell = WScript.CreateObject("WScript.Shell");
majorVersionYear = objMapForce.MajorVersion + 1998
strExampleFolder = objWshShell.ExpandEnvironmentStrings("%USERPROFILE%") + "\\Documents\
\Altova\\MapForce" + majorVersionYear + "\\MapForceExamples\\";

objMapForce.Documents.OpenDocument(strExampleFolder + "CompletePO.mfd");
objMapForce.Documents.OpenDocument(strExampleFolder + "Altova_Hierarchical_DB.mfd");

// *************************** code snippet pour "Simple Document Access"
***********************

// *************************** code snippet pour "Iteration"
**********************************

// parcourez tous les documents ouverts utilisant un énumérateur JScript
pour (var iterDocs = new Enumerator(objMapForce.Documents); !iterDocs.atEnd();
iterDocs.moveNext())
{
 objName = iterDocs.item().Name;
 WScript.Echo("Nom du document : " + objName);
}

// parcourez tous les documents ouverts utilisant l'accès basé sur l'index pour la
collection des documents
pour (i = objMapForce.Documents.Count; i > 0; i--)
 objMapForce.Documents.Item(i).Close();


// *************************** code snippet pour "Iteration"
**********************************

//objMapForce.Visible = false; // arrêtera l'application si elle n'a plus de connexions
COM
objMapForce.Visible = true; // maintiendra l'application en cours avec l'IU visible
```

Le code recensé ci-dessus est disponible en tant qu'exemple de fichier (voir Exemples JScript[1123]). Pour exécuter le script, démarrez-le d'une fenêtre d'invite de commandes ou depuis Windows Explorer.

## 17.6.3 Générer le code

La liste JScript ci-dessous montre comment ouvrir des documents, définir un document en tant que document actif, itérer à travers les documents ouverts, et générez le code C++.

```
// Initialisez l'objet COM de l'application. Ceci démarrera une nouvelle instance de
l'application et
// renvoie son objet COM principal. Dépendant des paramètres COM, l'objet COM principal
d'une
// application déjà en cours peut être renvoyé.
try { objMapForce = WScript.GetObject("", "MapForce.Application"); }
```

```
catch(err) {}

if( typeof( objMapForce ) == "undefined" )
{
try { objMapForce = WScript.GetObject("", "MapForce_x64.Application") }
catch(err)
 {
 WScript.Echo( "Ne peut accéder à ou créer MapForce.Application" );
 WScript.Quit();
 }
}

// si nouvellement lancée, l'application démarrera sans son IU visible. Mettez-la à
visible.
objMapForce.Visible = true;

// ************************** code snippet pour "Simple Document Access"
***********************

// Localisez des exemples via la variable d'environnement USERPROFILE. Le chemin doit
être adapté aux versions majeures de release.
objWshShell = WScript.CreateObject("WScript.Shell");
majorVersionYear = objMapForce.MajorVersion + 1998
strExampleFolder = objWshShell.ExpandEnvironmentStrings("%USERPROFILE%") + "\\Documents\
\Altova\\MapForce" + majorVersionYear + "\\MapForceExamples\\";

objMapForce.Documents.OpenDocument(strExampleFolder + "CompletePO.mfd");
//objMapForce.Documents.OpenDocument(strExampleFolder + "Altova_Hierarchical_DB.mfd");
objMapForce.Documents.NewDocument();

// ************************** code snippet pour "Simple Document Access"
***********************

// ************************** code snippet pour "Iteration"
***********************************

objText = "";
// parcourez tous les documents ouverts utilisant un énumérateur JScript et générez le
code c++
pour (var iterDocs = new Enumerator(objMapForce.Documents); !iterDocs.atEnd();
iterDocs.moveNext())
{
 objText += "Generated c++ code result for document " + iterDocs.item().Name + " :\n";
 objErrorMarkers = iterDocs.item().generateCodeEx(1); // ENUMProgrammingLanguage.eCpp = 1

 bSuccess = true;
for (var iterErrorMarkers = new Enumerator(objErrorMarkers); !iterErrorMarkers.atEnd();
iterErrorMarkers.moveNext())
 {
 bSuccess = false;
 objText += "\t" + iterErrorMarkers.item().Text + "\n";
 }

if (bSuccess)
 objText += "\tCode generation completed successfully.\n";
```

```
 objText += "\n";
}

WScript.Echo(objText);

// parcourez tous les documents ouverts utilisant l'accès basé sur l'index pour la
collection des documents
pour (i = objMapForce.Documents.Count; i > 0; i--)
 objMapForce.Documents.Item(i).Close();


// *************************** code snippet pour "Iteration"
**********************************

//objMapForce.Visible = false; // arrêtera l'application si elle n'a plus de connexions
COM
objMapForce.Visible = true; // maintiendra l'application en cours avec l'IU visible
```

Le code recensé ci-dessus est disponible en tant qu'exemple de fichier (voir Exemples JScript[1123]). Pour exécuter le script, démarrez-le d'une fenêtre d'invite de commandes ou depuis Windows Explorer.

## 17.6.4 Générer le code (alternatif)

L'exemple JScript suivant montre comment charger un document existant et générer différents types de code de mappage pour celui-ci.

```
// ------------------- commencer l'exemple JScript ---------------------
// Générer le code pour le mappage existant.
// fonctionne avec Windows scripting host.

// ----------------- fonction d'aide ------------------
function Exit(strErrorText)
{
 WScript.Echo(strErrorText);
 WScript.Quit(-1);
}

function ERROR(strText, objErr)
{
if (objErr != null)
 Exit ("ERROR: (" + (objErr.number & 0xffff) + ")" + objErr.description + " - " +
strText);
else
 Exit ("ERROR: " + strText);
}
// ------------------------------------------------

// ----------------- PRINCIPAL ------------------

// ----- crée le Shell et FileSystemObject du windows scripting
try
```

```
{
 objWshShell = WScript.CreateObject("WScript.Shell");
 objFSO = WScript.CreateObject("Scripting.FileSystemObject");
}
catch(err)
 { Exit("Can't create WScript.Shell object"); }

// ----- ouvrir MapForce ou accéder à une instance d'exécution et la rendre visible
try
{
 objMapForce = WScript.GetObject ("", "MapForce.Application");
 objMapForce.Visible = true; // supprimer cette ligne pour effectuer le traitement en
arrière-plan
}
catch(err) { WScript.Echo ("Ne peut pas accéder à ou créer MapForce.Application"); }

// ----- ouvrez un mappage existant. adaptez ceci à vos besoins !
objMapForce.OpenDocument(objFSO.GetAbsolutePathName ("Test.mfd"));

// ----- accédez au mappage pour avoir accès aux méthodes de génération de code
var objDoc = objMapForce.ActiveDocument;

// ----- définir les propriétés de sortie de génération de code et appeler toutes les
méthodes de génération de code.
// ----- adaptez les répertoires de sortie à vos besoins
try
{
// ----- la génération de code utilise quelques-unes de ces options
var objOptions = objMapForce.Options;

// ----- générer XSLT -----
 objOptions.XSLTDefaultOutputDirectory = "C:\\test\\TestCOMServer\\XSLT";
 objDoc.GenerateXSLT();

// ----- générer le code Java -----
 objOptions.CodeDefaultOutputDirectory = "C:\\test\\TestCOMServer\\Java";
 objDoc.GenerateJavaCode();

// ----- générer le code CPP, utiliser les mêmes options de code cpp comme la dernière
fois -----
 objOptions.CodeDefaultOutputDirectory = "C:\\test\\TestCOMServer\\CPP";
 objDoc.GenerateCppCode();

// ----- générer le code C#, utiliser les options de code C# code comme la dernière fois
-----
 objOptions.CodeDefaultOutputDirectory = "C:\\test\\TestCOMServer\\CHash";
 objDoc.GenerateCHashCode();
}
catch (err)
 { ERROR ("en générant XSL ou le code de programme", err); }


// masquer MapForce pour permettre son arrêt
objMapForce.Visible = false;
```

```
// -------------------- fin de l'exemple ---------------------
```

## 17.6.5 Exécuter un mappage

L'exemple Jscript suivant montre comment charger un document existant avec un mappage simple, accéder à ses composants, définir des noms de fichier d'instance d'entrée et de sortie et exécuter le mappage.

```
/*
 Cet exemple de fichier réalise les opérations suivantes :

 Charger le document de mappage de MapForce.
 Trouver le composant source et cible.
 Définir les noms de fichier d'instance d'entrée et de sortie.
 Exécuter la transformation.

 Fonctionne avec l'hôte de scripting Windows.
*/

// ---- assistance générale ------------------------------

function Exit( message )
{
 WScript.Echo( message );
 WScript.Quit(-1);
}

function ERROR( message, err )
{
if( err != null )
 Exit( "ERROR: (" + (err.number & 0xffff) + ") " + err.description + " - " + message );
else
 Exit( "ERROR: " + message );
}



// ---- constantes MapForce -----------------------

var eComponentUsageKind_Unknown = 0;
var eComponentUsageKind_Instance = 1;
var eComponentUsageKind_Input = 2;
var eComponentUsageKind_Output = 3;



// ---- assistance MapForce -----------------------

// Cherche dans le mappage spécifié pour un composant par le nom et le renvoie.
// Si pas trouvé, renvoie une erreur.
function FindComponent( mapping, component_name )
{
```

```
var components = mapping.Components;
for( var i = 0 ; i < components.Count ; ++i )
 {
var component = components.Item( i + 1 );
if( component.Name == component_name )
return component;
 }
throw new Error( "Cannot find component with name " + component_name );
}



// Parcourt les composants dans un mappage et renvoie le premier qu'il trouve agissant en
tant que
// composant source (c'est-à-dire qui a des connexions à sa droite).
function GetFirstSourceComponent( mapping )
{
var components = mapping.Components;
for( var i = 0 ; i < components.Count ; ++i )
 {
var component = components.Item( i + 1 );
if( component.UsageKind == eComponentUsageKind_Instance &&
 component.HasOutgoingConnections )
 {
return component;
 }
 }
throw new Error( "Cannot find a source component" );
}

// Parcourt les composants dans un mappage et renvoie le premier qu'il trouve agissant en
tant que
// composant cible (c'est-à-dire qui a des connexions à sa gauche).
function GetFirstTargetComponent( mapping )
{
var components = mapping.Components;
for( var i = 0 ; i < components.Count ; ++i )
 {
var component = components.Item( i + 1 );
if( component.UsageKind == eComponentUsageKind_Instance &&
 component.HasIncomingConnections )
 {
return component;
 }
 }
throw new Error( "Cannot find a target component" );
}



function IndentTextLines( s )
{
return "\t" + s.replace( /\n/g, "\n\t" );
}
```

```
function GetAppoutputLineFullText( oAppoutputLine )
{
var s = oAppoutputLine.GetLineText();
var oAppoutputChildLines = oAppoutputLine.ChildLines;
var i;

for( i = 0 ; i < oAppoutputChildLines.Count ; ++i )
 {
 oAppoutputChildLine = oAppoutputChildLines.Item( i + 1 );
 sChilds = GetAppoutputLineFullText( oAppoutputChildLine );
 s += "\n" + IndentTextLines( sChilds );
 }

return s;
}

// Créer un string bien formaté depuis AppOutputLines
function GetResultMessagesString( oAppoutputLines )
{
var s1 = "Transformation result messages:\n";
var oAppoutputLine;
var i;

for( i = 0 ; i < oAppoutputLines.Count ; ++i )
 {
 oAppoutputLine = oAppoutputLines.Item( i + 1 );
 s1 += GetAppoutputLineFullText( oAppoutputLine );
 s1 += "\n";
 }

return s1;
}



// ---- PRINCIPAL -------------------------------------

var wshShell;
var fso;
var mapforce;

// créer le Shell et FileSystemObject du windows scripting system
try
{
 wshShell = WScript.CreateObject( "WScript.Shell" );
 fso = WScript.CreateObject( "Scripting.FileSystemObject" );
}
catch( err )
 { ERROR( "Can't create windows scripting objects", err ); }

// ouvrir MapForce ou accéder à une instance en cours d'exécution
try
{
```

```
 mapforce = WScript.GetObject( "", "MapForce.Application" );
}
catch( err )
 { ERROR( "Can't access or create MapForce.Application", err ); }

try
{
// Rendre l'IU de MapForce visible. Il s'agit d'une exigence API pour la génération de
sortie.
 mapforce.Visible = true;

// ouvrir un mappage existant.
// **** ajustez le chemin des exemples à vos besoins ! **************
var sMapForceExamplesPath = fso.BuildPath(
 wshShell.SpecialFolders( "MyDocuments" ),
"Altova\\MapForce2024\\MapForceExamples" );
var sDocFilename = fso.BuildPath( sMapForceExamplesPath, "PersonList.mfd" );
var doc = mapforce.OpenDocument( sDocFilename );

// Trouver des composants existants par le nom dans le mappage principal.
// Notez que les noms des composants ne sont pas nécessairement uniques étant donné que
le nom du composant d'un schéma
// est dérivé de son nom de fichier du schéma.
var source_component = FindComponent( doc.MainMapping, "Employees" );
var target_component = FindComponent( doc.MainMapping, "PersonList" );
// Si vous ne connaissez pas les noms des composants pour une raison ou une autre, vous
pourrez
// utiliser les fonctions suivantes à la place de FindComponent.
//var source_component = GetFirstSourceComponent( doc.MainMapping );
//var target_component = GetFirstTargetComponent( doc.MainMapping );

// spécifier les fichiers d'entrée et de sortie désirés.
 source_component.InputInstanceFile = fso.BuildPath( sMapForceExamplesPath,
"Employees.xml" );
 target_component.OutputInstanceFile = fso.BuildPath( sMapForceExamplesPath,
"test_transformation_results.xml" );

// Effectuer la transformation.
// Vous pouvez utiliser doc.GenerateOutput() si vous n'avez pas besoin de messages de
résultat.
// Si vous avez un mappage avec plus d'un compostant cible et vous voulez
// exécuter la transformation uniquement pour un seul composant cible,
// appelez target_component.GenerateOutput() à la place.
var result_messages = doc.GenerateOutputEx();

var summary_info =
"Transformation performed from " + source_component.InputInstanceFile + "\n" +
"to " + target_component.OutputInstanceFile + "\n\n" +
 GetResultMessagesString( result_messages );
 WScript.Echo( summary_info );
}
catch( err )
{
```

```
 ERROR( "Failure", err );
}
```

## 17.6.6 Project Tasks

L'exemple JScript suivant montre comment utiliser l'API de MapForce pour automatiser les tâches pertinentes aux projets de MapForce. Avant d'exécuter l'exemple, assurez-vous d'éditer la variable strSamplePath pour qu'elle vous dirige vers le dossier de votre installation de MapForce : **C:\Users\<username>\Documents\Altova\MapForce2024\MapForceExamples**.

Afin d'exécuter avec succès toutes les opérations dans cet exemple ci-dessous, vous aurez besoin de l'édition Enterprise de MapForce. Si vous avez l'édition Professional, commentez les lignes qui insèrent le projet WebService.

```
// ############ variables globaux #################
var objMapForce = null;
var objWshShell = null;
var objFSO = null;

// !!! adaptez le chemin suivant à vos besoins. !!!
var strSamplePath = "C:\\Users\\<username>\\Documents\\Altova\\MapForce2024\
\MapForceExamples\\";

// ###################### Helpers ############################

function Exit(strErrorText)
{
 WScript.Echo(strErrorText);
 WScript.Quit(-1);
}

function ERROR(strText, objErr)
{
if (objErr != null)
 Exit ("ERROR: (" + (objErr.number & 0xffff) + ")" + objErr.description + " - " +
strText);
else
 Exit ("ERROR: " + strText);
}

function CreateGlobalObjects ()
{
// le Shell et FileSystemObject de l'hôte de scripting windows souvent utile
try
 {
 objWshShell = WScript.CreateObject("WScript.Shell");
 objFSO = WScript.CreateObject("Scripting.FileSystemObject");
 }
catch(err)
 { Exit("Can't create WScript.Shell object"); }
```

```
// créer la connexion de MapForce
// s'il y a une instance d'exécution de MapForce (qui n'a jamais eu de connexion) -
utilisez-la,
// autrement, nous créons automatiquement une nouvelle instance
try
 {
 objMapForce = WScript.GetObject ("", "MapForce.Application");
 }
catch(err)
 {
 { Exit("Can't access or create MapForce.Application"); }
 }
}

// ------------------------------------------------------------
// imprimer les items d'arborescence du projet et leurs propriétés de manière récursive.
// ------------------------------------------------------------
function PrintProjectTree( objProjectItemIter, strTab )
{
while ( ! objProjectItemIter.atEnd() )
 {
// obtenir l'item de projet actuel
 objItem = objProjectItemIter.item();

try
 {
// ----- imprimer les propriétés communes
 strGlobalText += strTab + "[" + objItem.Kind + "]" + objItem.Name + "\n";

// ----- imprimer les propriétés de génération de code, si disponible
try
 {
if ( objItem.CodeGenSettings_UseDefault )
 strGlobalText += strTab + " Use default code generation settings\n";
else
 strGlobalText += strTab + " code generation language is " +
 objItem.CodeGenSettings_Language +
" output folder is " + objItem.CodeGenSettings_OutputFolder + "\n";
 }
catch( err ) {}

// ----- imprimer les paramètres WSDL, si possible
try
 {
 strGlobalText += strTab + " WSDL File is " + objItem.WSDLFile +
" Qualified Name is " + objItem.QualifiedName + "\n";
 }
catch( err ) {}
 }
catch( ex )
 { strGlobalText += strTab + "[" + objItem.Kind + "]\n" }

// ---- recurse
 PrintProjectTree( new Enumerator( objItem ), strTab + ' ' );
```

```
 objProjectItemIter.moveNext();
 }
}

// ------------------------------------------------------------
// Charger l'exemple de projet installé avec MapForce.
// ------------------------------------------------------------
function LoadSampleProject()
{
// fermer le projet ouvert
 objProject = objMapForce.ActiveProject;
if ( objProject != null )
 objProject.Close();

// ouvrir l'exemple de projet et itérer à travers celui-ci.
 objProject = objMapForce.OpenProject(strSamplePath + "MapForceExamples.mfp");
// dump properties of all project items
 strGlobalText = '';
 PrintProjectTree( new Enumerator (objProject), ' ' )
 WScript.Echo( strGlobalText );

 objProject.Close();
}

// ------------------------------------------------------------
// Créer un nouveau projet avec quelques dossiers, mappages et un
// projet de service Web.
// ------------------------------------------------------------
function CreateNewProject()
{
try
 {
// créer un nouveau projet et spécifier le fichier où le mettre.
 objProject = objMapForce.NewProject(strSamplePath + "Sample.mfp");

// créer une structure de dossier simple
 objProject.CreateFolder( "New Folder 1");
 objFolder1 = objProject.Item(1);
 objFolder1.CreateFolder( "New Folder 2");
 objFolder2 = ( new Enumerator( objFolder1 ) ).item(); // an alternative to Item(0)

// ajouter deux différents mappages à la structure de dossier
 objFolder1.AddFile( strSamplePath + "DB_Altova_SQLXML.mfd");
 objMapForce.Documents.OpenDocument(strSamplePath + "InspectionReport.mfd");
 objFolder2.AddActiveFile();

// écraser les paramètres de génération de code pour ce dossier
 objFolder2.CodeGenSettings_UseDefault = false;
 objFolder2.CodeGenSettings_OutputFolder = strSamplePath + "SampleOutput"
 objFolder2.CodeGenSettings_Language = 1; //C++

// insérer le projet de service Web basé sur un fichier wsdl depuis les exemples
installés
 objProject.InsertWebService( strSamplePath + "TimeService/TimeService.wsdl",
```

```
"{http://www.Nanonull.com/TimeService/}TimeService",
"TimeServiceSoap",
true );
 objProject.Save();
if ( ! objProject.Saved )
 WScript.Echo("problem occurred when saving project");

// vider l'arborescence de projet
 strGlobalText = '';
 PrintProjectTree( new Enumerator (objProject), ' ' )
 WScript.Echo( strGlobalText );
 }
catch (err)
 { ERROR("while creating new project", err ); }
}

// ------------------------------------------------------------
// Générer le code pour une sous-arborescence de projet. Combiner le code par défaut
// paramètres de génération et paramètres surchargés.
// ------------------------------------------------------------
function GenerateCodeForNewProject()
{
// puisque le projet de service Web contient uniquement des mappages initiaux,
// nous ne générons du code que pour notre dossier client.
// les paramètres de génération de code du projet sont utilisés pour le Dossier1,
// tandis que le Dossier2 fournit les valeurs écrasées.
 objFolder = objProject.Item(1);
 objFolder1.GenerateCode();
}

// ////////////////////// PRINCIPAL ////////////////////////////

CreateGlobalObjects();
objMapForce.Visible = true;

LoadSampleProject();
CreateNewProject();
GenerateCodeForNewProject();

// supprimer pour arrêter l'application quand le script se termine
// objMapForce.Visible = false;
```

# 17.7 Référence d'objet

Cette section fournit une référence aux objets de l'API COM de MapForce. Les objets sont décrits de manière générique puisque l'API peut être utilisée avec tout langage qui prend en charge l'appel d'un objet COM. Pour les exemples spécifiques au langage, voir :

- Exemple Projet C# 1114
- Exemple Projet Java 1119
- Exemples JScript 1123

La référence API contient deux principales sections, chacune décrivant les interfaces et les types d'énumération utilisés respectivement dans l'API. Les valeurs d'énumération contiennent à la fois le nom de string et la valeurs numériques. Si votre environnement d'écriture de scripts ne prend pas en charge les énumérations, utilisez les valeurs numériques à la place.

Dans .NET, pour chaque interface de l'interface d'automatisation COM de MapForce, une classe .NET existe dotée du même nom. Les types COM seront également convertis en type .NET approprié. Par exemple, un type tel que `Long` dans l'API COM apparaîtrait en tant que `System.Int32` dans .NET.

Dans Java, veuillez noter les variations de la syntaxe suivantes :

- **Classes et noms de classe**. Pour chaque interface de l'interface d'automatisation de MapForce, une classe Java existe dotée du nom de l'interface.
- **Noms de méthodes**. Les noms de méthodes sur l'interface Java sont les mêmes que ceux utilisés sur les interfaces COM mais commencent par une minuscule pour être conforme aux conventions d'affectation de noms de Java. Pour avoir accès aux propriétés COM, les méthodes Java qui attribuent les préfixes « get » et « set » au nom de propriété peuvent être utilisées. Si une propriété ne prend pas en charge l'accès en écriture, aucune méthode setter n'est disponible. Par exemple, pour la propriété du `Nom` de l'interface `Document`, les méthodes de Java `getName` et `setName` sont disponibles.
- **Énumérations**. Pour chaque énumération définie dans l'interface d'automatisation, une énumération Java est définie avec les mêmes nom et valeurs.
- **Événements et gestionnaires d'événement**. Pour chaque interface dans l'interface d'automatisation qui prend en charge les événements, une interface Java avec le même nom plus 'Événement' est disponible. Pour simplifier la surcharge d'événements simples, une classe Java est fournie avec des implémentations par défaut pour tous les événements. Le nom de cette classe Java est le nom de l'interface d'événement plus « DefaultHandler ». Par exemple :

```
Application // classe Java pour accéder à l'application
ApplicationEvents // interface d'événements pour l'application
ApplicationEventsDefaultHandler // gestionnaire par défaut pour « ApplicationEvents »
```

## 17.7.1 Interfaces

### 17.7.1.1 Application

L'interface de l'`application` est l'interface d'un objet d'application de MapForce. Elle représente le point d'accès principal pour l'application MapForce elle-même. Cette interface est le point de départ pour toute opération supplémentaire avec MapForce ou pour extraire ou créer d'autres objets d'automatisation liés à

MapForce. Pour toute information concernant la création d'une instance de l'objet d'application, voir Accéder à l'API[1103].

Propriétés pour parcourir le modèle d'objet :

- `Application`
- `Parent`
- `Options`
- `Projet`
- `Documents`

Statut de l'application :

- `visible`
- `Nom`
- `Quitter`
- `Statut`
- `WindowHandle`

Designs de MapForce :

- `NewDocument`
- `OpenDocument`
- `OpenURL`
- `ActiveDocument`

Projets de MapForce :

- `NewProject`
- `OpenProject`
- `ProjetActif`

Génération de code de MapForce :

- `HighlightSerializedMarker`

Ressources globales :

- `GlobalResourceConfig`
- `GlobalResourceFile`

Information de version :

- `Édition`
- `IsAPISupported`
- `MajorVersion`
- `MinorVersion`

## Propriétés

| Nom | Description |
|---|---|
| `ActiveDocument`[1142] | Lecture seule. |

| Nom | Description |
|---|---|
| | Retourne l'objet d'automatisation du document actif actuel. Cette propriété retourne de la même manière que `Documents.ActiveDocument`. |
| `ProjetActif` 1142 | Lecture seule.<br>Retourne l'objet d'automatisation du projet actif actuel. |
| `Application` 1143 | Lecture seule.<br>Extrait l'objet de l'application de premier niveau. |
| `Documents` 1143 | Lecture seule.<br>Retourne une collection de tous les documents actuellement ouverts. |
| `Édition` 1143 | Lecture seule.<br>Renvoie l'édition de l'application, par ex., « Altova MapForce Enterprise Edition » pour l'édition d'enterprise. |
| `GlobalResourceConfig` 1144 | Obtient ou définit le nom du fichier de configuration de la ressource globale active. Par défaut, le fichier est appelé **GlobalResources.xml**.<br><br>Le fichier de configuration peut être renommé et enregistré sous tout emplacement. Vous pouvez donc avoir de multiples fichiers XML de ressources globales. Toutefois, seulement un de ces fichiers XML de ressources globales peut être actif par application, à usage unique, et uniquement les définitions contenues dans ce fichier seront disponibles pour l'application. |
| `GlobalResourceFile` 1144 | Obtient ou définit le fichier de définition de la ressource globale. Par défaut, le fichier est appelé GlobalResources.xml. |
| `IsAPISupported` 1145 | Lecture seule.<br>Renvoie true si l'API est prises en charge dans cette version de MapForce. |
| `LibraryImports` 1145 | Lecture seule.<br>Obtient une collection de bibliothèques importées. Dans l'interface utilisateur graphique de MapForce, celles-ci correspondent aux entrées de la fenêtre **Gérer les bibliothèques**, ajoutées au niveau de l'application. |
| `MajorVersion` 1145 | Lecture seule.<br>Obtient le numéro de version majeur de MapForce. La version est calculée commençant en 1998 et incrémentée de un chaque année. Par exemple, la version majeure est « 18 » pour la release 2016. |
| `MinorVersion` 1146 | Lecture seule.<br>Le numéro de version mineur du produit, par exemple 2 pour 2006 R2 SP1. |

| Nom | Description |
|---|---|
| Nom 1146 | Lecture seule.<br>Le nom de l'application. |
| Options 1147 | Lecture seule.<br>Cette propriété donne accès aux options qui configurent le code de génération. |
| Parent 1147 | Lecture seule.<br>L'objet parent selon le modèle d'objet. |
| ServicePackVersion 1147 | Lecture seule.<br>Le numéro de version du Service Pack, par exemple 1 pour 2016 R2 SP1. |
| Statut 1148 | Lecture seule.<br>Le statut de l'application. Il s'agit d'une des valeurs de l'énumération `ENUMApplicationStatus`. |
| Visible 1148 | **True** si MapForce est affiché à l'écran (cependant, cela pourrait être couvert par d'autres applications ou être iconisé).<br><br>**False** si MapForce est masqué. La valeur par défaut pour MapForce lorsqu'il est automatiquement démarré suite à une requête depuis le serveur Automation `Application` est false. Dans tous les autres cas, la propriété est initialisée par **true**.<br><br>On peut dire qu'une instance d'application qui est visible est contrôlée par l'utilisateur (et éventuellement par des clients connectés par le biais de l'interface automation). Elle ne s'arrêtera que si elle reçoit une requête explicite de l'utilisateur. Pour arrêter une instance d'application, définissez sa visibilité à false et effacez toutes les références de cette instance à l'intérieur de votre programme. L'instance d'application s'arrêtera automatiquement quand aucun autre client COM n'a de références y afférentes. |
| WindowHandle 1149 | Lecture seule.<br>Extrait Window Handle de l'application. |

## Méthodes

| Nom | Description |
|---|---|
| HighlightSerializedMarker 1149 | Utilisez cette méthode pour marquer un emplacement dans un fichier de mappage qui a été précédemment sérialisé. Si le document correspondant n'a pas encore été chargé, il sera chargé en premier. Voir `Document.GenerateCodeEx` pour une méthode d'extraction d'un marqueur sérialisé. |

| Nom | Description |
|---|---|
| NewDocument 1150 | Crée un nouveau document vide. Le nouveau document ouvert devient le `ActiveDocument`. Cette méthode est une forme raccourcie de `Documents.NewDocument`. |
| NewProject 1150 | Crée un nouveau document vide. Le projet actuel est fermé. Le nouveau projet est accessible sous `ActiveProject`. |
| NewWebServiceProject 1151 | Crée un nouveau projet Web Service vide. Le nouveau projet est accessible sous `ActiveProject`. Cette méthode est uniquement disponible dans MapForce Enterprise Edition. |
| OpenDocument 1151 | Charge un fichier de document précédemment enregistré et poursuit le travail dans le fichier. Le nouveau document ouvert devient le `ActiveDocument`. Cette méthode est une forme raccourcie de `Documents.OpenDocument`. |
| OpenProject 1151 | Ouvre un projet existant de Mapforce (*.mfp). Le projet actuel est fermé. Le projet récemment ouvert est accessible sous `ActiveProject`. |
| OpenURL 1152 | Charge un fichier de document précédemment enregistré depuis un emplacement URL. Permet la fourniture du nom d'utilisateur et du mot de passe. |
| Quitter 1153 | Se déconnecte de MapForce pour permettre à l'application de se fermer. Appeler cette méthode est optionnel puisque MapForce assure le suivi de connexions COM externes et reconnaît automatiquement une déconnexion. Pour plus d'information sur un arrêt automatique, voir la propriété `Visible`. |

## Événements

| Nom | Description |
|---|---|
| OnDocumentOpened 1153 | Cet événement est déclenché quand un document existant ou un nouveau document est ouvert. L'événement de fermeture correspondant est `Document.OnDocumentClosed`. |
| OnProjectOpened 1153 | Cet événement est déclenché quand un document existant ou un nouveau document est chargé dans l'application. L'événement de fermeture correspondant est `Document.OnDocumentClosed`. |
| OnShutdown 1154 | Cet événement est déclenché quand l'application est en cours d'arrêt. |

### 17.7.1.1.1 Propriétés

#### *17.7.1.1.1.1 DocumentActif*

Retourne l'objet d'automatisation du document actif actuel. Cette propriété retourne de la même manière que `Documents.ActiveDocument`.

#### Signature

```
ActiveDocument : Document
```

#### Erreurs

| Code d'erreur : | Description |
|---|---|
| 1000 | L'objet d'application n'est plus valide. |
| 1001 | L'adresse invalide pour le paramètre de retour a été spécifiée. |

#### *17.7.1.1.1.2 ProjetActif*

Retourne l'objet d'automatisation du projet actif actuel.

#### Signature

```
ActiveProject : Projet
```

#### Erreurs

| Code d'erreur : | Description |
|---|---|
| 1000 | L'objet d'application n'est plus valide. |
| 1001 | L'adresse invalide pour le paramètre de retour a été spécifiée. |

#### *17.7.1.1.1.3 Application*

Extrait l'objet de l'application de premier niveau.

### Signature

```
Application : Application
```

### Erreurs

| Code d'erreur : | Description |
|---|---|
| 1000 | L'objet d'application n'est plus valide. |
| 1001 | L'adresse invalide pour le paramètre de retour a été spécifiée. |

#### *17.7.1.1.1.4 Documents*

Retourne une collection de tous les documents actuellement ouverts.

### Signature

```
Documents : Documents
```

### Erreurs

| Code d'erreur | Description |
|---|---|
| 1000 | L'objet d'application n'est plus valide. |
| 1001 | L'adresse invalide pour le paramètre de renvoi a été spécifiée. |

#### *17.7.1.1.1.5 Édition*

Renvoie l'édition de l'application, par ex., « Altova MapForce Enterprise Edition » pour l'édition d'enterprise.

### Signature

```
Edition : String
```

### Erreurs

| Code d'erreur | Description |
|---|---|
| 1000 | L'objet d'application n'est plus valide. |
| 1001 | L'adresse invalide pour le paramètre de renvoi a été spécifiée. |

#### *17.7.1.1.1.6 GlobalResourceConfig*

Obtient ou définit le nom du fichier de configuration de la ressource globale active. Par défaut, le fichier est appelé **GlobalResources.xml**.

Le fichier de configuration peut être renommé et enregistré à tout emplacement. Vous pouvez donc avoir de multiples fichiers XML des Ressources globales. Toutefois, uniquement un seul fichier XML des Ressources globales peut être actif, par application, et une seule fois, et seules les définitions contenues dans ce fichier seront disponibles pour l'application.

### Signature

```
GlobalResourceConfig : String
```

### Erreurs

| Error code | Description |
|---|---|
| 1000 | L'objet d'application n'est plus valide. |
| 1001 | Adresse invalide a été spécifiée pour le paramètre de renvoi. |

#### *17.7.1.1.1.7 GlobalResourceFile*

Obtient ou définit le fichier de définition de la ressource globale. Par défaut, le fichier est appelé GlobalResources.xml.

### Signature

```
GlobalResourceFile : String
```

### Erreurs

| Error code | Description |
|---|---|
| 1000 | L'objet d'application n'est plus valide. |
| 1001 | Adresse invalide a été spécifiée pour le paramètre de renvoi. |

#### *17.7.1.1.1.8 IsAPISupported*

Renvoie true si l'API est prise en charge dans cette version de MapForce.

### Signature

```
IsAPISupported : booléen
```

### Erreurs

| Code d'erreur | Description |
|---|---|
| 1001 | Adresse invalide a été spécifiée pour le paramètre de renvoi. |

#### *17.7.1.1.1.9 LibraryImports*

Obtient une collection de bibliothèques importées. Dans l'interface utilisateur graphique de MapForce, celles-ci correspondent aux entrées de la fenêtre **Gérer les bibliothèques**, ajoutées au niveau de l'application.

### Signature

```
LibraryImports : LibraryImports
```

### Erreurs

| Code d'erreur | Description |
|---|---|
| 1200 | L'objet n'est plus valide. |
| 1201 | L'adresse invalide pour le paramètre de renvoi a été spécifiée. |

#### *17.7.1.1.1.10 MajorVersion*

Obtient le numéro de la version majeure de MapForce. La version est calculée commençant en 1998 et est incrémentée de un chaque année. Par exemple, la version majeure est « 18 » pour la release 2016.

### Signature

```
MajorVersion : Long
```

### Erreurs

| Code d'erreur | Description |
|---|---|
| 1000 | L'objet d'application n'est plus valide. |
| 1001 | Adresse invalide a été spécifiée pour le paramètre de renvoi. |

## *17.7.1.1.1.11 MinorVersion*

Le numéro de la version mineure du produit, par ex., 2 pour 2006 R2 SP1.

### Signature

```
MinorVersion : Long
```

### Erreurs

| Code d'erreur | Description |
|---|---|
| 1000 | L'objet d'application n'est plus valide. |
| 1001 | Adresse invalide a été spécifiée pour le paramètre de renvoi. |

## *17.7.1.1.1.12 Nom*

Le nom de l'application.

### Signature

```
Nom : String
```

### Erreurs

| Code d'erreur | Description |
|---|---|
| 1000 | L'objet d'application n'est plus valide. |
| 1001 | Adresse invalide a été spécifiée pour le paramètre de renvoi. |

### *17.7.1.1.1.13 Options*

Cette propriété donne accès aux options qui configurent le code de génération.

#### Signature

```
Options : Options
```

#### Erreurs

| Code d'erreur | Description |
|---|---|
| 1000 | L'objet d'application n'est plus valide. |
| 1001 | L'adresse invalide pour le paramètre de renvoi a été spécifiée. |

### *17.7.1.1.1.14 Parent*

L'objet parent selon le modèle d'objet.

#### Signature

```
Parent : Application
```

#### Erreurs

| Code d'erreur | Description |
|---|---|
| 1000 | L'objet d'application n'est plus valide. |
| 1001 | L'adresse invalide pour le paramètre de renvoi a été spécifiée. |

### *17.7.1.1.1.15 ServicePackVersion*

Le numéro de version du Service Pack, par exemple 1 pour 2016 R2 SP1.

#### Signature

```
ServicePackVersion : Long
```

### Erreurs

| Code d'erreur | Description |
|---|---|
| 1000 | L'objet d'application n'est plus valide. |
| 1001 | L'adresse invalide pour le paramètre de renvoi a été spécifiée. |

#### *17.7.1.1.1.16 Statut*

Le statut de l'application. Il s'agit d'une des valeurs de l'énumération `ENUMApplicationStatus`.

### Signature

```
Statut : ENUMApplicationStatus
```

### Erreurs

| Code d'erreur | Description |
|---|---|
| 1001 | L'adresse invalide pour le paramètre de renvoi a été spécifiée. |

#### *17.7.1.1.1.17 Visible*

**True** si MapForce est affiché à l'écran (cependant, cela pourrait être couvert par d'autres applications ou être iconisé).

**False** si MapForce est masqué. La valeur par défaut pour MapForce lorsqu'il est automatiquement démarré suite à une requête depuis le serveur Automation `Application` est false. Dans tous les autres cas, la propriété est initialisée par **true**.

On peut dire qu'une instance d'application qui est visible est contrôlée par l'utilisateur (et éventuellement par des clients connectés par le biais de l'interface automation). Elle ne s'arrêtera que si elle reçoit une requête explicite de l'utilisateur. Pour arrêter une instance d'application, définissez sa visibilité à false et effacez toutes les références de cette instance à l'intérieur de votre programme. L'instance d'application s'arrêtera automatiquement quand aucun autre client COM n'a de références y afférentes.

### Signature

```
Visible : Booléen
```

### Erreurs

| Code d'erreur | Description |
|---|---|
| 1000 | L'objet d'application n'est plus valide. |

| Code d'erreur | Description |
|---|---|
| 1001 | L'adresse invalide pour le paramètre de renvoi a été spécifiée. |

#### *17.7.1.1.1.18 WindowHandle*

Extrait Window Handle de l'application.

##### Signature

```
WindowHandle : Long
```

##### Erreurs

| Code d'erreur | Description |
|---|---|
| 1000 | L'objet d'application n'est plus valide. |
| 1001 | L'adresse invalide pour le paramètre de renvoi a été spécifiée. |

### 17.7.1.1.2 Méthodes

#### *17.7.1.1.2.1 HighlightSerializedMarker*

Utilisez cette méthode pour marquer un emplacement dans un fichier de mappage qui a été précédemment sérialisé. Si le document correspondant n'a pas encore été chargé, il sera chargé en premier. Voir `Document.GenerateCodeEx` pour une méthode d'extraction d'un marqueur sérialisé.

##### Signature

```
HighlightSerializedMarker(dans i_strSerializedMarker:String) -> Void
```

##### Paramètres

| Nom | Type | Description |
|---|---|---|
| **i_strSerializedMarker** | `String` | L'objet `ErrorMarker` pour mettre en évidence. Utiliser `ErrorMaker.Serialized` pour obtenir cette valeur. |

### Erreurs

| Code d'erreur | Description |
|---|---|
| 1000 | L'objet d'application n'est plus valide. |
| 1001 | L'adresse invalide pour le paramètre de renvoi a été spécifiée. |
| 1007 | Le string passé dans **i_strSerializedMarker** n'est pas reconnu comme marqueur MapForce sérialisé. |
| 1008 | Le marqueur pointe vers un emplacement qui n'est plus valide. |

## *17.7.1.1.2.2 NewDocument*

Crée un nouveau document vide. Le nouveau document ouvert devient le `ActiveDocument`. Cette méthode est une forme raccourcie de `Documents.NewDocument`.

### Signature

```
NewDocument() -> Document
```

### Erreurs

| Code d'erreur | Description |
|---|---|
| 1000 | L'objet d'application n'est plus valide. |
| 1001 | L'adresse invalide pour le paramètre de renvoi a été spécifiée. |

## *17.7.1.1.2.3 NewProject*

Crée un nouveau document vide. Le projet actuel est fermé. Le nouveau projet est accessible sous `ActiveProject`.

### Signature

```
NewProject() -> Projet
```

### Erreurs

| Code d'erreur | Description |
|---|---|
| 1000 | L'objet d'application n'est plus valide. |
| 1001 | L'adresse invalide pour le paramètre de renvoi a été spécifiée. |

#### *17.7.1.1.2.4 NewWebServiceProject*

Crée un nouveau projet Web Service vide. Le nouveau projet est accessible sous `ActiveProject`. Cette méthode est uniquement disponible dans MapForce Enterprise Edition.

### Signature

```
NewWebServiceProject() -> Project
```

### Erreurs

| Code d'erreur | Description |
|---|---|
| 1004 | Erreur en créant un nouveau projet. |
| 1005 | Mauvaise édition de MapForce. |

#### *17.7.1.1.2.5 OpenDocument*

Charge un fichier de document précédemment enregistré et poursuit le travail dans le fichier. Le nouveau document ouvert devient le `ActiveDocument`. Cette méthode est une forme raccourcie de `Documents.OpenDocument`.

### Signature

```
OpenDocument(dans i_strFileName:String) -> Document
```

### Paramètres

| Nom | Type | Description |
|---|---|---|
| **i_strFileName** | `String` | Le chemin du document à ouvrir. |

### Erreurs

| Code d'erreur | Description |
|---|---|
| 1000 | L'objet d'application n'est plus valide. |
| 1001 | L'adresse invalide pour le paramètre de renvoi a été spécifiée. |

#### *17.7.1.1.2.6 OpenProject*

Ouvre un projet existant de Mapforce (*.mfp). Le projet actuel est fermé. Le projet récemment ouvert est accessible sous `ActiveProject`.

### Signature

```
OpenProject(in i_strFileName:String) -> Project
```

### Paramètres

| Nom | Type | Description |
|---|---|---|
| **i_strFileName** | `String` | Le chemin du projet à ouvrir. |

### Erreurs

| Code d'erreur | Description |
|---|---|
| 1000 | L'objet d'application n'est plus valide. |
| 1001 | L'adresse invalide pour le paramètre de renvoi a été spécifiée. |
| 1002 | Le nom de fichier fourni n'est pas valide. |

## *17.7.1.1.2.7 OpenURL*

Charge un fichier de document précédemment enregistré depuis un emplacement URL. Permet la fourniture du nom d'utilisateur et du mot de passe.

### Signature

```
OpenURL(in strURL:String, in strUser:String, in strPassword:String) -> Void
```

### Paramètres

| Nom | Type | Description |
|---|---|---|
| **strURL** | `String` | L'URL depuis lequel le document doit être chargé. |
| **strUser** | `String` | Le nom d'utilisateur requis pour accéder l'URL. |
| **strPassword** | `String` | Le mot de passe requis pour accéder l'URL. |

### Erreurs

| Code d'erreur | Description |
|---|---|
| 1000 | L'objet d'application n'est plus valide. |

| Code d'erreur | Description |
|---|---|
| 1001 | L'adresse invalide pour le paramètre de renvoi a été spécifiée. |
| 1002 | L'URL fourni n'est pas valide. |
| 1006 | Erreur en ouvrant le fichier URL. |

#### *17.7.1.1.2.8 Quitter*

Se déconnecte de MapForce pour permettre à l'application de se fermer. Appeler cette méthode est optionnel puisque MapForce assure le suivi de connexions COM externes et reconnaît automatiquement une déconnexion. Pour plus d'information sur un arrêt automatique, voir la propriété `Visible`.

##### Signature

```
Quit() -> Void
```

##### Erreurs

| Code d'erreur | Description |
|---|---|
| 1000 | L'objet d'application n'est plus valide. |
| 1001 | L'adresse invalide pour le paramètre de renvoi a été spécifiée. |

### 17.7.1.1.3 Événements

#### *17.7.1.1.3.1 OnDocumentOpened*

Cet événement est déclenché quand un document existant ou un nouveau document est ouvert. L'événement de fermeture correspondant est `Document.OnDocumentClosed`.

##### Signature

```
OnDocumentOpened(in i_ipDocument:Document) : Annuler
```

#### *17.7.1.1.3.2 OnProjectOpened*

Cet événement est déclenché quand un document existant ou un nouveau document est chargé dans l'application. L'événement de fermeture correspondant est `Document.OnDocumentClosed`.

### Signature

```
OnProjectOpened(in i_ipProject:Project) : Annuler
```

#### *17.7.1.1.3.3 OnShutdown*

Cet événement est déclenché quand l'application est en cours d'arrêt.

### Signature

```
OnShutdown : Annuler
```

## 17.7.1.2 AppOutputLine

Représente une ligne de message. Contrairement à `ErrorMarker`, sa structure est plus détaillée et contient une collection de lignes enfant, formant pour cette raison une arborescence de lignes de message.

Propriétés pour parcourir le modèle d'objet :

- `Application`
- `Parent`

Accès de ligne :

- `GetLineSeverity`
- `GetLineSymbol`
- `GetLineText`
- `GetLineTextEx`
- `GetLineTextWithChildren`
- `GetLineTextWithChildrenEx`

Une seule `AppOutputLine` consiste en une ou plusieurs sous-lignes. Accès de sous-ligne :

- `GetLineCount`

Une sous-ligne consiste en une ou plusieurs cellules. Accès de cellule :

- `GetCellCountInLine`
- `GetCellIcon`
- `GetCellSymbol`
- `GetCellText`
- `GetCellTextDecoration`
- `GetIsCellText`

En dessous d'une `AppOutputLine`, il peut y avoir zéro, un ou plus de lignes enfant, qui elles-mêmes sont de type `AppOutputLine`, et forment donc une structure arborescente.

Accès de lignes enfant :

- `ChildLines`

## Propriétés

| Nom | Description |
|---|---|
| `Application` 1156 | Lecture seule.<br>Extrait l'objet de l'application de premier niveau. |
| `ChildLines` 1156 | Lecture seule.<br>Renvoie une collection de lignes enfant directes de la ligne actuelle. |
| `Parent` 1157 | Lecture seule.<br>L'objet parent selon le modèle d'objet. |

## Méthodes

| Nom | Description |
|---|---|
| `GetCellCountInLine` 1157 | Obtient le nombre de cellules dans la sous-ligne indiquée par `nLine` dans la `AppOutputLine`actuelle. |
| `GetCellIcon` 1158 | Obtient l'icône de la cellule indiquée par `nCell` dans la sous-ligne actuelle `AppOutputLine` indiquée par `nLine` |
| `GetCellSymbol` 1158 | Obtient le symbole de la cellule indiquée par `nCell` dans la sous-ligne actuelle `AppOutputLine` indiquée par `nLine` |
| `GetCellText` 1159 | Obtient le texte de la cellule indiquée par `nCell` dans la sous-ligne actuelle `AppOutputLine` indiquée par `nLine` |
| `GetCellTextDecoration` 1159 | Obtient la décoration de la cellule de texte indiquée par `nCell` dans la sous-ligne actuelle `AppOutputLine`indiquée par `nLine`. Il peut s'agir d'une des valeurs `ENUMAppOutputLine_TextDecoration`. |
| `GetIsCellText` 1160 | Renvoie true si la cellule indiquée par `nCell` dans la sous-ligne actuelle `AppOutputLine` indiquée par `nLine` est une cellule de texte. |
| `GetLineCount` 1161 | Obtient le nombre de sous-lignes dont la ligne actuelle est composée. |
| `GetLineSeverity` 1161 | Obtient la gravité de la ligne. Il peut s'agir d'une des valeurs `ENUMAppOutputLine_Severity`. |
| `GetLineSymbol` 1161 | Obtient le symbole assigné à toute la ligne. |
| `GetLineText` 1162 | Obtient les contenus de la ligne en tant que texte. |

| Nom | Description |
|---|---|
| GetLineTextEx 1162 | Obtient les contenus de la ligne en tant que texte utilisant la partie spécifiée et les séparateurs de lignes. |
| GetLineTextWithChildren 1163 | Obtient les contenus de la ligne, y compris toutes les lignes descendantes et enfant en tant que texte. |
| GetLineTextWithChildrenEx 1163 | Obtient les contenus de la ligne, y compris toutes les lignes descendantes et enfant en tant que texte utilisant la partie, la ligne, l'onglet et les séparateurs d'item spécifiés. |

### 17.7.1.2.1 Propriétés

#### *17.7.1.2.1.1 Application*

Extrait l'objet de l'application de premier niveau.

#### Signature

```
Application : Application
```

#### Erreurs

| Code d'erreur : | Description |
|---|---|
| 4100 | L'objet n'est plus valide. |
| 4101 | L'adresse invalide pour le paramètre de retour a été spécifiée. |

#### *17.7.1.2.1.2 ChildLines*

Renvoie une collection de lignes enfant directes de la ligne actuelle.

#### Signature

```
ChildLines : AppOutputLines
```

#### Erreurs

| Code d'erreur | Description |
|---|---|
| 4100 | L'objet n'est plus valide. |
| 4101 | L'adresse invalide pour le paramètre de retour a été spécifiée. |

#### *17.7.1.2.1.3 Parent*

L'objet parent selon le modèle d'objet.

### Signature

```
Parent : AppOutputLines
```

### Erreurs

| Code d'erreur | Description |
|---|---|
| 4100 | L'objet n'est plus valide. |
| 4101 | L'adresse invalide pour le paramètre de renvoi a été spécifiée. |

## 17.7.1.2.2 Méthodes

#### *17.7.1.2.2.1 GetCellCountInLine*

Obtient le nombre de cellules dans la sous-ligne indiquée par `nLine` dans la `AppOutputLine`actuelle.

### Signature

```
GetCellCountInLine(in nLine:Long) -> Long
```

### Paramètres

| Nom | Type | Description |
|---|---|---|
| **nLine** | `Long` | Spécifie l'index de la ligne basé sur zéro. |

### Erreurs

| Code d'erreur | Description |
|---|---|
| 4100 | L'objet n'est plus valide. |
| 4101 | L'adresse invalide pour le paramètre de renvoi a été spécifiée. |

### 17.7.1.2.2.2 GetCellIcon (obsolete)

Obtient l'icône de la cellule indiquée par `nCell` dans la sous-ligne actuelle `AppOutputLine` indiquée par `nLine`

#### Signature

```
GetCellIcon(in nLine:Long, in nCell:Long) -> Long
```

#### Paramètres

| Nom | Type | Description |
|---|---|---|
| **nLine** | **Long** | |
| **nCell** | **Long** | |

#### Erreurs

| Code d'erreur | Description |
|---|---|
| 4100 | L'objet n'est plus valide. |
| 4101 | L'adresse invalide pour le paramètre de renvoi a été spécifiée. |

### 17.7.1.2.2.3 GetCellSymbol

Obtient le symbole de la cellule indiquée par `nCell` dans la sous-ligne actuelle `AppOutputLine` indiquée par `nLine`

#### Signature

```
GetCellSymbol(in nLine:Long, in nCell:Long) -> AppOutputLineSymbol
```

#### Paramètres

| Nom | Type | Description |
|---|---|---|
| **nLine** | **Long** | Spécifie l'index de la ligne basé sur zéro. |
| **nCell** | **Long** | Spécifie l'index de la cellule basé sur zéro. |

### Erreurs

| Code d'erreur | Description |
|---|---|
| 4100 | L'objet n'est plus valide. |
| 4101 | L'adresse invalide pour le paramètre de renvoi a été spécifiée. |

## *17.7.1.2.2.4 GetCellText*

Obtient le texte de la cellule indiquée par `nCell` dans la sous-ligne actuelle `AppOutputLine` indiquée par `nLine`

### Signature

```
GetCellText(in nLine:Long, in nCell:Long) -> String
```

### Paramètres

| Nom | Type | Description |
|---|---|---|
| **nLine** | `Long` | Spécifie l'index de la ligne basé sur zéro. |
| **nCell** | `Long` | Spécifie l'index de la cellule basé sur zéro. |

### Erreurs

| Code d'erreur | Description |
|---|---|
| 4100 | L'objet n'est plus valide. |
| 4101 | L'adresse invalide pour le paramètre de renvoi a été spécifiée. |

## *17.7.1.2.2.5 GetCellTextDecoration*

Obtient la décoration de la cellule de texte indiquée par `nCell` dans la sous-ligne actuelle `AppOutputLine`indiquée par `nLine`. Il peut s'agir d'une des valeurs `ENUMAppOutputLine_TextDecoration`.

### Signature

```
GetCellTextDecoration(in nLine:Long, in nCell:Long) -> Long
```

### Paramètres

| Nom | Type | Description |
|---|---|---|
| **nLine** | **Long** | Spécifie l'index de la ligne basé sur zéro. |
| **nCell** | **Long** | Spécifie l'index de la cellule basé sur zéro. |

### Erreurs

| Code d'erreur | Description |
|---|---|
| 4100 | L'objet n'est plus valide. |
| 4101 | L'adresse invalide pour le paramètre de renvoi a été spécifiée. |

## *17.7.1.2.2.6 GetIsCellText*

Renvoie true si la cellule indiquée par `nCell` dans la sous-ligne actuelle `AppOutputLine` indiquée par `nLine` est une cellule de texte.

### Signature

```
GetIsCellText(in nLine:Long, in nCell:Long) -> Boolean
```

### Paramètres

| Nom | Type | Description |
|---|---|---|
| **nLine** | **Long** | Spécifie l'index de la ligne basé sur zéro. |
| **nCell** | **Long** | Spécifie l'index de la cellule basé sur zéro. |

### Erreurs

| Code d'erreur | Description |
|---|---|
| 4100 | L'objet n'est plus valide. |
| 4101 | L'adresse invalide pour le paramètre de renvoi a été spécifiée. |

### *17.7.1.2.2.7 GetLineCount*

Obtient le nombre de sous-lignes dont la ligne actuelle est composée.

#### Signature

```
GetLineCount() -> Long
```

#### Erreurs

| Code d'erreur | Description |
|---|---|
| 4100 | L'objet n'est plus valide. |
| 4101 | L'adresse invalide pour le paramètre de renvoi a été spécifiée. |

### *17.7.1.2.2.8 GetLineSeverity*

Obtient la gravité de la ligne. Il peut s'agir d'une des valeurs `ENUMAppOutputLine_Severity`.

#### Signature

```
GetLineSeverity() -> Long
```

#### Erreurs

| Code d'erreur | Description |
|---|---|
| 4100 | L'objet n'est plus valide. |
| 4101 | L'adresse invalide pour le paramètre de renvoi a été spécifiée. |

### *17.7.1.2.2.9 GetLineSymbol*

Obtient le symbole assigné à toute la ligne.

#### Signature

```
GetLineSymbol() -> AppOutputLineSymbol
```

### Erreurs

| Code d'erreur | Description |
|---|---|
| 4100 | L'objet n'est plus valide. |
| 4101 | L'adresse invalide pour le paramètre de renvoi a été spécifiée. |

## *17.7.1.2.2.10 GetLineText*

Obtient les contenus de la ligne en tant que texte.

### Signature

```
GetLineText() -> String
```

### Erreurs

| Code d'erreur | Description |
|---|---|
| 4100 | L'objet n'est plus valide. |
| 4101 | L'adresse invalide pour le paramètre de renvoi a été spécifiée. |

## *17.7.1.2.2.11 GetLineTextEx*

Obtient les contenus de la ligne en tant que texte utilisant la partie spécifiée et les séparateurs de lignes.

### Signature

```
GetLineTextEx(in psTextPartSeperator:String, in psLineSeperator:String) -> String
```

### Paramètres

| Nom | Type | Description |
|---|---|---|
| **psTextPartSeperator** | `String` | |
| **psLineSeperator** | `String` | |

### Erreurs

| Code d'erreur | Description |
|---|---|
| 4100 | L'objet n'est plus valide. |

| Code d'erreur | Description |
|---|---|
| 4101 | L'adresse invalide pour le paramètre de renvoi a été spécifiée. |

### *17.7.1.2.2.12 GetLineTextWithChildren*

Obtient les contenus de la ligne, y compris toutes les lignes descendantes et enfant en tant que texte.

#### Signature

```
GetLineTextWithChildren() -> String
```

#### Erreurs

| Code d'erreur | Description |
|---|---|
| 4100 | L'objet n'est plus valide. |
| 4101 | L'adresse invalide pour le paramètre de renvoi a été spécifiée. |

### *17.7.1.2.2.13 GetLineTextWithChildrenEx*

Obtient les contenus de la ligne, y compris toutes les lignes descendantes et enfant en tant que texte utilisant la partie, la ligne, l'onglet et les séparateurs d'item spécifiés.

#### Signature

```
GetLineTextWithChildrenEx(in psPartSep:String, in psLineSep:String, in psTabSep:String,
in psItemSep:String) -> String
```

#### Paramètres

| Nom | Type | Description |
|---|---|---|
| **psPartSep** | String | |
| **psLineSep** | String | |
| **psTabSep** | String | |
| **psItemSep** | String | |

### Erreurs

| Code d'erreur | Description |
|---|---|
| 4100 | L'objet n'est plus valide. |
| 4101 | L'adresse invalide pour le paramètre de renvoi a été spécifiée. |

## 17.7.1.3 AppOutputLines

Représente une collection de lignes de message `AppOutputLine`.

Propriétés pour parcourir le modèle d'objet :

- `Application`
- `Parent`

Itération à travers la collection :

- `Compter`
- `Item`

### Propriétés

| Nom | Description |
|---|---|
| `Application` 1165 | Lecture seule.<br>Extrait l'objet de l'application de premier niveau. |
| `Compter` 1165 | Lecture seule.<br>Extrait le nombre de lignes dans la collection. |
| `Item` 1166 | Lecture seule.<br>Extrait la ligne à l'index `n` de la collection. Les index commencent par 1. |
| `Parent` 1166 | Lecture seule.<br>L'objet parent selon le modèle d'objet. |

### 17.7.1.3.1 Propriétés

#### *17.7.1.3.1.1 Application*

Extrait l'objet de l'application de premier niveau.

#### Signature

```
Application : Application
```

#### Erreurs

| Code d'erreur : | Description |
|---|---|
| 4000 | L'objet n'est plus valide. |
| 4001 | L'adresse invalide pour le paramètre de retour a été spécifiée. |

#### *17.7.1.3.1.2 Count*

Extrait le nombre de lignes dans la collection.

#### Signature

```
Count : Entier
```

#### Erreurs

| Code d'erreur | Description |
|---|---|
| 4000 | L'objet n'est plus valide. |
| 4001 | L'adresse invalide pour le paramètre de renvoi a été spécifiée. |

#### *17.7.1.3.1.3 Item*

Extrait la ligne à l'index `n` de la collection. Les index commencent par 1.

### Signature

```
Item(in n:Integer) : AppOutputLine
```

### Erreurs

| Code d'erreur | Description |
|---|---|
| 4000 | L'objet n'est plus valide. |
| 4001 | L'adresse invalide pour le paramètre de renvoi a été spécifiée. |

#### *17.7.1.3.1.4 Parent*

L'objet parent selon le modèle d'objet.

### Signature

```
Parent : AppOutputLine
```

### Erreurs

| Code d'erreur | Description |
|---|---|
| 4000 | L'objet n'est plus valide. |
| 4001 | L'adresse invalide pour le paramètre de renvoi a été spécifiée. |

## 17.7.1.4 AppOutputLineSymbol

Un `AppOutputLineSymbol` représente un lien dans une ligne de message `AppOutputLine` sur laquelle vous pouvez cliquer dans la fenêtre des messages de MapForce. Il est appliqué à une cellule d'une `AppOutputLine` ou à toute la ligne elle-même.

Propriétés pour parcourir le modèle d'objet :

- `Application`
- `Parent`

Accès aux méthodes AppOutputLineSymbol :

- `GetSymbolHREF`

- GetSymbolID
- IsSymbolHREF

## Propriétés

| Nom | Description |
|---|---|
| Application 1167 | Lecture seule.<br>Extrait l'objet de l'application de premier niveau. |
| Parent 1168 | Lecture seule.<br>L'objet parent selon le modèle d'objet. |

## Méthodes

| Nom | Description |
|---|---|
| GetSymbolHREF 1168 | Si le symbole est de type URL, il renvoie l'URL en tant que string. |
| GetSymbolID 1169 | Obtient l'ID du symbole. |
| IsSymbolHREF 1169 | Renvoie true si le symbole est de type URL. |

### 17.7.1.4.1 Propriétés

#### *17.7.1.4.1.1 Application*

Extrait l'objet de l'application de premier niveau.

## Signature

```
Application : Application
```

## Erreurs

| Code d'erreur : | Description |
|---|---|
| 4200 | L'objet n'est plus valide. |
| 4201 | L'adresse invalide pour le paramètre de retour a été spécifiée. |

#### *17.7.1.4.1.2 Parent*

L'objet parent selon le modèle d'objet.

### Signature

```
Parent : Application
```

### Erreurs

| Code d'erreur | Description |
|---|---|
| 4200 | L'objet n'est plus valide. |
| 4201 | L'adresse invalide pour le paramètre de renvoi a été spécifiée. |

## 17.7.1.4.2 Méthodes

#### *17.7.1.4.2.1 GetSymbolHREF*

Si le symbole est de type URL, il renvoie l'URL en tant que string.

### Signature

```
GetSymbolHREF() -> String
```

### Erreurs

| Code d'erreur | Description |
|---|---|
| 4200 | L'objet n'est plus valide. |
| 4201 | L'adresse invalide pour le paramètre de renvoi a été spécifiée. |

#### *17.7.1.4.2.2 GetSymbolID*

Obtient l'ID du symbole.

### Signature

```
GetSymbolID() -> Long
```

### Erreurs

| Code d'erreur | Description |
|---|---|
| 4200 | L'objet n'est plus valide. |
| 4201 | L'adresse invalide pour le paramètre de renvoi a été spécifiée. |

#### *17.7.1.4.2.3 IsSymbolHREF*

Renvoie true si le symbole est de type URL.

### Signature

```
IsSymbolHREF() -> Boolean
```

### Erreurs

| Code d'erreur | Description |
|---|---|
| 4200 | L'objet n'est plus valide. |
| 4201 | L'adresse invalide pour le paramètre de renvoi a été spécifiée. |

## 17.7.1.5 Composant

Un `composant` représente un composant de MapForce.

Pour accéder au contrôle, utilisez les propriétés Application `et` Parent .

Propriétés de composant :

- `HasIncomingConnections`
- `HasOutgoingConnections`
- `CanChangeInputInstanceFile`
- `CanChangeOutputInstanceFile`
- `ComponentName`.
- `ID`

- IsParameterInputRequired
- IsParameterSequence
- Nom
- Aperçu
- Schéma
- Sous-type
- Type

Propriétés relatives à l'instance :

- InputInstanceFile
- OutputInstanceFile

Datapoints :

- GetRootDatapoint

Exécution :

- GenerateOutput

## Propriétés

| Nom | Description |
|---|---|
| Application[1172] | Lecture seule.<br>Extrait l'objet de l'application de premier niveau. |
| CanChangeInputInstanceFile[1173] | Lecture seule.<br>Indique si le nom de fichier de l'instance de sortie peut être modifié.<br><br>Renvoie false si le composant contient un nœud filename et ce nœud a une connexion de son côté gauche (entrée), autrement il renvoie true. Si le composant ne contient pas de nœud filename, false est renvoyé. |
| CanChangeOutputInstanceFile[1173] | Lecture seule.<br>Indique si le nom de fichier de l'instance de sortie peut être modifié.<br><br>Renvoie false si le composant contient un nœud filename et ce nœud a une connexion de son côté gauche (entrée), autrement il renvoie true.<br>Si le composant ne contient pas de nœud filename, false est renvoyé. |
| ComponentName[1174] | Obtient ou définit le nom de composant. |
| HasIncomingConnections[1174] | Lecture seule.<br>Indique si le composant contient des connexions entrantes (de son côté gauche) ne comprenant pas le nœud filename. Une |

| Nom | Description |
|---|---|
| | connexion entrante sur le nœud de connexion n'a pas d'effet sur la valeur renvoyée. |
| HasOutgoingConnections 1175 | Lecture seule.<br>Indique si le composant contient des connexions sortantes (de son côté droit). |
| ID 1175 | Lecture seule.<br>Extrait l'ID de composant. |
| InputInstanceFile 1175 | Obtient ou définit le fichier de l'instance d'entrée du composant. |
| IsParameterInputRequired 1176 | Obtient ou définit si le composant de paramètre d'entrée requiert une connexion ingoing sur le composant d'appel de la fonction définie par l'utilisateur dans lequel se trouve le composant de paramètre d'entrée. Cette propriété ne fonctionne que pour les composants qui sont des composants de paramètres d'entrée. |
| IsParameterSequence 1176 | Obtient ou définit si le composant de paramètre d'entrée ou de sortie prend en charge les séquences. Cette propriété fonctionne uniquement pour les composants qui sont des composants de paramètre d'entrée ou de sortie. |
| Nom 1177 | Lecture seule.<br>Obtient le nom de composant. |
| OutputInstanceFile 1177 | Obtient ou définit le fichier d'instance de sortie du composant.<br><br>Tenter d'accéder à `OutputInstanceFile` d'un composant via l'API ne renvoie aucune donnée si le « fichier » connecteur du composant a été connecté à un autre item dans le mappage. |
| Parent 1178 | Lecture seule.<br>L'objet parent selon le modèle d'objet. |
| Aperçu 1178 | Obtient ou définit si le composant est le composant preview actuel.<br><br>Cette propriété ne fonctionne que pour les composants qui sont des composants cibles dans le mappage principal du document. Uniquement un composant cible dans le mappage principal peut être le composant preview à tout moment.<br><br>Lorsque vous définissez cette propriété, il est uniquement possible de la définir en tant que true. Ceci définira aussi implicitement la propriété `Preview` de tous les autres composants comme false.<br><br>Il n'existe qu'un seul composant cible unique dans le mappage principal, il s'agit également du composant preview. |
| Schéma 1179 | Lecture seule. |

| Nom | Description |
|---|---|
| | Extrait le nom de fichier du schéma du composant. |
| Sous-type 1179 | Lecture seule.<br>Extrait le sous-type du composant. |
| Type 1179 | Lecture seule.<br>Extrait le type de composant. |
| UsageKind 1180 | Lecture seule.<br>Extrait le type d'utilisation du composant. |

### Méthodes

| Nom | Description |
|---|---|
| GenerateOutput 1180 | Génère le(s) fichier(s) de sortie défini(s) dans le mappage pour le composant actuel uniquement, en utilisant un langage de mappage interne de MapForce. Le(s) nom(s) du/des fichier(s) de sortie est/sont défini(s) comme propriété du composant actuel qui est l'item de sortie dans le mappage pour le processus de génération. |
| GetRootDatapoint 1181 | Obtient un datapoint racine du côté gauche (entrée) ou droit (sortie) du composant. Pour accéder aux enfants et descendants, l'objet `Datapoint` fournit des méthodes supplémentaires. |

## 17.7.1.5.1 Propriétés

### *17.7.1.5.1.1 Application*

Extrait l'objet de l'application de premier niveau.

### Signature

```
Application : Application
```

### Erreurs

| Code d'erreur : | Description |
|---|---|
| 1200 | L'objet n'est plus valide. |
| 1201 | L'adresse invalide pour le paramètre de retour a été spécifiée. |

### *17.7.1.5.1.2 CanChangeInputInstanceFile*

Indique si le nom de fichier de l'instance de sortie peut être modifié.

Renvoie false si le composant contient un nœud filename et ce nœud a une connexion de son côté gauche (entrée), autrement il renvoie true. Si le composant ne contient pas de nœud filename, false est renvoyé.

#### Signature

```
CanChangeInputInstanceFile : Booléen
```

#### Erreurs

| Code d'erreur | Description |
|---|---|
| 1200 | L'objet n'est plus valide. |
| 1201 | L'adresse invalide pour le paramètre de retour a été spécifiée. |

### *17.7.1.5.1.3 CanChangeOutputInstanceFile*

Indique si le nom de fichier de l'instance de sortie peut être modifié.

Renvoie false si le composant contient un nœud filename et ce nœud a une connexion de son côté gauche (entrée), autrement il renvoie true.
Si le composant ne contient pas de nœud filename, false est renvoyé.

#### Signature

```
CanChangeOutputInstanceFile : Booléen
```

#### Erreurs

| Code d'erreur | Description |
|---|---|
| 1200 | L'objet n'est plus valide. |
| 1201 | L'adresse invalide pour le paramètre de retour a été spécifiée. |

#### 17.7.1.5.1.4 ComponentName

Obtient ou définit le nom de composant.

### Signature

```
ComponentName : String
```

### Erreurs

| Code d'erreur | Description |
|---|---|
| 1200 | L'objet n'est plus valide. |
| 1201 | L'adresse invalide pour le paramètre de renvoi a été spécifiée. |
| 1246 | Le composant ne prend pas en charge la définition de son nom. |
| 1247 | Nom de composant invalide. |

#### 17.7.1.5.1.5 HasIncomingConnections

Indique si le composant contient des connexions entrantes (de son côté gauche) ne comprenant pas le nœud filename. Une connexion entrante sur le nœud de connexion n'a pas d'effet sur la valeur renvoyée.

### Signature

```
HasIncomingConnections : Booléenne
```

### Erreurs

| Code d'erreur | Description |
|---|---|
| 1200 | L'objet n'est plus valide. |
| 1201 | L'adresse invalide pour le paramètre de renvoi a été spécifiée. |

### *17.7.1.5.1.6 HasOutgoingConnections*

Indique si le composant contient des connexions sortantes (de son côté droit).

#### Signature

```
HasOutgoingConnections : Booléenne
```

#### Erreurs

| Code d'erreur | Description |
|---|---|
| 1200 | L'objet n'est plus valide. |
| 1201 | L'adresse invalide pour le paramètre de renvoi a été spécifiée. |

### *17.7.1.5.1.7 ID*

Extrait l'ID de composant.

#### Signature

```
ID : Long
```

#### Erreurs

| Code d'erreur | Description |
|---|---|
| 1200 | L'objet n'est plus valide. |
| 1201 | L'adresse invalide pour le paramètre de renvoi a été spécifiée. |

### *17.7.1.5.1.8 InputInstanceFile*

Obtient ou définit le fichier de l'instance d'entrée du composant.

#### Signature

```
InputInstanceFile : String
```

### Erreurs

| Code d'erreur | Description |
|---|---|
| 1200 | L'objet n'est plus valide. |
| 1201 | L'adresse invalide pour le paramètre de renvoi a été spécifiée. |

## *17.7.1.5.1.9 IsParameterInputRequired*

Obtient ou définit si le composant de paramètre d'entrée requiert une connexion ingoing sur le composant d'appel de la fonction définie par l'utilisateur dans lequel se trouve le composant de paramètre d'entrée. Cette propriété ne fonctionne que pour les composants qui sont des composants de paramètres d'entrée.

### Signature

```
IsParameterInputRequired : Booléenne
```

### Erreurs

| Code d'erreur | Description |
|---|---|
| 1200 | L'objet n'est plus valide. |
| 1201 | L'adresse invalide pour le paramètre de renvoi a été spécifiée. |
| 1232 | Cette opération ne fonctionne que pour un composant de paramètre d'entrée. |
| 1240 | Il n'est pas autorisé de modifier le document. Il est en lecture seule. |

## *17.7.1.5.1.10 IsParameterSequence*

Obtient ou définit si le composant de paramètre d'entrée ou de sortie prend en charge les séquences. Cette propriété fonctionne uniquement pour les composants qui sont des composants de paramètre d'entrée ou de sortie.

### Signature

```
IsParameterSequence : Booléenne
```

### Erreurs

| Code d'erreur | Description |
|---|---|
| 1200 | L'objet n'est plus valide. |

| Code d'erreur | Description |
|---|---|
| 1201 | L'adresse invalide pour le paramètre de renvoi a été spécifiée. |
| 1233 | Cette opération ne fonctionne que pour un composant de paramètre d'entrée ou de sortie. |
| 1240 | Il n'est pas autorisé de modifier le document. Il est en lecture seule. |

### *17.7.1.5.1.11 Nom*

Obtient le nom de composant.

#### Signature

```
Nom : String
```

#### Erreurs

| Code d'erreur | Description |
|---|---|
| 1200 | L'objet n'est plus valide. |
| 1201 | L'adresse invalide pour le paramètre de renvoi a été spécifiée. |

### *17.7.1.5.1.12 OutputInstanceFile*

Obtient ou définit le fichier d'instance de sortie du composant.

Tenter d'accéder à `OutputInstanceFile` d'un composant via l'API ne renvoie aucune donnée si le « fichier » connecteur du composant a été connecté à un autre item dans le mappage.

#### Signature

```
OutputInstanceFile : String
```

#### Erreurs

| Code d'erreur | Description |
|---|---|
| 1200 | L'objet n'est plus valide. |
| 1201 | L'adresse invalide pour le paramètre de renvoi a été spécifiée. |

### 17.7.1.5.1.13 Parent

L'objet parent selon le modèle d'objet.

#### Signature

```
Parent : Mappage
```

#### Erreurs

| Code d'erreur | Description |
|---|---|
| 1200 | L'objet n'est plus valide. |
| 1201 | L'adresse invalide pour le paramètre de renvoi a été spécifiée. |

### 17.7.1.5.1.14 Preview

Obtient ou définit si le composant est le composant preview actuel.

Cette propriété ne fonctionne que pour les composants qui sont des composants cibles dans le mappage principal du document. Uniquement un composant cible dans le mappage principal peut être le composant preview à tout moment.

Lorsque vous définissez cette propriété, il est uniquement possible de la définir en tant que true. Ceci définira aussi implicitement la propriété `Preview` de tous les autres composants comme false.

Il n'existe qu'un seul composant cible unique dans le mappage principal, il s'agit également du composant preview.

#### Signature

```
Preview : Booléenne
```

#### Erreurs

| Code d'erreur | Description |
|---|---|
| 1200 | L'objet n'est plus valide. |
| 1201 | L'adresse invalide pour le paramètre de renvoi a été spécifiée. |
| 1234 | Seul un composant cible dans le mappage principal peut être défini comme composant preview . |
| 1235 | Un composant ne peut pas être défini comme composant non-preview. Définir un autre composant comme composant preview à la place. |

### *17.7.1.5.1.15 Schéma*

Extrait le nom de fichier du schéma du composant.

#### Signature

```
Schéma : String
```

#### Erreurs

| Code d'erreur | Description |
|---|---|
| 1200 | L'objet n'est plus valide. |
| 1201 | L'adresse invalide pour le paramètre de renvoi a été spécifiée. |

### *17.7.1.5.1.16 SubType*

Extrait le sous-type du composant.

#### Signature

```
SubType : ENUMComponentSubType 1286
```

#### Erreurs

| Code d'erreur | Description |
|---|---|
| 1200 | L'objet n'est plus valide. |
| 1201 | L'adresse invalide pour le paramètre de renvoi a été spécifiée. |

### *17.7.1.5.1.17 Type*

Extrait le type de composant.

#### Signature

```
Type : ENUMComponentType 1286
```

#### *17.7.1.5.1.18 UsageKind*

Extrait le type d'utilisation du composant.

### Signature

```
UsageKind : ENUMComponentUsageKind
```

### Erreurs

| Code d'erreur | Description |
|---|---|
| 1200 | L'objet n'est plus valide. |
| 1201 | L'adresse invalide pour le paramètre de renvoi a été spécifiée. |

## 17.7.1.5.2 Méthodes

#### *17.7.1.5.2.1 GenerateOutput*

Génère le(s) fichier(s) de sortie défini(s) dans le mappage pour le composant actuel uniquement, en utilisant un langage de mappage interne de MapForce. Le(s) nom(s) du/des fichier(s) de sortie est/sont défini(s) comme propriété du composant actuel qui est l'item de sortie dans le mappage pour le processus de génération.

### Signature

```
GenerateOutput(out pbError:Boolean) -> AppOutputLines
```

### Paramètres

| Nom | Type | Description |
|---|---|---|
| **pbError** | **Booléenne** | Ceci est un paramètre output-only. Vous recevrez une valeur uniquement si le langage d'appel prend en charge les paramètres de sortie. Si ce n'est pas le cas, la valeur que vous transférez ici restera inchangée quand la fonction est terminée. |

### Erreurs

| Code d'erreur | Description |
|---|---|
| 1200 | L'objet n'est plus valide. |
| 1201 | L'adresse invalide pour le paramètre de renvoi a été spécifiée. |
| 1248 | Générer la sortie est uniquement pris en charge quand l'interface utilisateur graphique est visible. |

## *17.7.1.5.2.2 GetRootDatapoint*

Obtient un datapoint racine du côté gauche (entrée) ou droit (sortie) du composant. Pour accéder aux enfants et descendants, l'objet `Datapoint` fournit des méthodes supplémentaires.

### Signature

```
GetRootDatapoint(in side:ENUMComponentDatapointSide, in strNamespace:String, in strLocalName:String, in strParameterName:String) -> Datapoint
```

### Paramètres

| Nom | Type | Description |
|---|---|---|
| **side** | `ENUMComponentDatapointSide` | Le paramètre side indique si un datapoint d'entrée ou de sortie d'un composant doit être extrait. |
| **strNamespace** | `String` | Le nom d'espace spécifié et nom local indiquent le nom spécifique du nœud dont le datapoint doit être extrait. Pour les composants avec une information structurelle telle que les composants de schéma, vous allez devoir fournir un espace de nom ensemble avec le nom local, ou vous pouvez uniquement transférer un string vide pour l'espace de nom.<br><br>Les composants basés sur fichier comme le composant de schéma contiennent un nœud spécial dans leur racine, le nœud de filename. Là, `GetRootDatapoint` peut uniquement trouver le nœud de filename. Vous allez devoir transférer l'espace de nom |

| Nom | Type | Description |
|---|---|---|
| | | "http://www.altova.com/mapforce" et le nom local "FileInstance" pour extraire le datapoint de ce nœud. |
| **strLocalName** | String | Voir ci-dessus. |
| **strParameterName** | String | Le nom du paramètre spécifié devrait être un string vide, sauf si le composant en question est un composant d'appel de fonction. Puisqu'une fonction définie par l'utilisateur peut contenir des paramètres d'entrée ou de sortie de la même structure, le composant d'appel de fonction appelant cette fonction définie par l'utilisateur peut avoir plus d'un nœud racine avec un espace de nom et nom local identiques.<br><br>Ils seront alors différents uniquement par leurs noms de paramètre, qui sont en fait des noms de composants de paramètre correspondants dans le mappage de fonction définie par l'utilisateur lui-même.<br><br>Par contre, il n'est pas obligatoire de spécifier le nom de paramètre. Dans ce cas, la méthode retournera le premier datapoint racine correspondant à l'espace de nom spécifié et nom local. |

## Erreurs

| Code d'erreur | Description |
|---|---|
| 1200 | L'objet n'est plus valide. |
| 1201 | L'adresse invalide pour le paramètre de renvoi a été spécifiée. |
| 1248 | Datapoint pas trouvé. |

## 17.7.1.6 Composants

Représente une collection d'objets de `composant` objects.

Pour accéder au modèle d'objet, utilisez les propriétés suivantes :

- `Application`
- `Parent`

Pour itérer à travers la collection :

- `Compter`
- `Item`

### Propriétés

| Nom | Description |
|---|---|
| `Application` 1183 | Lecture seule.<br>Extrait l'objet de l'application de premier niveau. |
| `Compter` 1184 | Lecture seule.<br>Extrait le nombre de composants dans la collection. |
| `Item` 1184 | Lecture seule.<br>Extrait le composant à l'index `n` de la collection. Les index commencent par 1. |
| `Parent` 1185 | Lecture seule.<br>L'objet parent selon le modèle d'objet. |

### 17.7.1.6.1 Propriétés

#### *17.7.1.6.1.1 Application*

Extrait l'objet de l'application de premier niveau.

### Signature

```
Application : Application
```

### Erreurs

| Code d'erreur : | Description |
|---|---|
| 1200 | L'objet n'est plus valide. |

| Code d'erreur : | Description |
|---|---|
| 1201 | L'adresse invalide pour le paramètre de retour a été spécifiée. |

#### *17.7.1.6.1.2 Count*

Extrait le nombre de composants dans la collection.

### Signature

```
Count : Entier
```

### Erreurs

| Code d'erreur | Description |
|---|---|
| 1200 | L'objet n'est plus valide. |
| 1201 | L'adresse invalide pour le paramètre de renvoi a été spécifiée. |

#### *17.7.1.6.1.3 Item*

Extrait le composant à l'index `n` de la collection. Les index commencent par 1.

### Signature

```
Item(in n:Integer) : Composant
```

### Erreurs

| Code d'erreur | Description |
|---|---|
| 1200 | L'objet n'est plus valide. |
| 1201 | L'adresse invalide pour le paramètre de renvoi a été spécifiée. |

#### *17.7.1.6.1.4 Parent*

L'objet parent selon le modèle d'objet.

### Signature

```
Parent : Mappage
```

### Erreurs

| Code d'erreur | Description |
|---|---|
| 1200 | L'objet n'est plus valide. |
| 1201 | L'adresse invalide pour le paramètre de renvoi a été spécifiée. |

## 17.7.1.7 Connexion

Un objet de `connexion` représente un connecteur entre deux composants.

Pour accéder au modèle d'objet, utilisez les propriétés suivantes :

- `Application`
- `Parent`

Pour obetnir ou définir le type de connexion, utilisez `ConnectionType`.

### Propriétés

| Nom | Description |
|---|---|
| `Application` 1186 | Lecture seule.<br>Extrait l'objet de l'application de premier niveau. |
| `ConnectionType` 1186 | Obtient ou définit le type de connexion. |
| `Parent` 1187 | Lecture seule.<br>L'objet parent selon le modèle d'objet. |

### 17.7.1.7.1 Propriétés

#### *17.7.1.7.1.1 Application*

Extrait l'objet de l'application de premier niveau.

#### Signature

```
Application : Application
```

#### Erreurs

| Code d'erreur : | Description |
|---|---|
| 2100 | L'objet n'est plus valide. |
| 2101 | L'adresse invalide pour le paramètre de retour a été spécifiée. |

#### *17.7.1.7.1.2 ConnectionType*

Obtient ou définit le type de connexion.

#### Signature

```
ConnectionType : ENUMConnectionType 1287
```

#### Erreurs

| Code d'erreur | Description |
|---|---|
| 2100 | L'objet n'est plus valide. |
| 2101 | L'adresse invalide pour le paramètre de renvoi a été spécifiée. |
| 2102 | Il n'est pas autorisé de modifier le document. Il est en lecture seule. |
| 2103 | Échec de la modification du type de connexion. |

#### *17.7.1.7.1.3 Parent*

L'objet parent selon le modèle d'objet.

### Signature

```
Parent : Mappage
```

### Erreurs

| Code d'erreur | Description |
|---|---|
| 2100 | L'objet n'est plus valide. |
| 2101 | L'adresse invalide pour le paramètre de renvoi a été spécifiée. |

## 17.7.1.8 Datapoint

Un objet `Datapoint` représente une icône d'entrée ou de sortie d'un composant.

### Propriétés

| Nom | Description |
|---|---|
| `Application` 1188 | Lecture seule.<br>Extrait l'objet de l'application de premier niveau. |
| `Parent` 1188 | Lecture seule.<br>L'objet parent selon le modèle d'objet. |

### Méthodes

| Nom | Description |
|---|---|
| `GetChild` 1188 | Recherche un datapoint enfant direct du datapoint actuel, par le biais de l'espace de nom et le nom local.<br><br>Si un composant de schéma a des éléments qui contiennent du contenu mixte, chacun affiche un nœud enfant additionnel, le soi-disant nœud **text()**. Pour extraire un datapoint d'un nœud **text()**, vous devrez transférer un string vide dans `strNamespace` de même que `"#text"` dans `strLocalName` et `eSearchDatapointElement` dans `searchFlags`. |

### 17.7.1.8.1 Propriétés

#### *17.7.1.8.1.1 Application*

Extrait l'objet de l'application de premier niveau.

#### Signature

```
Application : Application
```

#### Erreurs

| Code d'erreur : | Description |
|---|---|
| 2000 | L'objet n'est plus valide. |
| 2001 | L'adresse invalide pour le paramètre de retour a été spécifiée. |

#### *17.7.1.8.1.2 Parent*

L'objet parent selon le modèle d'objet.

#### Signature

```
Parent : Composant
```

#### Erreurs

| Code d'erreur | Description |
|---|---|
| 2000 | L'objet n'est plus valide. |
| 2001 | L'adresse invalide pour le paramètre de renvoi a été spécifiée. |

### 17.7.1.8.2 Méthodes

#### *17.7.1.8.2.1 GetChild*

Recherche un datapoint enfant direct du datapoint actuel, par le biais de l'espace de nom et le nom local.

Si un composant de schéma a des éléments qui contiennent du contenu mixte, chacun affiche un nœud enfant additionnel, le soi-disant nœud **text()**. Pour extraire un datapoint d'un nœud **text()**, vous devrez transférer un

string vide dans `strNamespace` de même que `"#text"` dans `strLocalName` et `eSearchDatapointElement` dans `searchFlags`.

### Signature

```
GetChild(in strNamespace:String, in strLocalName:String, in
searchFlags:ENUMSearchDatapointFlags1289) -> Datapoint
```

### Paramètres

| Nom | Type | Description |
|---|---|---|
| **strNamespace** | `String` | L'espace de nom du datapoint enfant direct. |
| **strLocalName** | `String` | Le nom du datapoint enfant direct. |
| **searchFlags** | `ENUMSearchDatapointFlags`1289 | Les indicateurs recherche peuvent être transférés comme une combinaison de valeurs (combiné utilisant binaire OR) de l'énumération `ENUMSearchDatapointFlags`. |

### Erreurs

| Code d'erreur | Description |
|---|---|
| 2000 | L'objet n'est plus valide. |
| 2001 | L'adresse invalide pour le paramètre de renvoi a été spécifiée. |
| 2002 | Datapoint pas trouvé. |

## 17.7.1.9 Document

Un objet de `Document` représente un document MapForce (un fichier MFD chargé). Un document contient un mappage principal et zéro ou plus de mappages définis par l'utilisateur local.

Pour accéder au modèle d'objet, utilisez les propriétés suivantes :

- `Application`
- `Parent`

Pour la gestion du fichier, utilisez :

- `Activer`
- `Fermer`
- `FullName`

- Nom
- Chemin d'accès
- Enregistré
- Enregistrer
- SaveAs

Pour la gestion du mappage, utilisez :

- MainMapping
- Mappages
- CreateUserDefinedFunction

Pour la gestion du composant, utilisez :

- FindComponentByID

Pour la génération de code, utilisez :

- OutputSettings_ApplicationName
- JavaSettings_BasePackageName
- GenerateCHashCode
- GenerateCodeEx
- GenerateCppCode
- GenerateJavaCode
- GenerateXQuery
- GenerateXSLT
- GenerateXSLT2
- GenerateXSLT3
- HighlightSerializedMarker

Pour l'exécution de mappage, utilisez :

- GenerateOutput
- GenerateOutputEx

Accès d'affichage :

- MapForceView

Obsolète :

- OutputSettings_Encoding

## Propriétés

| Nom | Description |
|---|---|
| Application 1194 | Lecture seule.<br>Extrait l'objet de l'application de premier niveau. |
| FullName 1194 | Chemin et nom du fichier du document. |
| JavaSettings_BasePackageName 1194 | Définit ou extrait le nom du pack de base utilisé lors de la génération du code Java. Dans l'interface utilisateur graphique |

| Nom | Description |
|---|---|
| | de MapForce, ce paramètre est disponible dans la boîte de dialogue des **Paramètres de nmappage** (clic droit sur le mappage et sélectionnez les **Mapping Settings** depuis le menu contextuel). |
| LibraryImports 1195 | Lecture seule.<br>Obtient une collection de bibliothèques importées. Dans l'interface utilisateur graphique de MapForce, celles-ci correspondent aux entrées de la fenêtre **Gérer les bibliothèques**, ajoutées au niveau du document. |
| MainMapping 1195 | Lecture seule.<br>Extrait le mappage principal du document. |
| MapForceView 1196 | Lecture seule.<br>Cette propriété donne accès à la fonctionnalité spécifique à l'affichage de MapForce. |
| Mappages 1196 | Lecture seule.<br>Retourne une collection de mappages contenue dans le document. |
| Nom 1196 | Lecture seule.<br>Nom du fichier document sans chemin de fichier. |
| OutputSettings_ApplicationName 1197 | Définit ou extrait le nom de l'application disponible dans la boîte de dialogue **Paramètres de mappage** (pour afficher cette boîte de dialogue dans MapForce, clic droit sur le mappage et sélectionnez **Paramètres de mappage** depuis le menu contextuel). |
| OutputSettings_Encoding 1197 | Cette propriété n'est plus prise en charge. Les paramètres d'encodage de sortie n'existent plus. Les composants ont des paramètres d'encodage de sortie individuels. |
| Parent 1198 | Lecture seule.<br>L'objet parent selon le modèle d'objet. |
| Chemin d'accès 1198 | Lecture seule.<br>Le chemin du fichier de document sans nom. |
| Enregistré 1198 | Lecture seule.<br>**True** si le document n'a pas été modifié depuis la dernière opération de sauvegarde, **false**, si le contraire. |

## Méthodes

| Nom | Description |
|---|---|
| Activate 1199 | Rend ce document actif. |

| Nom | Description |
|---|---|
| Fermer 1199 | Ferme le document sans l'enregistrer. |
| CreateUserDefinedFunction 1200 | Crée une fonction définie par l'utilisateur dans le document actuel. |
| FindComponentByID 1201 | Cherche dans le document entier, aussi dans tous ses mappages, pour le composant avec l'ID spécifiée. |
| GenerateCHashCode 1201 | Génère le code C# qui exécutera le mappage. Utilise les propriétés définies dans `Application.Options` pour configurer la génération de code. |
| GenerateCodeEx 1202 | Génère le code qui exécutera le mappage. Le paramètre **i_nLanguage** spécifie le langage cible. La méthode renvoie un objet qui peut être utilisé pour énumérer tous les messages créés par le générateur de code. Il s'agit des mêmes messages qui sont affichés dans la fenêtre de messages de MapForce. |
| GenerateCppCode 1202 | Génère le code C++ qui exécutera le mappage. Utilise les propriétés définies dans `Application.Options` pour configurer la génération de code. |
| GenerateJavaCode 1203 | Génère le code Java qui exécutera le mappage. Utilise les propriétés définies dans `Application.Options` pour configurer la génération de code. |
| GenerateOutput 1203 | Génère tous les fichiers de sortie définis dans le mappage utilisant un langage de mappage interne de MapForce. Les noms des fichiers de sortie sont définis comme propriétés des éléments de sortie dans le mappage.<br><br>**Note :** Cette méthode peut uniquement être utilisée quand la fenêtre principale de MapForce (exécutée comme serveur COM) est visible, ou si elle est incorporée avec une interface utilisateur graphique. Si la méthode est appelée tandis que MapForce n'est pas visible, alors l'erreur apparaîtra. |
| GenerateOutputEx 1204 | Génère tous les fichiers de sortie définis dans le mappage utilisant un langage de mappage interne de MapForce. Les noms des fichiers de sortie sont définis comme propriétés des éléments de sortie dans le mappage. Cette méthode est identique à `GenerateOutput`, à l'exception de sa valeur de renvoi contenant les messages résultants, avertissements et erreurs organisés en tant qu'arborescences de `AppOutputLines`.<br><br>**Note :** Cette méthode peut uniquement être utilisée quand la fenêtre principale de MapForce (exécutée comme serveur COM) est visible, ou si elle est incorporée avec une interface utilisateur graphique. Si la méthode est appelée tandis que MapForce n'est pas visible, alors l'erreur apparaîtra. |

| Nom | Description |
|---|---|
| GenerateXQuery 1204 | Génère du code de mappage comme XQuery. Utilise les propriétés définies dans `Application.Options` pour configurer la génération de code. |
| GenerateXSLT 1205 | Génère du code de mappage en tant que XSLT. Utilise les propriétés définies dans `Application.Options` pour configurer la génération de code. |
| GenerateXSLT2 1205 | Génère le code de mappage comme XSLT2. Utilise les propriétés définies dans `Application.Options` pour configurer la génération de code. |
| GenerateXSLT3 1205 | Génère un code de mappage XSLT 3.0. Utilise les propriétés définies dans `Application.Options` pour configurer la génération de code. |
| HighlightSerializedMarker 1206 | Utilisez cette méthode pour marquer un emplacement dans un fichier de mappage qui a été précédemment sérialisé. Si le document correspondant n'est pas déjà chargé, il sera chargé d'abord. Voir `GenerateCodeEx` pour une méthode d'extraction d'un marqueur sérialisé. |
| Enregistrer 1207 | Enregistre le document vers le fichier défini par `Document.FullName`. |
| SaveAs 1207 | Enregistre le document vers le nom de fichier spécifique, et définit `Document.FullName` vers cette valeur si l'opération Enregistrer a réussi. |

## Événements

| Nom | Description |
|---|---|
| OnDocumentClosed 1208 | Cet événement est déclenché lorsqu'un document est fermé. L'objet de document passé au gestionnaire d'événements ne devrait pas être accédé. L'événement ouvert correspondant est `Application.OnDocumentOpened`. |
| OnModifiedFlagChanged 1208 | Cet événement est déclenché lorsque le statut de modification d'un document change. |

### 17.7.1.9.1 Propriétés

#### *17.7.1.9.1.1 Application*

Extrait l'objet de l'application de premier niveau.

##### Signature

```
Application : Application
```

##### Erreurs

| Code d'erreur : | Description |
|---|---|
| 1200 | L'objet n'est plus valide. |
| 1201 | L'adresse invalide pour le paramètre de retour a été spécifiée. |

#### *17.7.1.9.1.2 FullName*

Chemin et nom du fichier du document.

##### Signature

```
FullName : String
```

##### Erreurs

| Error code | Description |
|---|---|
| 1200 | L'objet n'est plus valide. |
| 1201 | L'adresse invalide a été spécifiée pour le paramètre de renvoi. |

#### *17.7.1.9.1.3 JavaSettings_BasePackageName*

Définit ou extrait le nom du pack de base utilisé lors de la génération du code Java. Dans l'interface utilisateur graphique de MapForce, ce paramètre est disponible dans la boîte de dialogue des **Paramètres de nmappage** (clic droit sur le mappage et sélectionnez les **Mapping Settings** depuis le menu contextuel).

##### Signature

```
JavaSettings_BasePackageName : String
```

### Erreurs

| Code d'erreur | Description |
|---|---|
| 1200 | L'objet n'est plus valide. |
| 1201 | L'adresse invalide pour le paramètre de renvoi a été spécifiée. |

#### *17.7.1.9.1.4 LibraryImports*

Obtient une collection de bibliothèques importées. Dans l'interface utilisateur graphique de MapForce, celles-ci correspondent aux entrées depuis la fenêtre **Gérer les biblitohèques**, ajoutées au niveau du document.

### Signature

```
LibraryImports : LibraryImports
```

### Erreurs

| Code d'erreur | Description |
|---|---|
| 1200 | L'objet n'est plus valide. |
| 1201 | L'adresse invalide pour le paramètre de renvoi a été spécifiée. |

#### *17.7.1.9.1.5 MainMapping*

Extrait le mappage principal du document.

### Signature

```
MainMapping : Mappage
```

### Erreurs

| Code d'erreur | Description |
|---|---|
| 1200 | L'objet n'est plus valide. |
| 1201 | L'adresse invalide pour le paramètre de renvoi a été spécifiée. |

### 17.7.1.9.1.6 MapForceView

Cette propriété donne accès à la fonctionnalité spécifique à l'affichage de MapForce.

#### Signature

```
MapForceView : MapForceView
```

#### Erreurs

| Code d'erreur | Description |
|---|---|
| 1200 | L'objet n'est plus valide. |
| 1201 | L'adresse invalide pour le paramètre de renvoi a été spécifiée. |

### 17.7.1.9.1.7 Mappages

Retourne une collection de mappages contenue dans le document.

#### Signature

```
Mappages : Mappages
```

#### Erreurs

| Code d'erreur | Description |
|---|---|
| 1200 | L'objet n'est plus valide. |
| 1201 | L'adresse invalide pour le paramètre de renvoi a été spécifiée. |

### 17.7.1.9.1.8 Nom

Nom du fichier du document sans chemin de fichier.

#### Signature

```
Nom : String
```

### Erreurs

| Code d'erreur | Description |
|---|---|
| 1200 | L'objet n'est plus valide. |
| 1201 | L'adresse invalide pour le paramètre de renvoi a été spécifiée. |

## *17.7.1.9.1.9 OutputSettings_ApplicationName*

Définit ou extrait le nom de l'application disponible dans la boîte de dialogue **Paramètres de mappage** (pour afficher cette boîte de dialogue dans MapForce, clic droit sur le mappage et sélectionnez **Paramètres de mappage** depuis le menu contextuel).

### Signature

```
OutputSettings_ApplicationName : String
```

### Erreurs

| Code d'erreur | Description |
|---|---|
| 1200 | L'objet n'est plus valide. |
| 1201 | L'adresse invalide pour le paramètre de renvoi a été spécifiée. |

## *17.7.1.9.1.10 OutputSettings_Encoding (obsolete)*

Cette propriété n'est plus prise en charge. Les paramètres d'encodage de sortie n'existent plus. Les composants ont des paramètres d'encodage de sortie individuels.

### Signature

```
OutputSettings_Encoding : String
```

### 17.7.1.9.1.11 Parent

L'objet parent selon le modèle d'objet.

#### Signature

```
Parent : Documents
```

#### Erreurs

| Code d'erreur | Description |
|---|---|
| 1200 | L'objet n'est plus valide. |
| 1201 | L'adresse invalide pour le paramètre de renvoi a été spécifiée. |

### 17.7.1.9.1.12 Chemin

Le chemin du fichier de document sans nom.

#### Signature

```
Chemin : String
```

#### Erreurs

| Code d'erreur | Description |
|---|---|
| 1200 | L'objet n'est plus valide. |
| 1201 | L'adresse invalide pour le paramètre de renvoi a été spécifiée. |

### 17.7.1.9.1.13 Enregistré

**True** si le document n'a pas été modifié depuis la dernière opération de sauvegarde, **false**, si le contraire.

#### Signature

```
Enregistré : Booléenne
```

### Erreurs

| Code d'erreur | Description |
|---|---|
| 1200 | L'objet n'est plus valide. |
| 1201 | L'adresse invalide pour le paramètre de renvoi a été spécifiée. |

## 17.7.1.9.2 Méthodes

### *17.7.1.9.2.1 Activate*

Rend ce document actif.

### Signature

```
Activate() -> Void
```

### Erreurs

| Code d'erreur : | Description |
|---|---|
| 1200 | L'objet n'est plus valide. |

### *17.7.1.9.2.2 Fermer*

Ferme le document sans l'enregistrer.

### Signature

```
Fermer() -> Résilier
```

### Erreurs

| Code d'erreur | Description |
|---|---|
| 1200 | L'objet n'est plus valide. |
| 1201 | L'adresse invalide pour le paramètre de retour a été spécifiée. |

### *17.7.1.9.2.3 CreateUserDefinedFunction*

Crée une fonction définie par l'utilisateur dans le document actuel.

#### Signature

```
CreateUserDefinedFunction(dans strFunctionName:String, dans strLibraryName:String, dans
strSyntax:String, dans strDetails:String, dans bInlinedUse:Boolean) -> Mapping
```

#### Paramètres

| Nom | Type | Description |
|---|---|---|
| **strFunctionName** | `String` | Le nom de la fonction. |
| **strLibraryName** | `String` | Le nom de la bibliothèque à laquelle cette fonction appartient. |
| **strSyntax** | `String` | Un string qui décrit la syntaxe de cette fonction (à des fins d'information uniquement). |
| **strDetails** | `String` | Une description de cette fonction. |
| **bInlinedUse** | `Booléen` | Indicateur booléen qui spécifie si la fonction a une utilisation inlined. |

#### Erreurs

| Code d'erreur | Description |
|---|---|
| 1200 | L'objet n'est plus valide. |
| 1201 | L'adresse invalide pour le paramètre de renvoi a été spécifiée. |
| 1208 | Échec de la création d'une fonction définie par l'utilisateur. |
| 1209 | Il n'est pas autorisé de modifier le document. Il est en lecture seule. |

### *17.7.1.9.2.4 FindComponentByID*

Cherche dans le document entier, aussi dans tous ses mappages, pour le composant avec l'ID spécifiée.

#### Signature

```
FindComponentByID(in nID:Unsigned Long) -> Composant
```

#### Paramètres

| Nom | Type | Description |
|---|---|---|
| **nID** | `Unsigned Long` | L'ID du composant à rechercher. |

#### Erreurs

| Code d'erreur | Description |
|---|---|
| 1200 | L'objet n'est plus valide. |
| 1201 | L'adresse invalide a été spécifiée pour le paramètre de renvoi. |

### *17.7.1.9.2.5 GenerateCHashCode*

Génère le code C# qui exécutera le mappage. Utilise les propriétés définies dans `Application.Options` pour configurer la génération de code.

#### Signature

```
GenerateCHashCode() -> Void
```

#### Erreurs

| Code d'erreur | Description |
|---|---|
| 1200 | Cet objet n'est plus valide. |
| 1201 | L'adresse invalide a été spécifiée pour le paramètre de renvoi. |
| 1205 | Erreur pendant la génération de code. |

### 17.7.1.9.2.6 GenerateCodeEx

Génère le code qui exécutera le mappage. Le paramètre **i_nLanguage** spécifie le langage cible. La méthode renvoie un objet qui peut être utilisé pour énumérer tous les messages créés par le générateur de code. Il s'agit des mêmes messages qui sont affichés dans la fenêtre de messages de MapForce.

#### Signature

```
GenerateCodeEx(in i_nLanguage:ENUMProgrammingLanguage1288) -> ErrorMarkers
```

#### Paramètres

| Nom | Type | Description |
|---|---|---|
| **i_nLanguage** | ENUMProgrammingLanguage1288 | Spécifie le langage de génération de code cible. |

#### Errors

| Error code | Description |
|---|---|
| 1200 | L'objet n'est plus valide. |
| 1201 | L'adresse invalide a été spécifiée pour le paramètre de renvoi. |
| 1205 | Erreur pendant la génération de code |

### 17.7.1.9.2.7 GenerateCppCode

Génère le code C++ qui exécutera le mappage. Utilise les propriétés définies dans `Application.Options` pour configurer la génération de code.

#### Signature

```
GenerateCppCode() -> Void
```

#### Erreurs

| Code d'erreur | Description |
|---|---|
| 1200 | L'objet n'est plus valide. |
| 1201 | L'adresse invalide pour le paramètre de renvoi a été spécifiée. |
| 1205 | Erreur pendant la génération de code |

### *17.7.1.9.2.8 GenerateJavaCode*

Génère le code Java qui exécutera le mappage. Utilise les propriétés définies dans `Application.Options` pour configurer la génération de code.

#### Signature

```
GenerateJavaCode() -> Void
```

#### Erreurs

| Code d'erreur | Description |
|---|---|
| 1200 | L'objet n'est plus valide. |
| 1201 | L'adresse invalide pour le paramètre de renvoi a été spécifiée. |
| 1205 | Erreur pendant la génération de code |

### *17.7.1.9.2.9 GenerateOutput*

Génère tous les fichiers de sortie définis dans le mappage utilisant un langage de mappage interne de MapForce. Les noms des fichiers de sortie sont définis comme propriétés des éléments de sortie dans le mappage.

**Note :** Cette méthode peut uniquement être utilisée quand la fenêtre principale de MapForce (exécutée comme serveur COM) est visible, ou si elle est incorporée avec une interface utilisateur graphique. Si la méthode est appelée tandis que MapForce n'est pas visible, alors l'erreur apparaîtra.

#### Signature

```
GenerateOutput() -> Void
```

#### Erreurs

| Code d'erreur | Description |
|---|---|
| 1200 | L'objet n'est plus valide. |
| 1201 | L'adresse invalide pour le paramètre de renvoi a été spécifiée. |
| 1206 | Erreur durant l'exécution de l'algorithme de mappage. |
| 1210 | Générer la sortie est uniquement pris en charge quand l'interface utilisateur graphique est visible. |

### *17.7.1.9.2.10 GenerateOutputEx*

Génère tous les fichiers de sortie définis dans le mappage utilisant un langage de mappage interne de MapForce. Les noms des fichiers de sortie sont définis comme propriétés des éléments de sortie dans le mappage. Cette méthode est identique à `GenerateOutput`, à l'exception de sa valeur de renvoi contenant les messages résultants, avertissements et erreurs organisés en tant qu'arborescences de `AppOutputLines`.

**Note :** Cette méthode peut uniquement être utilisée quand la fenêtre principale de MapForce (exécutée comme serveur COM) est visible, ou si elle est incorporée avec une interface utilisateur graphique. Si la méthode est appelée tandis que MapForce n'est pas visible, alors l'erreur apparaîtra.

#### Signature

```
GenerateOutputEx() -> AppOutputLines
```

#### Erreurs

| Code d'erreur | Description |
|---|---|
| 1200 | L'objet n'est plus valide. |
| 1201 | L'adresse invalide pour le paramètre de renvoi a été spécifiée. |
| 1206 | Erreur durant l'exécution de l'algorithme de mappage. |
| 1210 | Générer la sortie est uniquement pris en charge quand l'interface utilisateur graphique est visible. |

### *17.7.1.9.2.11 GenerateXQuery*

Génère du code de mappage comme XQuery. Utilise les propriétés définies dans `Application.Options` pour configurer la génération de code.

#### Signature

```
GenerateXQuery() -> Void
```

#### Erreurs

| Code d'erreur | Description |
|---|---|
| 1200 | L'objet n'est plus valide. |
| 1201 | L'adresse invalide pour le paramètre de renvoi a été spécifiée. |
| 1204 | Erreur durant la génération de code XSLT/XSLT2/XQuery. |

### *17.7.1.9.2.12 GenerateXSLT*

Génère du code de mappage en tant que XSLT. Utilise les propriétés définies dans `Application.Options` pour configurer la génération de code.

#### Signature

```
GenerateXSLT() -> Void
```

#### Erreurs

| Code d'erreur | Description |
|---|---|
| 1200 | L'objet n'est plus valide. |
| 1201 | L'adresse invalide pour le paramètre de renvoi a été spécifiée. |
| 1204 | Erreur durant la génération de code XSLT/XSLT2/XQuery. |

### *17.7.1.9.2.13 GenerateXSLT2*

Génère le code de mappage comme XSLT2. Utilise les propriétés définies dans `Application.Options` pour configurer la génération de code.

#### Signature

```
GenerateXSLT2() -> Void
```

#### Erreurs

| Code d'erreur | Description |
|---|---|
| 1200 | L'objet n'est plus valide. |
| 1201 | L'adresse invalide pour le paramètre de renvoi a été spécifiée. |
| 1204 | Erreur durant la génération de code XSLT/XSLT2/XQuery. |

### *17.7.1.9.2.14 GenerateXSLT3*

Génère le code de mappage comme XSLT 3.0. Utilise les propriétés définies dans `Application.Options` pour configurer la génération de code.

### Signature

```
GenerateXSLT3() -> Void
```

### Errors

| Code d'erreur | Description |
|---|---|
| 1200 | L'objet n'est plus valide. |
| 1201 | L'adresse invalide pour le paramètre de renvoi a été spécifiée. |
| 1204 | Erreur durant la génération de code XSLT/XSLT2/XSLT3/XQuery. |

## *17.7.1.9.2.15 HighlightSerializedMarker*

Utilisez cette méthode pour marquer un emplacement dans un fichier de mappage qui a été précédemment sérialisé. Si le document correspondant n'est pas déjà chargé, il sera chargé d'abord. Voir `GenerateCodeEx` pour une méthode d'extraction d'un marqueur sérialisé.

### Signature

```
HighlightSerializedMarker(dans i_strSerializedMarker:String) -> Void
```

### Paramètres

| Nom | Type | Description |
|---|---|---|
| **i_strSerializedMarker** | `String` | L'objet `ErrorMarker` pour mettre en évidence. Utiliser `ErrorMaker.Serialized` pour obtenir cette valeur. |

### Erreurs

| Code d'erreur | Description |
|---|---|
| 1000 | L'objet n'est plus valide. |
| 1001 | L'adresse invalide pour le paramètre de renvoi a été spécifiée. |
| 1007 | Le string passé dans `i_strSerializedMarker` n'est pas reconnu marqueur MapForce sérialisé. |
| 1008 | Le marqueur pointe vers un emplacement qui n'est plus valide. |

### *17.7.1.9.2.16 Enregistrer*

Enregistre le document vers le fichier défini par `Document.FullName`.

#### Signature

```
Enregistrer() -> Void
```

#### Erreurs

| Code d'erreur | Description |
|---|---|
| 1200 | L'objet n'est plus valide. |
| 1201 | L'adresse invalide pour le paramètre de renvoi a été spécifiée. |

### *17.7.1.9.2.17 SaveAs*

Enregistre le document vers le nom de fichier spécifique, et définit `Document.FullName` vers cette valeur si l'opération Enregistrer a réussi.

#### Signature

```
SaveAs(in i_strFileName:String) -> Void
```

#### Paramètres

| Nom | Type | Description |
|---|---|---|
| **i_strFileName** | `String` | Spécifie le chemin où le document doit être enregistré. |

#### Erreurs

| Code d'erreur | Description |
|---|---|
| 1200 | L'objet n'est plus valide. |
| 1201 | L'adresse invalide pour le paramètre de renvoi a été spécifiée. |

### 17.7.1.9.3 Événements

#### *17.7.1.9.3.1 OnDocumentClosed*

Cet événement est déclenché lorsqu'un document est fermé. L'objet de document passé au gestionnaire d'événements ne devrait pas être accédé. L'événement ouvert correspondant est `Application.OnDocumentOpened`.

#### Signature

```
OnDocumentClosed(dans i_ipDocument:Document) : Annuler
```

#### *17.7.1.9.3.2 OnModifiedFlagChanged*

Cet événement est déclenché lorsque le statut de modification d'un document change.

#### Signature

```
OnModifiedFlagChanged(dans i_bIsModified:Boolean) : Annuler
```

## 17.7.1.10 Documents

Représente une collection d'objets de `document` objects.

Propriétés pour parcourir le modèle d'objet :

- `Application`
- `Parent`

Ouvrir et créer des mappages :

- `OpenDocument`
- `NewDocument`

Itération à travers la collection :

- `Compter`
- `Item`
- `ActiveDocument`

### Propriétés

| Nom | Description |
|---|---|
| `ActiveDocument` 1209 | Lecture seule. |

| Nom | Description |
|---|---|
| | Extrait le document actif. Si aucun document n'est ouvert, `null` est retourné. |
| `Application` 1210 | Lecture seule.<br>Extrait l'objet de l'application de premier niveau. |
| `Compter` 1210 | Lecture seule.<br>Extrait le nombre de documents dans la collection. |
| `Item` 1210 | Lecture seule.<br>Extrait le document à l'index `n` de la collection. Les index commencent par 1. |
| `Parent` 1211 | Lecture seule.<br>L'objet parent selon le modèle d'objet. |

## Méthodes

| Nom | Description |
|---|---|
| `NewDocument` 1211 | Crée un nouveau document, l'ajoute à la fin de la collection, et rend le document actif. |
| `OpenDocument` 1212 | Ouvre un document de mappage existant (*.mfd). Ajoute un document nouvellement ouvert à la fin de la collection et rend le document actif. |

### 17.7.1.10.1 Propriétés

#### *17.7.1.10.1.1 ActiveDocument*

Extrait le document actif. Si aucun document n'est ouvert, `null` est retourné.

## Signature

```
ActiveDocument : Document
```

## Erreurs

| Code d'erreur : | Description |
|---|---|
| 1600 | L'objet n'est plus valide. |
| 1601 | L'adresse invalide pour le paramètre de retour a été spécifiée. |

### 17.7.1.10.1.2 Application

Extrait l'objet de l'application de premier niveau.

#### Signature

```
Application : Application
```

#### Erreurs

| Code d'erreur : | Description |
|---|---|
| 1600 | L'objet n'est plus valide. |
| 1601 | L'adresse invalide pour le paramètre de retour a été spécifiée. |

### 17.7.1.10.1.3 Count

Extrait le nombre de documents dans la collection.

#### Signature

```
Count : Entier
```

#### Erreurs

| Code d'erreur | Description |
|---|---|
| 1600 | L'objet n'est plus valide. |
| 1601 | L'adresse invalide pour le paramètre de renvoi a été spécifiée. |

### 17.7.1.10.1.4 Item

Extrait le document à l'index n de la collection. Les index commencent par 1.

#### Signature

```
Item(in n:Integer) : Document
```

### Erreurs

| Code d'erreur | Description |
|---|---|
| 1600 | L'objet n'est plus valide. |
| 1601 | L'adresse invalide pour le paramètre de renvoi a été spécifiée. |

#### *17.7.1.10.1.5 Parent*

L'objet parent selon le modèle d'objet.

### Signature

```
Parent : Application
```

### Erreurs

| Code d'erreur | Description |
|---|---|
| 1600 | L'objet n'est plus valide. |
| 1601 | L'adresse invalide pour le paramètre de renvoi a été spécifiée. |

## 17.7.1.10.2 Méthodes

#### *17.7.1.10.2.1 NewDocument*

Crée un nouveau document, l'ajoute à la fin de la collection, et rend le document actif.

### Signature

```
NewDocument() -> Document
```

### Erreurs

| Code d'erreur | Description |
|---|---|
| 1600 | L'objet n'est plus valide. |
| 1601 | L'adresse invalide pour le paramètre de renvoi a été spécifiée. |

#### *17.7.1.10.2.2 OpenDocument*

Ouvre un document de mappage existant (*.mfd). Ajoute un document nouvellement ouvert à la fin de la collection et rend le document actif.

### Signature

```
OpenDocument(dans strPath :String) -> Document
```

### Paramètres

| Nom | Type | Description |
|---|---|---|
| **strPath** | `String` | Le chemin du fichier de mappage. |

### Erreurs

| Code d'erreur | Description |
|---|---|
| 1600 | L'objet n'est plus valide. |
| 1601 | L'adresse invalide pour le paramètre de renvoi a été spécifiée. |

## 17.7.1.11 ErrorMarker

Représente une ligne de message simple. Contrairement aux objets `AppOutputLine`, les marqueurs d'erreur n'ont pas de structure hiérarchique.

Propriétés pour parcourir le modèle d'objet :

- `Application`
- `Parent`

Accès à l'information du message :

- `DocumentFileName`
- `ErrorLevel`
- `Mettre en surbrillance`
- `Sérialisation`
- `Texte`

### Propriétés

| Nom | Description |
|---|---|
| `Application` 1213 | Propriétés<br>Extrait l'objet de premier niveau de l'application. |

| Nom | Description |
|---|---|
| DocumentFileName 1214 | Propriétés<br>Extrait le nom du fichier de mappage avec lequel le marqueur d'erreur est associé. |
| ErrorLevel 1214 | Propriétés<br>Extrait la gravité de l'erreur. |
| Parent 1215 | Propriétés<br>L'objet parent conformément au modèle d'objet. |
| Sérialisation 1215 | Propriétés<br>Sérialiser le marqueur d'erreur dans un string. Utiliser ce string dans des appels vers `Application.HighlightSerializedMarker` ou `Document.HighlightSerializedMarker` pour mettre en surbrillance l'item marqué dans le mappage. Ce string peut être persistant et utilisé dans d'autres instanciations de MapForce ou son contrôle. |
| Texte 1215 | Propriétés<br>Extrait le texte de message. |

## Méthodes

| Nom | Description |
|---|---|
| Mettre en surbrillance 1216 | Met en surbrillance l'item avec lequel le marqueur d'erreur est associé. Si le document correspondant n'est pas ouvert, il sera ouvert. |

### 17.7.1.11.1 Propriétés

#### *17.7.1.11.1.1 Application*

Extrait l'objet de l'application de premier niveau.

## Signature

```
Application : Application
```

## Erreurs

| Code d'erreur : | Description |
|---|---|
| 1900 | L'objet n'est plus valide. |

| Code d'erreur : | Description |
|---|---|
| 1901 | L'adresse invalide pour le paramètre de retour a été spécifiée. |

### *17.7.1.11.1.2 DocumentFileName*

Extrait le nom du fichier de mappage avec lequel le marqueur d'erreur est associé.

#### Signature

```
DocumentFileName : String
```

#### Erreurs

| Code d'erreur | Description |
|---|---|
| 1900 | L'objet n'est plus valide. |
| 1901 | L'adresse invalide pour le paramètre de renvoi a été spécifiée. |

### *17.7.1.11.1.3 ErrorLevel*

Extrait la gravité de l'erreur.

#### Signature

```
ErrorLevel : ENUMCodeGenErrorLevel 1285
```

#### Erreurs

| Code d'erreur | Description |
|---|---|
| 1900 | L'objet n'est plus valide. |
| 1901 | L'adresse invalide a été spécifiée pour le paramètre de renvoi. |

### *17.7.1.11.1.4 Parent*

L'objet parent selon le modèle d'objet.

#### Signature

```
Parent : ErrorMarkers
```

#### Erreurs

| Code d'erreur | Description |
|---|---|
| 1900 | L'objet n'est plus valide. |
| 1901 | L'adresse invalide pour le paramètre de renvoi a été spécifiée. |

### *17.7.1.11.1.5 Sérialisation*

Sérialiser le marqueur d'erreur dans un string. Utilisez ce string dans des appels à `Application.HighlightSerializedMarker` ou `Document.HighlightSerializedMarker` pour surligner l'item marqué dans le mappage. Le string peut être persistant et utilisé dans d'autres instanciations de MapForce ou son contrôle.

#### Signature

```
Sérialisation : String
```

#### Erreurs

| Code d'erreur | Description |
|---|---|
| 1900 | L'objet n'est plus valide. |
| 1901 | L'adresse invalide pour le paramètre de renvoi a été spécifiée. |

### *17.7.1.11.1.6 Texte*

Extrait le texte de message.

#### Signature

```
Texte : String
```

### Erreurs

| Code d'erreur | Description |
|---|---|
| 1900 | L'objet n'est plus valide. |
| 1901 | L'adresse invalide pour le paramètre de renvoi a été spécifiée. |

## 17.7.1.11.2 Méthodes

### *17.7.1.11.2.1 Mise en surbrillance*

Met en surbrillance l'item avec lequel le marqueur d'erreur est associé. Si le document correspondant n'est pas ouvert, il sera ouvert.

### Signature

```
Highlight() -> Void
```

### Erreurs

| Code d'erreur | Description |
|---|---|
| 1900 | L'objet n'est plus valide. |
| 1901 | L'adresse invalide pour le paramètre de renvoi a été spécifiée. |
| 1008 | Le marqueur pointe vers un emplacement qui n'est plus valide. |

# 17.7.1.12 ErrorMarkers

Represents a collection of `ErrorMarker` objects.

Properties to navigate the object model:

- `Application`
- `Parent`

Iterating through the collection:

- `Count`
- `Item`

### Properties

| Name | Description |
|---|---|
| Application 1217 | Read-only.<br>Retrieves the application's top-level object. |
| Count 1217 | Read-only.<br>Retrieves the number of error markers in the collection. |
| Item 1218 | Read-only.<br>Retrieves the error marker at index n from the collection. Indices start with 1. |
| Parent 1218 | Read-only.<br>The parent object according to the object model. |

## 17.7.1.12.1 Propriétés

### *17.7.1.12.1.1 Application*

Extrait l'objet de l'application de premier niveau.

### Signature

```
Application : Application
```

### Erreurs

| Code d'erreur | Description |
|---|---|
| 1800 | L'objet n'est plus valide. |
| 1801 | L'adresse invalide pour le paramètre de renvoi a été spécifiée. |

### *17.7.1.12.1.2 Count*

Extrait le nombre de marqueurs d'erreurs dans la collection.

### Signature

```
Count : Entier
```

### Erreurs

| Code d'erreur | Description |
|---|---|
| 1800 | L'objet n'est plus valide. |
| 1801 | L'adresse invalide pour le paramètre de renvoi a été spécifiée. |

#### *17.7.1.12.1.3 Item*

Extrait le marqueur d'erreur à l'index `n` de la collection. Les indices démarrent avec 1.

### Signature

```
Item(in n:Integer) : ErrorMarker
```

### Erreurs

| Code d'erreur | Description |
|---|---|
| 1800 | L'objet n'est plus valide. |
| 1801 | L'adresse invalide pour le paramètre de renvoi a été spécifiée. |

#### *17.7.1.12.1.4 Parent*

L'objet parent selon le modèle d'objet.

### Signature

```
Parent : Application
```

### Erreurs

| Code d'erreur | Description |
|---|---|
| 1800 | L'objet n'est plus valide. |
| 1801 | L'adresse invalide pour le paramètre de renvoi a été spécifiée. |

## 17.7.1.13 LibraryImport

Une importation `LibraryImport` représente un fichier bibliothèque importé (une entrée depuis la fenêtre **Gérer les bibliothèques**).

## Propriétés

| Nom | Description |
|---|---|
| `Application` 1220 | Lecture seule.<br>Extrait l'objet de premier niveau de l'application. |
| `Parent` 1220 | Lecture seule.<br>Extrait l'objet parent conformément au modèle d'objet. |
| `Chemin` 1221 | Propriétés<br>Obtient le chemin de la bibliothèque importée. |
| `SaveRelativePath` 1221 | Lorsque vous enregistrez le document, cette propriété spécifie si le chemin d'accès à la bibliothèque devrait être enregistré en tant qu'absolu ou relatif. Si **true**, le chemin d'accès à la bibliothèque sera relatif au document. Quand **false**, le chemin d'accès à la bibliothèque sera absolu.<br><br>Ne vous fiez pas à cette propriété pour déterminer si le chemin est absolu ou relatif puisque le chemin peut avoir changé (soit depuis l'interface utilisateur ou via l'API), parce que le document a été chargé depuis le fichier .mfd.<br><br>Si vous définissez la propriété (soit via API, soit via l'interface utilisateur), la fenêtre « Gérer les bibliothèques » affiche immédiatement le statut correct du chemin dans l'interface utilisateur. En interne, toutefois, le `Chemin` de l'objet `ImportedLibrary` ne sera pas modifié jusqu'à l'enregistrement du document.<br><br>Les bibliothèques importées globalement ne peuvent pas être enregistrées avec un chemin relatif. Seules les bibliothèques importées au niveau du document peuvent être enregistrées. |

### 17.7.1.13.1 Propriétés

#### *17.7.1.13.1.1 Application*

Extrait l'objet de l'application de premier niveau.

#### Signature

```
Application : Application
```

#### Erreurs

| Code d'erreur | Description |
|---|---|
| 2500 | L'objet n'est plus valide. |
| 2501 | L'adresse invalide pour le paramètre de renvoi a été spécifiée. |

#### *17.7.1.13.1.2 Parent*

Extrait l'objet parent conformément au modèle d'objet.

#### Signature

```
Parent : LibraryImports
```

#### Erreurs

| Code d'erreur | Description |
|---|---|
| 2500 | L'objet n'est plus valide. |
| 2501 | L'adresse invalide pour le paramètre de renvoi a été spécifiée. |

### *17.7.1.13.1.3 Chemin*

Obtient le chemin de la bibliothèque importée.

#### Signature

```
Chemin : String
```

#### Erreurs

| Code d'erreur | Description |
|---|---|
| 2500 | L'objet n'est plus valide. |
| 2501 | L'adresse invalide pour le paramètre de renvoi a été spécifiée. |

### *17.7.1.13.1.4 SaveRelativePath*

Lorsque vous enregistrez le document, cette propriété spécifie si le chemin d'accès à la bibliothèque devrait être enregistré en tant qu'absolu ou relatif. Si **true**, le chemin d'accès à la bibliothèque sera relatif au document. Quand **false**, le chemin d'accès à la bibliothèque sera absolu.

Ne vous fiez pas à cette propriété pour déterminer si le chemin est absolu ou relatif puisque le chemin peut avoir changé (soit depuis l'interface utilisateur ou via l'API), parce que le document a été chargé depuis le fichier .mfd.

Si vous définissez la propriété (soit via API, soit via l'interface utilisateur), la fenêtre « Gérer les bibliothèques » affiche immédiatement le statut correct du chemin dans l'interface utilisateur. En interne, toutefois, le `Chemin` de l'objet `ImportedLibrary` ne sera pas modifié jusqu'à l'enregistrement du document.

> Les bibliothèques importées globalement ne peuvent pas être enregistrées avec un chemin relatif. Seules les bibliothèques importées au niveau du document peuvent être enregistrées.

#### Signature

```
SaveRelativePath : Booléen
```

#### Erreurs

| Code d'erreur | Description |
|---|---|
| 2500 | L'objet n'est plus valide. |
| 2501 | L'adresse invalide pour le paramètre de renvoi a été spécifiée. |
| 2502 | Les importations globales ne peuvent pas être enregistrées avec un chemin relatif. |

## 17.7.1.14 LibraryImports

Représente une collection de bibliothèques importées (objets `LibraryImport`). Utilisez les propriétés `Application` et `Parent` pour parcourir le modèle d'objet. Utilisez les propriétés `Count` et `Item` pour itérer à travers la collection. Vous pouvez obtenir cette collection comme suit :

- Localement (au niveau du document), par le biais de la propriété `Document.LibraryImports`
- Globalement (au niveau de l'application), par le biais de la propriété `Application.LibraryImports`.

> Si vous obtenez la collection `LibraryImports` depuis l'objet de l'application, la propriété `Parent` de la collection sera nulle.

### Propriétés

| Nom | Description |
|---|---|
| `Application` 1223 | Lecture seule.<br>Obtient l'objet de l'application de premier niveau. |
| `Compter` 1223 | Lecture seule.<br>Obtient le compte d'objets `LibraryImport` dans cette collection. |
| `Item` 1224 | Lecture seule.<br>Extrait une entrée de bibliothèque à l'index *n* de cette collection. L'index est basé sur 1. |
| `Parent` 1224 | Lecture seule.<br>Obtient le document parent pour les importations de la bibliothèque locale. Si vous obtenez la collection `LibraryImports` depuis l'objet de l'application, la propriété `Parent` de la collection sera nulle. |

### Méthodes

| Nom | Description |
|---|---|
| `Ajouter` | Ajoute une nouvelle bibliothèque à cet objet `LibraryImports`. La nouvelle bibliothèque aura le chemin fourni par le paramètre **i_strFileName**. |
| `Chercher` | Renvoie une référence bibliothèque en fonction du chemin vers le fichier bibliothèque. |
| `Supprimer` 1282 | Supprime une référence bibliothèque depuis la fenêtre **Gérer les Bibliothèques**. |

### 17.7.1.14.1 Propriétés

#### *17.7.1.14.1.1 Application*

Obtient l'objet de l'application de premier niveau.

#### Signature

```
Application : Application
```

#### Erreurs

| Code d'erreur | Description |
|---|---|
| 2400 | L'objet n'est plus valide. |
| 2401 | L'adresse invalide pour le paramètre de renvoi a été spécifiée. |

#### *17.7.1.14.1.2 Count*

Obtient le compte d'objets `LibraryImport` dans cette collection.

#### Signature

```
Count : Entier
```

#### Erreurs

| Code d'erreur | Description |
|---|---|
| 2400 | L'objet n'est plus valide. |
| 2401 | L'adresse invalide pour le paramètre de renvoi a été spécifiée. |

#### *17.7.1.14.1.3 Item*

Extrait une entrée de bibliothèque à l'index *n* de cette collection. L'index est basé sur 1.

##### Signature

```
Item(in n:Integer) : LibraryImport
```

##### Erreurs

| Code d'erreur | Description |
|---|---|
| 2400 | L'objet n'est plus valide. |
| 2401 | L'adresse invalide pour le paramètre de renvoi a été spécifiée. |

#### *17.7.1.14.1.4 Parent*

Obtient le document parent pour les importations de la bibliothèque locale. Si vous obtenez la collection `LibraryImports` depuis l'objet de l'application, la propriété `Parent` de la collection sera nulle.

##### Signature

```
Parent : Document
```

##### Erreurs

| Code d'erreur | Description |
|---|---|
| 2400 | L'objet n'est plus valide. |
| 2401 | L'adresse invalide pour le paramètre de renvoi a été spécifiée. |

### 17.7.1.14.2 Méthodes

#### *17.7.1.14.2.1 Add (2)*

Adds a new library to this `LibraryImports` object. The new library will have the path supplied by the **i_strFileName** parameter.

##### Signature

```
Add(in i_strFileName:String) -> LibraryImport
```

### Parameters

| Name | Type | Description |
|---|---|---|
| **i_strFileName** | `String` | Specifies the path of the library file. This path can be either absolute or relative to the mapping, depending on the state in which it was passed to the object. When the document is saved, the path will be made relative if the `LibraryImport.SaveRelativePath` flag is **true**; otherwise, it will be made absolute. |

### Errors

| Error code | Description |
|---|---|
| 2400 | The object is no longer valid. |
| 2401 | Invalid address for the return parameter was specified. |
| 2402 | Adding library file failed. |

## *17.7.1.14.2.2 Find (2)*

Returns a library reference given the path to the library file.

### Signature

```
Find(in i_strFileName:String) -> LibraryImport
```

### Parameters

| Name | Type | Description |
|---|---|---|
| **i_strFileName** | `String` | The path of the library file to search for. For locally imported libraries, you can specify either the absolute or the relative path to the library file (unlike the `Remove` method, which requires the exact path).<br><br>For globally imported libraries, the path must always be absolute |

| Name | Type | Description |
|---|---|---|
| | | (since globally imported libraries cannot have a relative path). |

## Errors

| Error code | Description |
|---|---|
| 2400 | The object is no longer valid. |
| 2401 | Invalid address for the return parameter was specified. |

### *17.7.1.14.2.3 Remove (2)*

Removes a library reference from the **Manage Libraries** window.

## Signature

```
Remove(in i_strFileName:String) -> Void
```

## Parameters

| Name | Type | Description |
|---|---|---|
| **i_strFileName** | `String` | The path of the library file to remove. Note that the path must reflect exactly the current (most recent) state of the `LibraryImport` object. Remember that the path may be either relative or absolute, and it may have changed if you saved the document, depending on the `LibraryImport.SaveRelativePath` flag. Therefore, if the `LibraryImport` object currently contains a relative path, then you should supply a relative path as value of this parameter. Otherwise, the library is not found and the `Remove` method fails.<br><br>The above applies only for locally imported libraries. For globally imported libraries, the path must always be absolute (since |

| Name | Type | Description |
|---|---|---|
| | | globally imported libraries cannot have a relative path) |

### Errors

| Error code | Description |
|---|---|
| 2400 | The object is no longer valid. |

## 17.7.1.15 MapForceView

Représente l'affichage actuel dans l'onglet de mappage de MapForce pour un document. Un document a exactement un affichage `MapForceView`, qui affiche le mappage actuellement actif.

Propriétés pour parcourir le modèle d'objet :

- `Application`
- `Parent`

Activation et propriétés d'affichage :

- `Activé`
- `ShowItemTypes`
- `ShowLibraryInFunctionHeader`
- `HighlightMyConnections`
- `HighlightMyConnectionsRecursivly`

Propriétés relatives au mappage :

- `ActiveMapping`
- `ActiveMappingName`

Ajouter des éléments :

- `InsertWSDLCall`
- `InsertXMLFile`
- `InsertXMLSchema`
- `InsertXMLSchemaWithSample`

### Propriétés

| Nom | Description |
|---|---|
| `Activé` 1229 | Utilisez cette propriété pour demander si la vue de mappage est la vue active, ou définissez cette vue en tant que vue active. |

| Nom | Description |
|---|---|
| ActiveMapping 1229 | Obitent ou définit le mappage actif actuel dans le document auquel ce MapForceView appartient. |
| ActiveMappingName 1230 | Obtient ou définit le mappage actif actuel par le nom dans le document auquel ce MapForceView appartient. |
| Application 1230 | Lecture seule.<br>Extrait l'objet de l'application de premier niveau. |
| HighlightMyConnections 1230 | Cette propriété définit si uniquement les connexions depuis les éléments sélectionnés doivent être mises en surbrillance. |
| HighlightMyConnectionsRecursively 1231 | Cette propriété définit si uniquement les connexions venant directement ou indirectement des éléments sélectionnés doivent être mis en surbrillance. |
| Parent 1243 | Lecture seule.<br>L'objet parent selon le modèle d'objet. |
| ShowItemTypes 1231 | Cette propriété définit si les types d'items devraient être affichés dans le diagramme de mappage. |
| ShowLibraryInFunctionHeader 1232 | Cette propriété définit si le nom de la bibliothèque de fonction devrait faire partie de noms de fonction. |

## Méthodes

| Nom | Description |
|---|---|
| InsertWSDLCall 1232 | Ajoute un nouveau composant d'appel WSDL au mappage. |
| InsertXMLFile 1233 | MapForceView.InsertXMLFile est obsolète. Utilisez Mapping.InsertXMLFile à la place. |
| InsertXMLSchema 1233 | MapForceView.InsertXMLSchema est obsolète. Utilisez Mapping.InsertXMLSchema à la place. |
| InsertXMLSchemaWithSample 1233 | MapForceView.InsertXMLSchemaWithSample est obsolète. Utilisez Mapping.InsertXMLFile à la place. Notez que Mapping.InsertXMLFile ne requiert pas de paramètre pour passer l'élément racine. L'élément racine est automatiquement définit en tant que nom d'élément de la racine du fichier XML. |

### 17.7.1.15.1 Propriétés

#### *17.7.1.15.1.1 Activé*

Utilisez cette propriété pour demander si la vue de mappage est la vue active, ou définissez cette vue en tant que vue active.

#### Signature

```
Active : Booléen
```

#### Erreurs

| Code d'erreur : | Description |
|---|---|
| 1300 | L'objet n'est plus valide. |
| 1301 | L'adresse invalide pour le paramètre de retour a été spécifiée. |

#### *17.7.1.15.1.2 ActiveMapping*

Obtient ou définit le mappage actif actuel dans le document auquel ce `MapForceView` appartient.

#### Signature

```
ActiveMapping : Mappage
```

#### Erreurs

| Code d'erreur : | Description |
|---|---|
| 1300 | L'objet n'est plus valide. |
| 1301 | L'adresse invalide pour le paramètre de retour a été spécifiée. |

### *17.7.1.15.1.3 ActiveMappingName*

Obtient ou définit le mappage actif actuel par le nom dans le document auquel ce MapForceView appartient.

#### Signature

```
ActiveMappingName : String
```

#### Erreurs

| Code d'erreur : | Description |
|---|---|
| 1300 | L'objet n'est plus valide. |
| 1301 | L'adresse invalide pour le paramètre de retour a été spécifiée. |

### *17.7.1.15.1.4 Application*

Extrait l'objet de l'application de premier niveau.

#### Signature

```
Application : Application
```

#### Erreurs

| Code d'erreur | Description |
|---|---|
| 1300 | L'objet n'est plus valide. |
| 1301 | L'adresse invalide pour le paramètre de renvoi a été spécifiée. |

### *17.7.1.15.1.5 HighlightMyConnections*

Cette propriété définit si uniquement les connexions depuis les éléments sélectionnés doivent être mises en surbrillance.

#### Signature

```
HighlightMyConnections : Booléenne
```

### Erreurs

| Code d'erreur | Description |
|---|---|
| 1300 | L'objet n'est plus valide. |
| 1301 | L'adresse invalide pour le paramètre de renvoi a été spécifiée. |

### *17.7.1.15.1.6 HighlightMyConnectionsRecursively*

Cette propriété définit si uniquement les connexions venant directement ou indirectement des éléments sélectionnés doivent être mis en surbrillance.

### Signature

```
HighlightMyConnectionsRecursively : Booléenne
```

### Erreurs

| Code d'erreur | Description |
|---|---|
| 1300 | L'objet n'est plus valide. |
| 1301 | L'adresse invalide pour le paramètre de renvoi a été spécifiée. |

### *17.7.1.15.1.7 ShowItemTypes*

Cette propriété définit si les types d'items devraient être affichés dans le diagramme de mappage.

### Signature

```
ShowItemTypes : Booléen
```

### Erreurs

| Code d'erreur | Description |
|---|---|
| 1300 | L'objet n'est plus valide. |
| 1301 | L'adresse invalide pour le paramètre de renvoi a été spécifiée. |

#### *17.7.1.15.1.8 ShowLibraryInFunctionHeader*

Cette propriété définit si le nom de la bibliothèque de fonction devrait faire partie de noms de fonction.

### Signature

```
ShowLibraryInFunctionHeader : Booléen
```

### Erreurs

| Code d'erreur | Description |
|---|---|
| 1300 | L'objet n'est plus valide. |
| 1301 | L'adresse invalide pour le paramètre de renvoi a été spécifiée. |

## 17.7.1.15.2 Méthodes

#### *17.7.1.15.2.1 InsertWSDLCall*

Ajoute un nouveau composant d'appel WSDL au mappage.

### Signature

```
InsertWSDLCall(dans i_strWSDLFileName :String) -> Void
```

### Paramètres

| Nom | Type | Description |
|---|---|---|
| **i_strWSDLFileName** | String | Spécifie le chemin du fichier WSDL pour ajouter au mappage. |

### Erreurs

| Code d'erreur | Description |
|---|---|
| 1300 | L'objet n'est plus valide. |
| 1301 | L'adresse invalide pour le paramètre de renvoi a été spécifiée. |

### *17.7.1.15.2.2 InsertXMLFile (obsolete)*

`MapForceView.InsertXMLFile` est obsolète. Utilisez `Mapping.InsertXMLFile` à la place.

#### Signature

```
InsertXMLFile(in i_strFileName:String, dans i_strXMLRootName:String) -> Void
```

#### Paramètres

| Nom | Type | Description |
|---|---|---|
| **i_strFileName** | **String** | |
| **i_strXMLRootName** | **String** | |

### *17.7.1.15.2.3 InsertXMLSchema (obsolete)*

`MapForceView.InsertXMLSchema` est obsolète. Utilisez `Mapping.InsertXMLSchema` à la place.

#### Signature

```
InsertXMLSchema(in i_strSchemaFileName:String, in i_strXMLRootName:String) -> Void
```

#### Paramètres

| Nom | Type | Description |
|---|---|---|
| **i_strSchemaFileName** | **String** | |
| **i_strXMLRootName** | **String** | |

### *17.7.1.15.2.4 InsertXMLSchemaWithSample (obsolete)*

`MapForceView.InsertXMLSchemaWithSample` est obsolète. Utilisez `Mapping.InsertXMLFile` à la place. Notez que `Mapping.InsertXMLFile` ne requiert pas de paramètre pour passer l'élément racine. L'élément racine est automatiquement définit en tant que nom d'élément de la racine du fichier XML.

#### Signature

```
InsertXMLSchemaWithSample(in i_strSchemaFileName:String, in i_strXMLExampleFile:String,
in i_strXMLRootName:String) -> Void
```

### Paramètres

| Nom | Type | Description |
|---|---|---|
| **i_strSchemaFileName** | `String` | |
| **i_strXMLExampleFile** | `String` | |
| **i_strXMLRootName** | `String` | |

## 17.7.1.16 Mappage

Un objet de `mappage` représente un mappage dans un document, soit le mappage principal, soit le mappage de fonction définie par l'utilisateur.

Propriétés pour parcourir le modèle d'objet :

- `Application`
- `Parent`

Propriétés au mappage :

- `IsMainMapping`
- `Nom`

Composants dans le mappage :

- `Composants`

Ajouter des éléments :

- `CreateConnection`
- `InsertFunctionCall`
- `InsertXMLFile`
- `InsertXMLSchema`
- `InsertXMLSchemaInputParameter`
- `InsertXMLSchemaOutputParameter`

### Propriétés

| Nom | Description |
|---|---|
| `Application` 1245 | Lecture seule.<br>Extrait l'objet de l'application de premier niveau. |
| `Composants` 1236 | Lecture seule.<br>Renvoie une collection de tous les composants dans le mappage actuel. |
| `IsMainMapping` 1237 | Lecture seule. |

| Nom | Description |
|---|---|
| | Indique si le mappage actuel est le mappage principal du document dans lequel le mappage se trouve.<br><br>**True** signifie qu'il s'agit du mappage principal.<br>**False** signifie qu'il s'agit d'une fonction définie par l'utilisateur (UDF). |
| Nom 1237 | Lecture seule.<br>Le nom du mappage ou de la fonction définie par l'utilisateur (UDF). |
| Parent 1251 | Lecture seule.<br>L'objet parent selon le modèle d'objet. |

## Méthodes

| Nom | Description |
|---|---|
| CreateConnection 1237 | Crée une connexion entre les deux datapoints fournis (`DatapointFrom` & `DatapointTo`).<br><br>La connexion ne sera pas établie si `DatapointFrom` n'est pas un datapoint output-side, si `DatapointTo` n'est pas un datapoint input-side, ou si une connexion entre ces deux datapoints existe déjà. |
| InsertFunctionCall 1238 | Insère un composant d'appel de fonction dans le mappage actuel.<br><br>Les noms de bibliothèque et de fonction spécifiés indiquent la fonction ou la fonction définie par l'utilisateur à appeler. |
| InsertXMLFile 1239 | Ajoute un nouveau compostant de schéma XML au mappage.<br><br>La structure interne du composant est déterminée par le schéma référencé dans le fichier XML spécifié (`i_strFileName`) ou, si le fichier XML ne référence pas de fichier de schéma, par le fichier de schéma spécifié séparément (`i_strSchemaFileName`).<br><br>Si le fichier XML a une référence de fichier de schéma, alors le paramètre `i_strSchemaFileName` est ignoré.<br><br>L'élément racine du fichier XML sera utilisé dans le composant.<br><br>Le fichier XML spécifié est utilisé comme échantillon d'entrée pour évaluer le mappage. |
| InsertXMLSchema 1240 | Ajoute un nouveau compostant de schéma XML au mappage. |

| Nom | Description |
|---|---|
| | La structure interne du composant est déterminée par le fichier de schéma spécifié dans le premier paramètre.<br><br>Le deuxième paramètre définit l'élément racine de ce schéma s'il existe plus d'un candidat.<br><br>Si l'élément racine passé est un string vide et plus de candidats sont disponibles, une boîte de dialogue **Sélectionner l'élément racine** apparaîtra si MapForce est visible. Si MapForce est invisible, aucune boîte de dialogue n'apparaîtra et seule une erreur est retournée.<br><br>Aucun échantillon d'entrée XML n'est assigné à ce composant. |
| `InsertXMLSchemaInputParameter` 1241 | Insère un composant de paramètre d'entrée du schéma XML dans le mappage actuel.<br><br>Le mappage actuel doit être une fonction définie par l'utilisateur. Tenter de l'insérer (le paramètre d'entrée de schéma) dans le mappage principal échouera. |
| `InsertXMLSchemaOutputParameter` 1242 | Insère un composant de paramètre de sortie de schéma XML dans le mappage actuel.<br><br>Le mappage actuel doit être une fonction définie par l'utilisateur. Tenter de l'insérer (le paramètre de sortie du schéma) dans le mappage principal échouera. |

## 17.7.1.16.1 Propriétés

### *17.7.1.16.1.1 Composants*

Renvoie une collection de tous les composants dans le mappage actuel.

#### Signature

```
Components : Composants
```

#### Erreurs

| Code d'erreur | Description |
|---|---|
| 1200 | L'objet n'est plus valide. |
| 1201 | L'adresse invalide pour le paramètre de renvoi a été spécifiée. |

#### *17.7.1.16.1.2 IsMainMapping*

Indique si le mappage actuel est le mappage principal du document dans lequel le mappage se trouve.

**True** signifie qu'il s'agit du mappage principal.
**False** signifie qu'il s'agit d'une fonction définie par l'utilisateur (UDF).

##### Signature

```
IsMainMapping : Booléenne
```

##### Erreurs

| Code d'erreur | Description |
|---|---|
| 1200 | L'objet n'est plus valide. |
| 1201 | L'adresse invalide pour le paramètre de renvoi a été spécifiée. |

#### *17.7.1.16.1.3 Nom*

Le nom du mappage ou de la fonction définie par l'utilisateur (UDF).

##### Signature

```
Nom : String
```

##### Erreurs

| Code d'erreur | Description |
|---|---|
| 1200 | L'objet n'est plus valide. |
| 1201 | L'adresse invalide pour le paramètre de renvoi a été spécifiée. |

### 17.7.1.16.2 Méthodes

#### *17.7.1.16.2.1 CreateConnection*

Crée une connexion entre les deux datapoints fournis (`DatapointFrom` & `DatapointTo`).

La connexion ne sera pas établie si `DatapointFrom` n'est pas un datapoint output-side, si `DatapointTo` n'est pas un datapoint input-side, ou si une connexion entre ces deux datapoints existe déjà.

### Signature

```
CreateConnection(dans DatapointFrom:Datapoint, dans DatapointTo:Datapoint) -> Connection
```

### Paramètres

| Nom | Type | Description |
|---|---|---|
| **DatapointFrom** | `Datapoint` | Le datapoint duquel la connexion doit être créée. |
| **DatapointTo** | `Datapoint` | Le datapoint de destination. |

### Erreurs

| Code d'erreur | Description |
|---|---|
| 1200 | L'objet n'est plus valide. |
| 1201 | L'adresse invalide pour le paramètre de renvoi a été spécifiée. |
| 1240 | Il n'est pas autorisé de modifier le document. Il est en lecture seule. |
| 1241 | Échec de la création de la connexion. |

## *17.7.1.16.2.2 InsertFunctionCall*

Insère un composant d'appel de fonction dans le mappage actuel.

Les noms de bibliothèque et de fonction spécifiés indiquent la fonction ou la fonction définie par l'utilisateur à appeler.

### Signature

```
InsertFunctionCall(dans strFunctionName:String, dans strLibraryName:String) -> Component
```

### Paramètres

| Nom | Type | Description |
|---|---|---|
| **strFunctionName** | `String` | Le nom de la fonction à être insérée. |
| **strLibraryName** | `String` | Le nom de la bibliothèque de la fonction à insérer |

### Erreurs

| Code d'erreur | Description |
|---|---|
| 1200 | L'objet n'est plus valide. |
| 1201 | L'adresse invalide pour le paramètre de renvoi a été spécifiée. |
| 1240 | Il n'est pas autorisé de modifier le document. Il est en lecture seule. |
| 1242 | Échec lors de la création du composant d'appel de la fonction. |

## *17.7.1.16.2.3 InsertXMLFile*

Ajoute un nouveau compostant de schéma XML au mappage.

La structure interne du composant est déterminée par le schéma référencé dans le fichier XML spécifié (`i_strFileName`) ou, si le fichier XML ne référence pas de fichier de schéma, par le fichier de schéma spécifié séparément (`i_strSchemaFileName`).

Si le fichier XML a une référence de fichier de schéma, alors le paramètre `i_strSchemaFileName` est ignoré.

L'élément racine du fichier XML sera utilisé dans le composant.

Le fichier XML spécifié est utilisé comme échantillon d'entrée pour évaluer le mappage.

### Signature

```
InsertXMLFile(in i_strFileName:String, dans i_strSchemaFileName:String) -> Component
```

### Paramètres

| Nom | Type | Description |
|---|---|---|
| **i_strFileName** | `String` | Le chemin de l'instance du fichier XML à ajouter. |
| **i_strSchemaFileName** | `String` | Le chemin du fichier de la Définition de schéma XML à ajouter. |

### Erreurs

| Code d'erreur | Description |
|---|---|
| 1200 | L'objet n'est plus valide. |
| 1201 | L'adresse invalide pour le paramètre de renvoi a été spécifiée. |

| Code d'erreur | Description |
|---|---|
| 1240 | Il n'est pas autorisé de modifier le document. Il est en lecture seule. |
| 1244 | Échec de la création du composant. |

### *17.7.1.16.2.4 InsertXMLSchema*

Ajoute un nouveau compostant de schéma XML au mappage.

La structure interne du composant est déterminée par le fichier de schéma spécifié dans le premier paramètre.

Le deuxième paramètre définit l'élément racine de ce schéma s'il existe plus d'un candidat.

Si l'élément racine passé est un string vide et plus de candidats sont disponibles, une boîte de dialogue **Sélectionner l'élément racine** apparaîtra si MapForce est visible. Si MapForce est invisible, aucune boîte de dialogue n'apparaîtra et seule une erreur est retournée.

Aucun échantillon d'entrée XML n'est assigné à ce composant.

#### Signature

```
InsertXMLSchema(dans i_strSchemaFileName:String, dans i_strXMLRootName:String) ->
Component
```

#### Paramètres

| Nom | Type | Description |
|---|---|---|
| **i_strSchemaFileName** | `String` | Le chemin du fichier de la Définition de schéma XML à ajouter. |
| **i_strXMLRootName** | `String` | L'élément racine du schéma (si applicable quand le schéma a plus d'un élément racine). |

#### Erreurs

| Code d'erreur | Description |
|---|---|
| 1200 | L'objet n'est plus valide. |
| 1201 | L'adresse invalide pour le paramètre de renvoi a été spécifiée. |
| 1240 | Il n'est pas autorisé de modifier le document. Il est en lecture seule. |
| 1244 | Échec de la création du composant. |

### *17.7.1.16.2.5 InsertXMLSchemaInputParameter*

Insère un composant de paramètre d'entrée du schéma XML dans le mappage actuel.

Le mappage actuel doit être une fonction définie par l'utilisateur. Tenter de l'insérer (le paramètre d'entrée de schéma) dans le mappage principal échouera.

## Signature

```
InsertXMLSchemaInputParameter(dans strParamName:String, in strSchemaFileName:String, dans
strXMLRootElementName:String) -> Component
```

## Paramètres

| Nom | Type | Description |
|---|---|---|
| **strParamName** | `String` | Le nom du composant de paramètre d'entrée à créer. |
| **strSchemaFileName** | `String` | Le chemin du fichier de la Définition de schéma XML à ajouter. |
| **strXMLRootElementName** | `String` | L'élément racine du schéma (si applicable quand le schéma a plus d'un élément racine). Si l'élément racine passé est un string vide et plus de candidats sont disponibles, un dialogue **Sélectionner l'élément racine** apparaîtra si MapForce est visible. Si MapForce est invisible, aucune boîte de dialogue n'apparaîtra et seule une erreur est retournée. |

## Erreurs

| Code d'erreur | Description |
|---|---|
| 1200 | L'objet n'est plus valide. |
| 1201 | L'adresse invalide pour le paramètre de renvoi a été spécifiée. |
| 1240 | Il n'est pas autorisé de modifier le document. Il est en lecture seule. |
| 1243 | Échec de la création du composant de paramètre. |
| 1245 | Cette opération n'est pas prise en charge pour le mappage principal. |

### *17.7.1.16.2.6 InsertXMLSchemaOutputParameter*

Insère un composant de paramètre de sortie de schéma XML dans le mappage actuel.

Le mappage actuel doit être une fonction définie par l'utilisateur. Tenter de l'insérer (le paramètre de sortie du schéma) dans le mappage principal échouera.

## Signature

```
InsertXMLSchemaOutputParameter(dans strParamName:String, in strSchemaFileName:String, in
strXMLRootElementName:String) -> Component
```

## Paramètres

| Nom | Type | Description |
|---|---|---|
| **strParamName** | `String` | Le nom du composant de paramètre de sortie à créer. |
| **strSchemaFileName** | `String` | Le chemin du fichier de la Définition de schéma XML à ajouter. |
| **strXMLRootElementName** | `String` | L'élément racine du schéma (si applicable quand le schéma a plus d'un élément racine). Si l'élément racine passé est un string vide et plus de candidats sont disponibles, un dialogue **Sélectionner l'élément racine** apparaîtra si MapForce est visible. Si MapForce est invisible, aucune boîte de dialogue n'apparaîtra et seule une erreur est retournée. |

## Erreurs

| Code d'erreur | Description |
|---|---|
| 1200 | L'objet n'est plus valide. |
| 1201 | L'adresse invalide pour le paramètre de renvoi a été spécifiée. |
| 1240 | Il n'est pas autorisé de modifier le document. Il est en lecture seule. |
| 1243 | Échec de la création du composant de paramètre. |
| 1245 | Cette opération n'est pas prise en charge pour le mappage principal. |

## 17.7.1.17 Mappages

Représente une collection d'objets de `mappage`.

Propriétés pour parcourir le modèle d'objet :

- `Application`
- `Parent`

Itération à travers la collection :

- `Compter`
- `Item`

### Propriétés

| Nom | Description |
|---|---|
| `Application` 1257 | Lecture seule.<br>Extrait l'objet de l'application de premier niveau. |
| `Compter` 1257 | Lecture seule.<br>Extrait le nombre de mappages dans la collection. |
| `Item` 1258 | Lecture seule.<br>Extrait le mappage à l'index `n` de la collection. Les index commencent par 1. |
| `Parent` 1261 | Lecture seule.<br>L'objet parent selon le modèle d'objet. |

### 17.7.1.17.1 Propriétés

#### *17.7.1.17.1.1 Parent*

L'objet parent selon le modèle d'objet.

### Signature

```
Parent : Document
```

### Erreurs

| Code d'erreur | Description |
|---|---|
| 1300 | L'objet n'est plus valide. |

| Code d'erreur | Description |
|---|---|
| 1301 | L'adresse invalide pour le paramètre de renvoi a été spécifiée. |

## 17.7.1.18 Options

Cet objet donne accès à toutes les options disponibles dans le dialogue **Outils | Options**.

Propriétés pour parcourir le modèle d'objet :

- `Application`
- `Parent`

Options générales :

- `ShowLogoOnPrint`
- `ShowLogoOnStartup`
- `UseGradientBackground`

Options pour la génération de code :

- `DefaultOutputEncoding`
- `DefaultOutputByteOrder`
- `DefaultOutputByteOrderMark`
- `XSLTDefaultOutputDirectory`
- `CodeDefaultOutputDirectory`
- `CPPSettings_DOMType`
- `CPPSettings_GenerateVC6ProjectFile`
- `CppSettings_GenerateVSProjectFile`
- `CPPSettings_LibraryType`
- `CPPSettings_UseMFC`
- `CSharpSettings_ProjectType`

### Propriétés

| Nom | Description |
|---|---|
| `Application` 1272 | Lecture seule.<br>Extrait l'objet de l'application de premier niveau. |
| `CodeDefaultOutputDirectory` 1246 | Spécifie le répertoire cible où les fichiers générés par `Document.GenerateCppCode`, `Document.GenerateJavaCode` et `Document.GenerateCHashCode` sont placés. |
| `CPPSettings_DOMType` 1246 | Spécifie le type de DOM utilisé par `Document.GenerateCppCode`. |
| `CPPSettings_GenerateVC6ProjectFile` 1247 | Spécifie si les fichiers de projet VisualC++ 6.0 devraient être générés par `Document.GenerateCppCode`. |
| `CppSettings_GenerateVSProjectFile` 1247 | Spécifie la version de Visual Studio dans laquelle les fichiers de projet devraient être générés par `Document.GenerateCppCode`. |

| Nom | Description |
|---|---|
| CPPSettings_LibraryType 1248 | Spécifie le type de bibliothèque utilisé par Document.GenerateCppCode. |
| CPPSettings_UseMFC 1248 | Spécifie si la prise en charge MFC devrait être utilisée par C++ code généré par Document.GenerateCppCode. |
| CSharpSettings_ProjectType 1248 | Spécifie le type de projet C# utilisé par Document.GenerateCHashCode. |
| DefaultOutputByteOrder 1249 | Ordre d'octet pour l'encodage de fichier utilisé pour les fichiers de sortie. |
| DefaultOutputByteOrderMark 1249 | Indique si une marque d'ordre d'octet (BOM) doit être incluse dans l'encodage de fichier des fichiers de sortie. |
| DefaultOutputEncoding 1250 | Encodage de fichier utilisé pour les fichiers de sortie. |
| GenerateWrapperClasses 1250 | Indique si les classes de wrapper doivent également être générées lorsqu'elles génèrent du code. |
| JavaSettings_ApacheAxisVersion 1250 | Cette propriété est obsolète. |
| Parent 1275 | Lecture seule.<br>L'objet parent selon le modèle d'objet. |
| ShowLogoOnPrint 1251 | Afficher ou masquer le logo de MapForce sur des sorties imprimées. |
| ShowLogoOnStartup 1251 | Afficher ou masquer le logo de MapForce sur démarrage de l'application. |
| UseGradientBackground 1252 | Définit ou extrait le mode de couleur d'arrière-plan pour une fenêtre de mappage. |
| XSLTDefaultOutputDirectory 1252 | Spécifie le répertoire cible où les fichiers générés par Document.GenerateXSLT sont placés. |

## 17.7.1.18.1 Propriétés

### *17.7.1.18.1.1 Application*

Extrait l'objet de l'application de premier niveau.

#### Signature

```
Application : Application
```

### Erreurs

| Code d'erreur : | Description |
|---|---|
| 1200 | L'objet n'est plus valide. |
| 1201 | L'adresse invalide pour le paramètre de retour a été spécifiée. |

## *17.7.1.18.1.2 CodeDefaultOutputDirectory*

Spécifie le répertoire cible où les fichiers générés par `Document.GenerateCppCode`, `Document.GenerateJavaCode` et `Document.GenerateCHashCode` sont placés.

### Signature

```
CodeDefaultOutputDirectory : String
```

### Erreurs

| Code d'erreur | Description |
|---|---|
| 1400 | L'objet n'est plus valide. |
| 1401 | L'adresse invalide pour le paramètre de renvoi a été spécifiée. |

## *17.7.1.18.1.3 CPPSettings_DOMType*

Spécifie le type de DOM utilisé par `Document.GenerateCppCode`.

### Signature

```
CPPSettings_DOMType : ENUMDOMType 1287
```

### Erreurs

| Code d'erreur | Description |
|---|---|
| 1400 | L'objet n'est plus valide. |
| 1401 | L'adresse invalide pour le paramètre de renvoi a été spécifiée. |
| 1402 | La valeur de paramètre est en dehors de la plage. |
| 1403 | La valeur de paramètre n'est plus disponible. |

### *17.7.1.18.1.4 CPPSettings_GenerateVC6ProjectFile (obsolete)*

Spécifie si les fichiers de projet VisualC++ 6.0 devraient être générés par `Document.GenerateCppCode`.

#### Signature

```
CPPSettings_GenerateVC6ProjectFile : Booléen
```

#### Erreurs

| Code d'erreur | Description |
|---|---|
| 1400 | L'objet n'est plus valide. |
| 1401 | L'adresse invalide pour le paramètre de renvoi a été spécifiée. |
| 1402 | La valeur de paramètre est en dehors de la plage. |
| 1403 | La valeur de paramètre n'est plus disponible. |

### *17.7.1.18.1.5 CppSettings_GenerateVSProjectFile*

Spécifie la version de Visual Studio dans laquelle les fichiers de projet devraient être générés par `Document.GenerateCppCode`.

#### Signature

```
CppSettings_GenerateVSProjectFile : ENUMProjectType 1289
```

#### Erreurs

| Code d'erreur | Description |
|---|---|
| 1400 | L'objet n'est plus valide. |
| 1401 | L'adresse invalide pour le paramètre de renvoi a été spécifiée. |
| 1402 | La valeur de paramètre est en dehors de la plage. |
| 1403 | La valeur de paramètre n'est plus disponible. |

### 17.7.1.18.1.6 CPPSettings_LibraryType

Spécifie le type de bibliothèque utilisé par `Document.GenerateCppCode`.

#### Signature

```
CPPSettings_LibraryType : ENUMLibType
```

#### Erreurs

| Code d'erreur | Description |
|---|---|
| 1400 | L'objet n'est plus valide. |
| 1401 | L'adresse invalide pour le paramètre de renvoi a été spécifiée. |

### 17.7.1.18.1.7 CPPSettings_UseMFC

Spécifie si la prise en charge MFC devrait être utilisée par C++ code généré par `Document.GenerateCppCode`.

#### Signature

```
CPPSettings_UseMFC : Booléen
```

#### Erreurs

| Code d'erreur | Description |
|---|---|
| 1400 | L'objet n'est plus valide. |
| 1401 | L'adresse invalide pour le paramètre de renvoi a été spécifiée. |

### 17.7.1.18.1.8 CSharpSettings_ProjectType

Spécifie le type de projet C# utilisé par `Document.GenerateCHashCode`.

#### Signature

```
CSharpSettings_ProjectType : ENUMProjectType
```

#### Erreurs

| Code d'erreur | Description |
|---|---|
| 1400 | L'objet n'est plus valide. |

| Code d'erreur | Description |
|---|---|
| 1401 | L'adresse invalide pour le paramètre de renvoi a été spécifiée. |
| 1402 | La valeur de paramètre est en dehors de la plage. |
| 1403 | La valeur de paramètre n'est plus disponible. |

### *17.7.1.18.1.9 DefaultOutputByteOrder*

Ordre d'octet pour l'encodage utilisé pour les fichiers de sortie.

#### Signature

```
DefaultOutputByteOrder : String
```

#### Erreurs

| Code d'erreur | Description |
|---|---|
| 1400 | L'objet n'est plus valide. |
| 1401 | L'adresse invalide pour le paramètre de renvoi a été spécifiée. |

### *17.7.1.18.1.10 DefaultOutputByteOrderMark*

Indique si une marque d'ordre d'octet (BOM) doit être incluse dans l'encodage de fichier des fichiers de sortie.

#### Signature

```
DefaultOutputByteOrderMark : Booléen
```

#### Erreurs

| Code d'erreur | Description |
|---|---|
| 1400 | L'objet n'est plus valide. |
| 1401 | L'adresse invalide pour le paramètre de renvoi a été spécifiée. |

### 17.7.1.18.1.11 DefaultOutputEncoding

Encodage de fichier utilisé pour les fichiers de sortie.

#### Signature

```
DefaultOutputEncoding : String
```

#### Erreurs

| Code d'erreur | Description |
|---|---|
| 1400 | L'objet n'est plus valide. |
| 1401 | L'adresse invalide pour le paramètre de renvoi a été spécifiée. |

### 17.7.1.18.1.12 GenerateWrapperClasses

Indique si les classes de wrapper doivent également être générées lorsqu'elles génèrent du code.

#### Signature

```
GenerateWrapperClasses : Booléenne
```

#### Erreurs

| Code d'erreur | Description |
|---|---|
| 1400 | L'objet n'est plus valide. |
| 1401 | L'adresse invalide pour le paramètre de renvoi a été spécifiée. |

### 17.7.1.18.1.13 JavaSettings_ApacheAxisVersion (obsolete)

Cette propriété est obsolète.

#### Signature

```
JavaSettings_ApacheAxisVersion : ENUMApacheAxisVersion 1283
```

### *17.7.1.18.1.14 Parent*

L'objet parent selon le modèle d'objet.

#### Signature

```
Parent : Document
```

#### Erreurs

| Code d'erreur | Description |
|---|---|
| 1200 | L'objet n'est plus valide. |
| 1201 | L'adresse invalide pour le paramètre de renvoi a été spécifiée. |

### *17.7.1.18.1.15 ShowLogoOnPrint*

Afficher ou masquer le logo de MapForce sur des sorties imprimées.

#### Signature

```
ShowLogoOnPrint : Booléen
```

#### Erreurs

| Code d'erreur | Description |
|---|---|
| 1400 | L'objet n'est plus valide. |
| 1401 | L'adresse invalide pour le paramètre de renvoi a été spécifiée. |

### *17.7.1.18.1.16 ShowLogoOnStartup*

Afficher ou masquer le logo de MapForce sur démarrage de l'application.

#### Signature

```
ShowLogoOnStartup : Booléen
```

### Erreurs

| Code d'erreur | Description |
|---|---|
| 1400 | L'objet n'est plus valide. |
| 1401 | L'adresse invalide pour le paramètre de renvoi a été spécifiée. |

#### *17.7.1.18.1.17 UseGradientBackground*

Définit ou extrait le mode de couleur d'arrière-plan pour une fenêtre de mappage.

### Signature

```
UseGradientBackground : Booléen
```

### Erreurs

| Code d'erreur | Description |
|---|---|
| 1400 | L'objet n'est plus valide. |
| 1401 | L'adresse invalide pour le paramètre de renvoi a été spécifiée. |

#### *17.7.1.18.1.18 XSLTDefaultOutputDirectory*

Spécifie le répertoire cible où les fichiers générés par `Document.GenerateXSLT` sont placés.

### Signature

```
XSLTDefaultOutputDirectory : String
```

### Erreurs

| Code d'erreur | Description |
|---|---|
| 1400 | L'objet n'est plus valide. |
| 1401 | L'adresse invalide pour le paramètre de renvoi a été spécifiée. |

## 17.7.1.19 Projet

Un objet de `Projet` représente un projet et son arborescence d'éléments de projet dans MapForce.

Propriétés pour parcourir le modèle d'objet :

- `Application`
- `Parent`

Gestion de fichier :

- `FullName`
- `Nom`
- `Chemin`
- `Enregistré`
- `Enregistrer`
- `Fermer`

Navigation de l'arborescence de projet :

- `Compter`
- `Item`
- `_NewEnum`

Manipulation de l'arborescence de projet :

- `AddActiveFile`
- `AddFile`
- `InsertWebService` (uniquement Enterprise edition)
- `CreateFolder`

Génération de code :

- `Output_Folder`
- `Output_Language`
- `Output_TextEncoding`
- `Java_BasePackageName`
- `GenerateCode`
- `GenerateCodeEx`
- `GenerateCodeIn`
- `GenerateCodeInEx`

Pour des exemples sur comment utiliser les propriétés et méthodes recensées ci-dessous, voir Exemple : Tâches de projet[1133]. Notez que pour réaliser des opérations qui impliquent les services Web, MapForce Enterprise edition est requis.

## Propriétés

| Nom | Description |
|---|---|
| `_NewEnum`[1256] | Propriétés<br>Cette propriété prend en charge une énumération standard spécifique au langage. |
| `Application` | Lecture seule.<br>Extrait l'objet de l'application de premier niveau. |
| `Compter`[1273] | Lecture seule. |

| Nom | Description |
|---|---|
| | Extrait le nombre d'enfants de l'item racine du projet. Pour des exemples, voir `Item` ou `_NewEnum` |
| `FullName` 1258 | Chemin et nom du fichier de projet. |
| `Item` 1274 | Lecture seule.<br>Retourne l'enfant à la position *n* de la racine du projet. L'index est basé sur 1 (le premier index est 1). L'index le plus grand valide est `Count`. Pour une alternative, voir `_NewEnum`. |
| `Java_BasePackageName` 1258 | Définit ou obtient le nom du package de base des packages Java qui seront générés. Cette propriété est utilisée uniquement lorsque le code Java est généré. |
| `Nom` 1259 | Lecture seule.<br>Nom du fichier de projet sans chemin de fichier. |
| `Output_Folder` 1259 | Définit ou obtient le dossier de sortie par défaut utilisé avec `GenerateCode` et `GenerateCodeIn`. Les éléments de projet peuvent écraser cette valeur dans leur propriété `CodeGenSettings_OutputFolder`, quand `CodeGenSettings_UseDefault` est définie à false. |
| `Output_Language` 1260 | Définit ou obtient le langage par défaut pour la génération de code en utilisant `GenerateCode`. Les éléments de projet peuvent écraser cette valeur dans leur propriété `CodeGenSettings_OutputLanguage` quand `CodeGenSettings_UseDefault` est défini à false. |
| `Output_TextEncoding` 1260 | Définit ou obtient l'encodage de texte utilisé lorsque le code basé sur XML est généré. |
| `Parent` | Lecture seule.<br>L'objet parent selon le modèle d'objet. |
| `Chemin` 1261 | Lecture seule.<br>Chemin du fichier de projet sans nom. |
| `Enregistré` 1261 | Lecture seule.<br>**True** si le document n'a pas été modifié depuis la dernière opération de `sauvegarde`, **false**, si le contraire. |

## Méthodes

| Nom | Description |
|---|---|
| `AddActiveFile` 1262 | Ajoute le document actuellement ouvert au dossier de mappage de la racine du projet. |
| `AddFile` 1263 | Ajoute le document spécifié au dossier de mappage de la racine du projet. |

| Nom | Description |
|---|---|
| Fermer 1263 | Ferme le projet sans l'enregistrer. |
| CreateFolder 1264 | Crée un nouveau dossier en tant qu'enfant de l'item racine du projet. |
| GenerateCode 1264 | Génère le code pour tous les items de projet du projet. Le langage de code et l'emplacement de sortie sont déterminés par des propriétés du projet et des items de projet. |
| GenerateCodeEx 1264 | Génère le code pour tous les items de projet du projet. Le langage de code et l'emplacement de sortie sont déterminés par des propriétés du projet et des items de projet. Un objet qui peut être utilisé pour itérer à travers tous les messages publiés par le processus de générateur de code est retourné. Ces messages sont les mêmes que ceux affichés dans la fenêtre de messages de MapForce. |
| GenerateCodeIn 1265 | Génère le code pour tous les items de projet du projet dans le langage spécifié. L'emplacement de sortie est déterminé par les propriétés du projet et les items de projet. |
| GenerateCodeInEx 1265 | Génère le code pour tous les items de projet du projet dans le langage spécifié. L'emplacement de sortie est déterminé par les propriétés du projet et les items de projet. Renvoie un objet qui peut être utilisé pour itérer à travers tous les messages émis par le processus de génération de code. Ces messages sont les mêmes que ceux affichés dans la fenêtre **Messages** de MapForce. |
| InsertWebService 1266 | Insère un nouveau projet de service Web dans le dossier de service Web du projet. Si **i_bGenerateMappings** est true, les documents de mappage initial pour tous les ports sont générés automatiquement. |
| Enregistrer 1267 | Enregistre le projet vers le fichier défini par `FullName`. |

## Événements

| Nom | Description |
|---|---|
| OnProjectClosed 1267 | Cet événement est déclenché quand le projet est fermé. L'objet du projet passé dans le gestionnaire d'événement ne devrait pas être accédé. L'événement ouvert correspondant est `Application.OnProjectOpened`. |

### 17.7.1.19.1 Propriétés

#### *17.7.1.19.1.1 _NewEnum*

Cette propriété prend en charge une énumération standard spécifique au langage.

#### Signature

```
_NewEnum : IUnknown
```

#### Erreurs

| Code d'erreur : | Description |
|---|---|
| 1500 | L'objet n'est plus valide. |

#### Exemples

```
// ------------------------------------------------------------
// JScript sample - enumeration of a project's project items.
function AllChildrenOfProjectRoot()
{
    objProject = objMapForce.ActiveProject;
    if ( objProject != null )
    {

for ( objProjectIter = new Enumerator(objProject); ! objProjectIter.atEnd(); objProjectIt
er.moveNext() )
        {
            objProjectItem = objProjectItemIter.item();

            // do something with project item here
        }
    }
}
```

```
// ------------------------------------------------------------
// JScript sample - iterate all project items, depth first.
function IterateProjectItemsRec(objProjectItemIter)
{
    while ( ! objProjectItemIter.atEnd() )
    {
        objProjectItem = objProjectItemIter.item();
        // do something with project item here

        IterateProjectItemsRec( new Enumerator(objProjectItem) );

        objProjectItemIter.moveNext();
    }
```

```
}
function IterateAllProjectItems()
{
    objProject = objMapForce.ActiveProject;
    if ( objProject != null )
    {
        IterateProjectItemsRec( new Enumerator(objProject) );
    }
}
```

### *17.7.1.19.1.2 Application*

Extrait l'objet de l'application de premier niveau.

#### Signature

```
Application : Application
```

#### Erreurs

| Code d'erreur : | Description |
|---|---|
| 1200 | L'objet n'est plus valide. |
| 1201 | L'adresse invalide pour le paramètre de retour a été spécifiée. |

### *17.7.1.19.1.3 Count*

Extrait le nombre de mappages dans la collection.

#### Signature

```
Count : Entier
```

#### Erreurs

| Code d'erreur | Description |
|---|---|
| 1200 | L'objet n'est plus valide. |
| 1201 | L'adresse invalide pour le paramètre de renvoi a été spécifiée. |

#### *17.7.1.19.1.4 FullName*

Chemin et nom du fichier de projet.

### Signature

```
FullName : String
```

### Erreurs

| Code d'erreur | Description |
|---|---|
| 1500 | Cet objet n'est plus valide. |
| 1501 | L'adresse invalide a été spécifiée pour le paramètre de renvoi. |

#### *17.7.1.19.1.5 Item*

Extrait le mappage à l'index `n` de la collection. Les index commencent par 1.

### Signature

```
Item(in n:Integer) : Mappage
```

### Erreurs

| Code d'erreur | Description |
|---|---|
| 1200 | L'objet n'est plus valide. |
| 1201 | L'adresse invalide pour le paramètre de renvoi a été spécifiée. |

#### *17.7.1.19.1.6 Java_BasePackageName*

Définit ou obtient le nom du package de base des packages Java qui seront générés. Cette propriété est utilisée uniquement lorsque le code Java est généré.

### Signature

```
Java_BasePackageName : String
```

### Erreurs

| Code d'erreur | Description |
|---|---|
| 1500 | L'objet n'est plus valide. |
| 1501 | Nom du pack invalide spécifié. L'adresse invalide pour le paramètre de renvoi a été spécifiée. |

## *17.7.1.19.1.7 Nom*

Nom du fichier de projet sans chemin de fichier.

### Signature

```
Nom : String
```

### Erreurs

| Code d'erreur | Description |
|---|---|
| 1500 | L'objet n'est plus valide. |
| 1501 | L'adresse invalide pour le paramètre de renvoi a été spécifiée. |

## *17.7.1.19.1.8 Output_Folder*

Définit ou obtient le dossier de sortie par défaut utilisé avec `GenerateCode` et `GenerateCodeIn`. Les éléments de projet peuvent écraser cette valeur dans leur propriété `CodeGenSettings_OutputFolder`, quand `CodeGenSettings_UseDefault` est définie à false.

### Signature

```
Output_Folder : String
```

### Erreurs

| Code d'erreur | Description |
|---|---|
| 1500 | L'objet n'est plus valide. |
| 1501 | Nom de dossier invalide spécifié. L'adresse invalide pour le paramètre de renvoi a été spécifiée. |

### 17.7.1.19.1.9 Output_Language

Définit ou obtient le langage par défaut pour la génération de code en utilisant `GenerateCode`. Les éléments de projet peuvent écraser cette valeur dans leur propriété `CodeGenSettings_OutputLanguage` quand `CodeGenSettings_UseDefault` est défini à false.

#### Signature

```
Output_Language : ENUMProgrammingLanguage
```

#### Erreurs

| Code d'erreur | Description |
|---|---|
| 1500 | L'objet n'est plus valide. |
| 1501 | Le langage invalide est spécifié. L'adresse invalide pour le paramètre de renvoi a été spécifiée. |

### 17.7.1.19.1.10 Output_TextEncoding

Définit ou obtient l'encodage de texte utilisé lorsque le code basé sur XML est généré.

#### Signature

```
Output_TextEncoding : String
```

#### Erreurs

| Code d'erreur | Description |
|---|---|
| 1500 | L'objet n'est plus valide. |
| 1501 | L'encodage de texte invalide est spécifié. L'adresse invalide pour le paramètre de renvoi a été spécifiée. |

### *17.7.1.19.1.11 Parent*

L'objet parent selon le modèle d'objet.

#### Signature

```
Parent : Document
```

#### Erreurs

| Code d'erreur | Description |
|---|---|
| 1200 | L'objet n'est plus valide. |
| 1201 | L'adresse invalide pour le paramètre de renvoi a été spécifiée. |

### *17.7.1.19.1.12 Path*

Chemin du fichier de projet sans nom.

#### Signature

```
Path : String
```

#### Erreurs

| Code d'erreur | Description |
|---|---|
| 1500 | L'objet n'est plus valide. |
| 1501 | L'adresse invalide pour le paramètre de renvoi a été spécifiée. |

### *17.7.1.19.1.13 Enregistré*

**True** si le document n'a pas été modifié depuis la dernière opération de `sauvegarde`, **false**, si le contraire.

#### Signature

```
Enregistré : Booléenne
```

### Erreurs

| Code d'erreur | Description |
|---|---|
| 1500 | L'objet n'est plus valide. |
| 1501 | L'adresse invalide pour le paramètre de renvoi a été spécifiée. |

## 17.7.1.19.2 Méthodes

### *17.7.1.19.2.1 AddActiveFile*

Ajoute le document actuellement ouvert au dossier de mappage de la racine du projet.

### Signature

```
AddActiveFile() -> ProjectItem
```

### Erreurs

| Code d'erreur : | Description |
|---|---|
| 1500 | L'objet n'est plus valide. |
| 1501 | L'adresse invalide pour le paramètre de retour a été spécifiée. |
| 1503 | Aucun document valide n'est disponible. |
| 1504 | Un nom de chemin doit être attribué aux documents actifs avant qu'il ne puisse être ajouté au projet. |
| 1705 | Le mappage n'a pas pu être attribué au projet. Peut-être est-il déjà contenu dans le dossier cible. |

### *17.7.1.19.2.2 AddFile*

Ajoute le document spécifié au dossier de mappage de la racine du projet.

#### Signature

```
AddFile(in i_strFileName:String) -> ProjectItem
```

#### Paramètres

| Nom | Type | Description |
|---|---|---|
| **i_strFileName** | `String` | Spécifie le chemin d'accès du document à ajouter. |

#### Erreurs

| Code d'erreur : | Description |
|---|---|
| 1500 | L'objet n'est plus valide. |
| 1501 | L'adresse invalide pour le paramètre de retour a été spécifiée. |
| 1705 | Le mappage n'a pas pu être attribué au projet. Le fichier n'existe pas ou n'est pas un mappage de MapForce. Peut-être que le fichier est déjà attribué au dossier cible. |

### *17.7.1.19.2.3 Fermer*

Ferme le projet sans l'enregistrer.

#### Signature

```
Fermer() -> Résilier
```

#### Erreurs

| Code d'erreur | Description |
|---|---|
| 1500 | L'objet n'est plus valide. |

### 17.7.1.19.2.4 CreateFolder

Crée un nouveau dossier en tant qu'enfant de l'item racine du projet.

#### Signature

```
CreateFolder(dans i_strFileName:String) -> ProjectItem
```

#### Paramètres

| Nom | Type | Description |
|---|---|---|
| **i_strFolderName** | `String` | Le nom du dossier à créer. |

#### Erreurs

| Code d'erreur | Description |
|---|---|
| 1500 | L'objet n'est plus valide. |
| 1501 | Le nom du dossier invalide ou l'adresse invalide pour le paramètre de renvoi a été spécifié. |

### 17.7.1.19.2.5 GenerateCode

Génère le code pour tous les items de projet du projet. Le langage de code et l'emplacement de sortie sont déterminés par des propriétés du projet et des items de projet.

#### Signature

```
GenerateCode() -> Void
```

#### Erreurs

| Code d'erreur | Description |
|---|---|
| 1500 | L'objet n'est plus valide. |
| 1706 | Erreur pendant la génération de code. |

### 17.7.1.19.2.6 GenerateCodeEx

Génère le code pour tous les items de projet du projet. Le langage de code et l'emplacement de sortie sont déterminés par des propriétés du projet et des items de projet. Un objet qui peut être utilisé pour itérer à travers

tous les messages publiés par le processus de générateur de code est retourné. Ces messages sont les mêmes que ceux affichés dans la fenêtre de messages de MapForce.

### Signature

```
GenerateCodeEx() -> ErrorMarkers
```

### Erreurs

| Code d'erreur | Description |
|---|---|
| 1500 | L'objet n'est plus valide. |
| 1501 | Adresse invalide a été spécifiée pour le paramètre de renvoi. |
| 1706 | Erreur pendant la génération de code. |

## *17.7.1.19.2.7 GenerateCodeIn*

Génère le code pour tous les items de projet du projet dans le langage spécifié. L'emplacement de sortie est déterminé par les propriétés du projet et de l'élément de projet..

### Signature

```
GenerateCodeIn(in i_nLanguage:ENUMProgrammingLanguage[1288]) -> Void
```

### Paramètres

| Nom | Type | Description |
|---|---|---|
| **i_nLanguage** | ENUMProgrammingLanguage[1288] | Spécifie le langage de programmation dans lequel le code devrait être généré. |

### Erreurs

| Code d'erreur | Description |
|---|---|
| 1500 | L'objet n'est plus valide. |
| 1706 | Erreur pendant la génération de code. |

## *17.7.1.19.2.8 GenerateCodeInEx*

Génère le code pour tous les items de projet du projet dans le langage spécifié. L'emplacement de sortie est déterminé par les propriétés du projet et de l'élément de projet. Renvoie un objet qui peut être utilisé pour itérer

à travers tous les messages émis par le processus de génération de code. Ces messages sont les mêmes que ceux affichés dans la fenêtre **Messages** de MapForce.

### Signature

```
GenerateCodeInEx(in i_nLanguage:ENUMProgrammingLanguage1288) -> ErrorMarkers
```

### Paramètres

| Nom | Type | Description |
|---|---|---|
| **i_nLanguage** | ENUMProgrammingLanguage1288 | Spécifie le langage de programmation dans lequel le code devrait être généré. |

### Erreurs

| Code d'erreur | Description |
|---|---|
| 1500 | L'objet n'est plus valide. |
| 1501 | L'adresse invalide pour le paramètre de renvoi a été spécifiée. |
| 1706 | Erreur pendant la génération de code. |

## *17.7.1.19.2.9 InsertWebService*

Insère un nouveau projet de service Web dans le dossier de service Web du projet. Si **i_bGenerateMappings** est true, les documents de mappage initial pour tous les ports sont générés automatiquement.

### Signature

```
InsertWebService(in i_strWSDLFile:String, in i_strService:String, dans i_strPort:String,
in i_bGenerateMappings:Boolean) -> ProjectItem
```

### Paramètres

| Nom | Type | Description |
|---|---|---|
| **i_strWSDLFile** | String | Spécifie le chemin du fichier WSDL à ajouter. |
| **i_strService** | String | Spécifie le nom du service Web à ajouter. |
| **i_strPort** | String | Spécifie le port du service Web à ajouter. |

| Nom | Type | Description |
|---|---|---|
| **i_bGenerateMappings** | **Booléenne** | Si ce paramètre est true, les documents de mappage initial pour tous les ports sont générés automatiquement. |

### Erreurs

| Code d'erreur | Description |
|---|---|
| 1500 | L'objet n'est plus valide. |
| 1501 | Le fichier WDSL ne peut pas être trouvé ou est invalide. Le service ou les noms de port sont invalides. L'adresse invalide pour le paramètre de renvoi a été spécifiée. |
| 1503 | L'opération n'est pas prise en charge par l'édition actuelle. |

#### *17.7.1.19.2.10 Enregistrer*

Enregistre le projet vers le fichier défini par `FullName`.

### Signature

```
Enregistrer() -> Void
```

### Erreurs

| Code d'erreur | Description |
|---|---|
| 1500 | L'objet n'est plus valide. |
| 1502 | Ne peut pas enregistrer le fichier. |

## 17.7.1.19.3 Événements

#### *17.7.1.19.3.1 OnProjectClosed*

Cet événement est déclenché quand le projet est fermé. L'objet du projet passé dans le gestionnaire d'événement ne devrait pas être accédé. L'événement ouvert correspondant est `Application.OnDocumentOpened`.

### Signature

```
OnProjectClosed(in i_ipProject:Project) : Annuler
```

## 17.7.1.20 ProjectItem

Un objet `ProjectItem` représente un élément dans une arborescence de projet.

Propriétés pour parcourir le modèle d'objet :

- `Application`
- `Parent`

Navigation de l'arborescence de projet :

- `Compter`
- `Item`
- `_NewEnum`

Propriétés d'élément de projet :

- `Genre`
- `Nom`
- `WSDLFile` (uniquement disponible pour les éléments de projet du service Web)
- `QualifiedName` (uniquement disponible pour les éléments de projet du service Web)

Manipulation de l'arborescence de projet :

- `AddActiveFile` (uniquement disponible pour les éléments de dossier)
- `AddFile` (uniquement disponible pour les éléments de dossier)
- `CreateFolder` (uniquement disponible pour les éléments de dossier)
- `CreateMappingForProject` (uniquement disponible pour les opérations du service Web)
- `Supprimer`

Accès au document :

- `Ouvrir` (uniquement disponible pour les éléments de mappage et opérations du service Web)

Code-generation :

- `CodeGenSettings_UseDefault`
- `CodeGenSettings_OutputFolder`
- `CodeGenSettings_Language`
- `GenerateCode`
- `GenerateCodeEx`
- `GenerateCodeIn`
- `GenerateCodeInEx`

Pour des exemples sur comment utiliser les propriétés et méthodes recensées ci-dessous, voir Exemple : Tâches de projet 1133. Pour des opérations avec les services Web, MapForce Enterprise edition est requis.

### Propriétés

| Nom | Description |
|---|---|
| `_NewEnum` 1271 | Lecture seule. |

| Nom | Description |
|---|---|
| | Cette propriété prend en charge une énumération standard spécifique au langage. Pour des exemples, voir `Project.Item` ou `Project._NewEnum`. |
| `Application` | Lecture seule.<br>Extrait l'objet de l'application de premier niveau. |
| `CodeGenSettings_Language` 1272 | Obtient ou définit le langage à utiliser avec `GenerateCode` ou `Project.GenerateCode`. Cette propriété est consultée uniquement si `CodeGenSettings_UseDefault` est défini comme false. |
| `CodeGenSettings_OutputFolder` 1272 | Obtient ou définit le répertoire de sortie à utiliser avec `GenerateCode`, `GenerateCodeIn`, `Project.GenerateCode` ou `Project.GenerateCodeIn`. Cette propriété est consultée uniquement si `CodeGenSettings_UseDefault` est défini comme false. |
| `CodeGenSettings_UseDefault` 1273 | Obtient ou définit si le répertoire de sortie et le langage de code sont utilisés tels que définis soit par (a) les dossiers parents, ou (b) la racine du projet. Cette propriété est utilisée avec des appels pour `GenerateCode`, `GenerateCodeIn`, `Project.GenerateCode` et `Project.GenerateCodeIn`. Si la propriété est définie en tant que false, les valeurs de `CodeGenSettings_OutputFolder` et `CodeGenSettings_Language` sont utilisées pour générer le code pour cet item de projet. |
| `Compter` | Lecture seule.<br>Extrait le nombre d'enfants de l'item de ce projet. Voir aussi `Item`. Pour des exemples, voir `Project.Item` ou `Project._NewEnum`. |
| `Item` | Lecture seule.<br>Retourne l'enfant à la position `n` de cet item de projet. L'index est basé sur 1 (le premier index est 1). L'index valide le plus grand est `ProjectItem.Count`. Pour une alternative, voir `ProjectItem._NewEnum`. Pour des exemples, voir `Project.Item` ou `Project._NewEnum`. |
| `Genre` 1274 | Propriétés<br>Extrait le genre d'élément de projet. La disponibilité de quelques propriétés et l'applicabilité de certaines méthodes sont restreintes aux genres spécifiques d'éléments de projet. La description de toutes les méthodes et propriétés contient des informations sur ces restrictions. |
| `Nom` 1275 | Extrait ou définit le nom d'un élément de projet. Le nom de la plupart des éléments est en lecture seule. Les exceptions sont les dossiers créés par l'utilisateur, dont les noms peuvent être altérés après la création. |

| Nom | Description |
|---|---|
| Parent | Lecture seule.<br>Extrait le projet dont cet item est un enfant. A la même effect en tant que Application.ActiveProject. |
| QualifiedName 1275 | Propriétés<br>Extrait le nom qualifié d'un élément de service Web. |
| WSDLFile 1276 | Propriétés<br>Extrait le nom du fichier du fichier WSDL définissant le service Web qui héberge l'item de projet actuel. |

## Méthodes

| Nom | Description |
|---|---|
| AddActiveFile 1276 | Ajoute le document actuellement actif à cet item de projet s'il s'agit d'un enfant valide. Autrement, le document est ajouté au dossier de mappage de la racine de projet. |
| AddFile 1277 | Ajoute le document spécifié à cet item de projet s'il s'agit d'un enfant valide. Autrement, le document est ajouté au dossier de mappage de la racine de projet. |
| CreateFolder 1278 | Crée un nouveau dossier en tant qu'enfant de cet item de projet. |
| CreateMappingForProject 1278 | Crée un document de mappage initial pour une opération de service Web et l'enregistre dans **i_strFileName**. Lorsque vous utilisez Project.InsertWebService, vous pouvez définir l'indicateur **i_bGenerateMappings** de telle façon que MapForce puisse générer automatiquement les mappages initiaux pour tous les ports. |
| GenerateCode 1279 | Génère le code pour tous les items de projet et leurs enfants. Le langage de code et l'emplacement de sortie sont déterminés par CodeGenSettings_UseDefault, CodeGenSettings_Language and CodeGenSettings_OutputFolder. Les enfants de cet item de projet peuvent avoir leurs propres paramètres de propriété associés à la génération-code. |
| GenerateCodeEx 1279 | Génère le code pour tous les items de projet et leurs enfants. Le langage de code et l'emplacement de sortie sont déterminés par CodeGenSettings_UseDefault, CodeGenSettings_Language et CodeGenSettings_OutputFolder. Les enfants de cet item de projet peuvent avoir leurs propres paramètres de propriété associés à la génération-code. |
| GenerateCodeIn 1280 | Génère le code pour l'élément de projet et ses enfants dans le langage spécifié. L'emplacement de sortie est déterminé par CodeGenSettings_UseDefault et CodeGenSettings_OutputFolder. Les enfants de cet item de |

| Nom | Description |
|---|---|
| | projet peuvent avoir leurs propres paramètres de propriété associés à la génération-code. |
| GenerateCodeInEx 1280 | Génère le code pour l'élément de projet et ses enfants dans le langage spécifié. L'emplacement de sortie est déterminé par `CodeGenSettings_UseDefault` et `CodeGenSettings_OutputFolder`. Les enfants de cet item de projet peuvent avoir leurs propres paramètres de propriété associés à la génération-code.<br><br>Un objet qui peut être utilisé pour itérer à travers tous les messages publiés par le processus de génération de code est retourné. Ces messages sont les mêmes que ceux affichés dans la fenêtre de messages de MapForce. |
| Ouvrir 1281 | Ouvre l'élément de projet comme document ou rend le document correspondant actif, s'il est déjà ouvert. L'élément de projet doit être un mappage MapForce ou, pour Enterprise edition uniquement, une opération de service Web. |
| Supprimer 1282 | Supprimer cet élément de projet et tous ses enfants depuis l'arborescence du projet. |

## Événements

| Nom | Description |
|---|---|
| OnModifiedFlagChanged 1283 | Apparaît lorsque le statut de modification `ProjectItem` change. |
| OnProjectClosed 1283 | Cet événement est déclenché quand le projet est fermé. L'objet du projet passé dans le gestionnaire d'événement ne devrait pas être accédé. L'événement ouvert correspondant est `Application.OnProjectOpened`. |

### 17.7.1.20.1 Propriétés

#### *17.7.1.20.1.1 _NewEnum*

Cette propriété prend en charge une énumération standard spécifique au langage. Pour des exemples, voir `Project.Item` ou `Project._NewEnum`.

## Signature

```
_NewEnum : IUnknown
```

### Erreurs

| Code d'erreur : | Description |
|---|---|
| 1700 | L'objet n'est plus valide. |

## *17.7.1.20.1.2 Application*

Extrait l'objet de l'application de premier niveau.

### Signature

```
Application : Application
```

### Erreurs

| Code d'erreur : | Description |
|---|---|
| 1400 | L'objet n'est plus valide. |
| 1401 | L'adresse invalide pour le paramètre de retour a été spécifiée. |

## *17.7.1.20.1.3 CodeGenSettings_Language*

Obtient ou définit le langage à utiliser avec `GenerateCode` ou `Project.GenerateCode`. Cette propriété est consultée uniquement si `CodeGenSettings_UseDefault` est défini comme false.

### Signature

```
CodeGenSettings_Language : ENUMProgrammingLanguage 1288
```

### Erreurs

| Code d'erreur | Description |
|---|---|
| 1700 | L'objet n'est plus valide. |
| 1701 | Le langage invalide ou l'adresse invalide pour le paramètre de renvoi a été spécifié. |

## *17.7.1.20.1.4 CodeGenSettings_OutputFolder*

Obtient ou définit le répertoire de sortie à utiliser avec `GenerateCode`, `GenerateCodeIn`, `Project.GenerateCode` ou `Project.GenerateCodeIn`. Cette propriété est consultée uniquement si `CodeGenSettings_UseDefault` est défini comme false.

### Signature

```
CodeGenSettings_OutputFolder : String
```

### Erreurs

| Code d'erreur | Description |
|---|---|
| 1700 | L'objet n'est plus valide. |
| 1701 | Le langage invalide ou l'adresse invalide pour le paramètre de renvoi a été spécifié. |

## *17.7.1.20.1.5 CodeGenSettings_UseDefault*

Obtient ou définit si le répertoire de sortie et le langage de code sont utilisés tels que définis soit par (a) les dossiers parents, ou (b) la racine du projet. Cette propriété est utilisée avec des appels pour `GenerateCode`, `GenerateCodeIn`, `Project.GenerateCode` et `Project.GenerateCodeIn`. Si la propriété est définie en tant que false, les valeurs de `CodeGenSettings_OutputFolder` et `CodeGenSettings_Language` sont utilisées pour générer le code pour cet item de projet.

### Signature

```
CodeGenSettings_UseDefault : Booléen
```

### Erreurs

| Code d'erreur | Description |
|---|---|
| 1700 | L'objet n'est plus valide. |
| 1701 | L'adresse invalide pour le paramètre de renvoi a été spécifiée. |

## *17.7.1.20.1.6 Count*

Extrait le nombre d'enfants de l'item racine du projet. Pour des exemples, voir `Item` ou `_NewEnum`

### Signature

```
Count : Entier
```

### Erreurs

| Code d'erreur | Description |
|---|---|
| 1500 | L'objet n'est plus valide. |

### *17.7.1.20.1.7 Item*

Retourne l'enfant à la position *n* de la racine du projet. L'index est basé sur 1 (le premier index est 1). L'index le plus grand valide est `Count`. Pour une alternative, voir `_NewEnum`.

#### Signature

```
Item(in n:Integer) : ProjectItem
```

#### Erreurs

| Code d'erreur | Description |
|---|---|
| 1500 | L'objet n'est plus valide. |

#### Exemples

```
// ------------------------------------------------------
// JScript code snippet - énumérer les enfants en utilisant Count et Item.
for( nItemIndex = 1; nItemIndex <= objProject.Count; nItemIndex++ )
{
    objProjectItem = objProject.Item(nItemIndex);
    // faire quelque chose avec l'item de projet ici
}
```

### *17.7.1.20.1.8 Genre*

Extrait le genre d'élément de projet. La disponibilité de quelques propriétés et l'applicabilité de certaines méthodes sont restreintes aux genres spécifiques d'éléments de projet. La description de toutes les méthodes et propriétés contient des informations sur ces restrictions.

#### Signature

```
Genre : ENUMProjectItemType 1288
```

#### Erreurs

| Code d'erreur | Description |
|---|---|
| 1700 | L'objet n'est plus valide. |
| 1701 | L'adresse invalide pour le paramètre de renvoi a été spécifiée. |

### *17.7.1.20.1.9 Nom*

Extrait ou définit le nom d'un élément de projet. Le nom de la plupart des éléments est en lecture seule. Les exceptions sont les dossiers créés par l'utilisateur, dont les noms peuvent être altérés après la création.

#### Signature

```
Nom : String
```

#### Erreurs

| Code d'erreur | Description |
|---|---|
| 1700 | L'objet n'est plus valide. |
| 1701 | L'adresse invalide pour le paramètre de renvoi a été spécifiée. |
| 1702 | L'élément de projet ne permet pas d'altérer son nom. |

### *17.7.1.20.1.10 Parent*

L'objet parent selon le modèle d'objet.

#### Signature

```
Parent : Application
```

#### Erreurs

| Code d'erreur | Description |
|---|---|
| 1400 | L'objet n'est plus valide. |
| 1401 | L'adresse invalide pour le paramètre de renvoi a été spécifiée. |

### *17.7.1.20.1.11 QualifiedName*

Extrait le nom qualifié d'un élément de service Web.

#### Signature

```
QualifiedName : String
```

### Erreurs

| Code d'erreur | Description |
|---|---|
| 1700 | L'objet n'est plus valide. |
| 1701 | L'adresse invalide pour le paramètre de renvoi a été spécifiée. |
| 1702 | L'élément de projet ne fait pas partie du service Web. |

#### *17.7.1.20.1.12 WSDLFile*

Extrait le nom du fichier du fichier WSDL définissant le service Web qui héberge l'item de projet actuel.

### Signature

```
WSDLFile : String
```

### Erreurs

| Code d'erreur | Description |
|---|---|
| 1700 | L'objet n'est plus valide. |
| 1701 | L'adresse invalide pour le paramètre de renvoi a été spécifiée. |
| 1702 | L'élément de projet ne fait pas partie du service Web. |

## 17.7.1.20.2 Méthodes

#### *17.7.1.20.2.1 AddActiveFile*

Ajoute le document actuellement actif à cet item de projet s'il s'agit d'un enfant valide. Autrement, le document est ajouté au dossier de mappage de la racine de projet.

### Signature

```
AddActiveFile() -> ProjectItem
```

### Erreurs

| Code d'erreur : | Description |
|---|---|
| 1700 | L'objet n'est plus valide. |

| Code d'erreur : | Description |
|---|---|
| 1701 | Le nom du fichier est vide. L'adresse invalide pour le paramètre de retour a été spécifiée. |
| 1703 | Aucun document valide n'est disponible. |
| 1704 | Un nom de chemin doit être attribué aux documents actifs avant qu'il ne puisse être ajouté au projet. |
| 1705 | Le mappage n'a pas pu être attribué au projet. Le fichier n'existe pas ou n'est pas un mappage de MapForce. Peut-être que le fichier est déjà attribué au dossier cible. |

### *17.7.1.20.2.2 AddFile*

Ajoute le document spécifié à cet item de projet s'il s'agit d'un enfant valide. Autrement, le document est ajouté au dossier de mappage de la racine de projet.

#### Signature

```
AddFile(dans i_strFileName:String) -> ProjectItem
```

#### Paramètres

| Nom | Type | Description |
|---|---|---|
| **i_strFilePath** | `String` | Le chemin du document à ajouter. |

#### Erreurs

| Code d'erreur : | Description |
|---|---|
| 1700 | L'objet n'est plus valide. |
| 1701 | Le nom du fichier est vide. L'adresse invalide pour le paramètre de retour a été spécifiée. |
| 1705 | Le mappage n'a pas pu être attribué au projet. Le fichier n'existe pas ou n'est pas un mappage de MapForce. Peut-être que le fichier est déjà attribué au dossier cible. |

### 17.7.1.20.2.3 CreateFolder

Crée un nouveau dossier en tant qu'enfant de cet item de projet.

#### Signature

```
CreateFolder(dans i_strFileName:String) -> ProjectItem
```

#### Paramètres

| Nom | Type | Description |
|---|---|---|
| **i_strFolderName** | `String` | Le nom du dossier à créer. |

#### Erreurs

| Code d'erreur | Description |
|---|---|
| 1700 | L'objet n'est plus valide. |
| 1701 | Le nom du dossier invalide ou l'adresse invalide pour le paramètre de renvoi a été spécifié. |
| 1702 | L'item de projet ne prend pas en charge les enfants. |

### 17.7.1.20.2.4 CreateMappingForProject

Crée un document de mappage initial pour une opération de service Web et l'enregistre dans **i_strFileName**. Lorsque vous utilisez `Project.InsertWebService`, vous pouvez définir l'indicateur **i_bGenerateMappings** de telle façon que MapForce puisse générer automatiquement les mappages initiaux pour tous les ports.

#### Signature

```
CreateMappingForProject(dans i_strFileName:String) -> ProjectItem
```

#### Paramètres

| Nom | Type | Description |
|---|---|---|
| **i_strFileName** | `String` | Spécifie le chemin dans lequel le mappage devrait être enregistré. |

### Erreurs

| Code d'erreur | Description |
|---|---|
| 1700 | L'objet n'est plus valide. |
| 1701 | L'adresse invalide pour le paramètre de renvoi a été spécifiée. |
| 1707 | Ne peut pas créer de nouveau mappage. L'item de projet ne prend pas en charge la création automatique de mappages initiaux ou un mappage existe déjà. |
| 1708 | L'opération n'est pas prise en charge dans l'édition actuelle. |

## *17.7.1.20.2.5 GenerateCode*

Génère le code pour tous les items de projet et leurs enfants. Le langage de code et l'emplacement de sortie sont déterminés par `CodeGenSettings_UseDefault`, `CodeGenSettings_Language` and `CodeGenSettings_OutputFolder`. Les enfants de cet item de projet peuvent avoir leurs propres paramètres de propriété associés à la génération-code.

### Signature

```
GenerateCode() -> Void
```

### Erreurs

| Code d'erreur | Description |
|---|---|
| 1700 | L'objet n'est plus valide. |
| 1706 | Erreur pendant la génération de code. |

## *17.7.1.20.2.6 GenerateCodeEx*

Génère le code pour tous les items de projet et leurs enfants. Le langage de code et l'emplacement de sortie sont déterminés par `CodeGenSettings_UseDefault`, `CodeGenSettings_Language` and `CodeGenSettings_OutputFolder`. Les enfants de cet item de projet peuvent avoir leurs propres paramètres de propriété associés à la génération-code.

### Signature

```
GenerateCodeEx() -> ErrorMarkers
```

### Erreurs

| Code d'erreur | Description |
|---|---|
| 1700 | L'objet n'est plus valide. |
| 1701 | L'adresse invalide pour le paramètre de renvoi a été spécifiée. |
| 1706 | Erreur pendant la génération de code. |

## *17.7.1.20.2.7 GenerateCodeIn*

Génère le code pour l'élément de projet et ses enfants dans le langage spécifié. L'emplacement de sortie est déterminé par `CodeGenSettings_UseDefault` et `CodeGenSettings_OutputFolder`. Les enfants de cet item de projet peuvent avoir leurs propres paramètres de propriété associés à la génération-code.

### Signature

```
GenerateCodeIn(dans i_nLanguage:ENUMProgrammingLanguage1288) -> Void
```

### Paramètres

| Nom | Type | Description |
|---|---|---|
| **i_nLanguage** | ENUMProgrammingLanguage1288 | Spécifie le langage de programmation pour la génération de code. |

### Erreurs

| Code d'erreur | Description |
|---|---|
| 1700 | L'objet n'est plus valide. |
| 1701 | Le langage invalide est spécifié. |
| 1706 | Erreur pendant la génération de code. |

## *17.7.1.20.2.8 GenerateCodeInEx*

Génère le code pour l'élément de projet et ses enfants dans le langage spécifié. L'emplacement de sortie est déterminé par `CodeGenSettings_UseDefault` et `CodeGenSettings_OutputFolder`. Les enfants de cet item de projet peuvent avoir leurs propres paramètres de propriété associés à la génération-code.

Un objet qui peut être utilisé pour itérer à travers tous les messages publiés par le processus de génération de code est retourné. Ces messages sont les mêmes que ceux affichés dans la fenêtre de messages de MapForce.

### Signature

```
GenerateCodeInEx(in i_nLanguage:ENUMProgrammingLanguage1288) -> ErrorMarkers
```

### Paramètres

| Nom | Type | Description |
|---|---|---|
| **i_nLanguage** | ENUMProgrammingLanguage1288 | Spécifie le langage de programmation pour la génération de code. |

### Erreurs

| Code d'erreur | Description |
|---|---|
| 1700 | L'objet n'est plus valide. |
| 1701 | Le langage invalide spécifié ou l'adresse invalide pour le paramètre de renvoi a été spécifié. |
| 1706 | Erreur pendant la génération de code. |

## *17.7.1.20.2.9 Ouvrir*

Ouvre l'élément de projet comme document ou rend le document correspondant actif, s'il est déjà ouvert. L'élément de projet doit être un mappage MapForce ou, pour Enterprise edition uniquement, une opération de service Web.

### Signature

```
Ouvrir() -> Document
```

### Erreurs

| Code d'erreur | Description |
|---|---|
| 1700 | L'objet n'est plus valide. |
| 1701 | L'adresse invalide pour le paramètre de renvoi a été spécifiée. |
| 1702 | L'élément de projet ne se réfère pas à un fichier de mappage MapForce. |
| 1708 | L'opération n'est pas prise en charge dans l'édition actuelle. |

### *17.7.1.20.2.10 Supprimer*

Supprime une référence bibliothèque depuis la fenêtre **Gérer les Bibliothèques**.

## Signature

```
Remove(in i_strFileName:String) -> Void
```

## Paramètres

| Nom | Type | Description |
|---|---|---|
| **i_strFileName** | `String` | Le chemin du fichier bibliothèque à supprimer. Veuillez noter que le chemin doit refléter exactement le statut actuel (le plus récent) de l'objet `LibraryImport`. Rappelez-vous que le chemin peut être relatif ou absolu, et il peut avoir changé si vous avez enregistré le document, dépendant de l'indicateur `LibraryImport.SaveRelativePath`. Pour cette raison, si l'objet `LibraryImport` contient actuellement un chemin relatif, alors vous devriez fournir un chemin relatif comme valeur de ce paramètre. Autrement, la bibliothèque n'est pas trouvée et la méthode `Supprimer` échoue.<br><br>Ce qui est expliqué ci-dessus ne s'applique que pour les bibliothèques importées localement. En ce qui concerne les bibliothèques importées globalement, le chemin doit toujours être absolu (puisque les bibliothèques importées globalement ne peuvent pas avoir de chemin relatif) |

## Erreurs

| Code d'erreur | Description |
|---|---|
| 2400 | L'objet n'est plus valide. |

#### 17.7.1.20.3 Événements

##### *17.7.1.20.3.1 OnModifiedFlagChanged*

Apparaît lorsque le statut de modification `ProjectItem` change.

###### Signature

```
OnModifiedFlagChanged(dans i_bIsModified:Boolean) : Annuler
```

##### *17.7.1.20.3.2 OnProjectClosed*

Cet événement est déclenché quand le projet est fermé. L'objet du projet passé dans le gestionnaire d'événement ne devrait pas être accédé. L'événement ouvert correspondant est `Application.OnProjectOpened`.

###### Signature

```
OnProjectClosed(in i_ipProject:Project) : Annuler
```

## 17.7.2 Énumérations

### 17.7.2.1 ENUMApacheAxisVersion (obsolete)

Ce type d'énumération est obsolète.

#### Membres

eApacheAxisVersion_Axis = 1

eApacheAxisVersion_Axis2 = 2

### 17.7.2.2 ENUMApplicationStatus

Valeurs d'énumération pour indiquer le statut de l'application.

#### Membres

eApplicationRunning = 0

eApplicationAfterLicenseCheck = 1

eApplicationBeforeLicenseCheck = 2

eApplicationConcurrentLicenseCheckFailed = 3

eApplicationProcessingCommandLine = 4

### 17.7.2.3 ENUMAppOutputLine_Severity

Valeurs d'énumération pour identifier la gravité de `AppOutputLine`.

#### Membres

eSeverity_Undefined = -1

eSeverity_Info = 0

eSeverity_Warning = 1

eSeverity_Error = 2

eSeverity_CriticalError = 3

eSeverity_Success = 4

eSeverity_Summary = 5

eSeverity_Progress = 6

eSeverity_DataEdit = 7

eSeverity_ParserInfo = 8

eSeverity_PossibleInconsistencyWarning = 9

eSeverity_Message = 10

eSeverity_Document = 11

eSeverity_Rest = 12

eSeverity_NoSelect = 13

eSeverity_Select = 14

eSeverity_Autoinsertion = 15

eSeverity_GlobalResources_DefaultWarning = 16

eSeverity_XPath_Styles_Changed = 17

eSeverity_XPath_Styles_Unchanged = 18

eSeverity_XPath_Styles_Skipped = 19

eSeverity_XPath_ComboBox_Values_Changed = 20

eSeverity_XPath_ComboBox_Values_Unchanged = 21

eSeverity_XPath_ComboBox_Values_Skipped = 22

eSeverity_XPath_Assertions_Changed = 23

eSeverity_XPath_Assertions_Unchanged = 24

eSeverity_XPath_Assertions_Skipped = 25

## 17.7.2.4 ENUMAppOutputLine_TextDecoration

Valeurs d'énumération de différents types de décoration de texte d'un `AppOutputLine`.

### Membres

eTextDecorationDefault = 0

eTextDecorationBold = 1

eTextDecorationDebugValues = 2

eTextDecorationDB_ObjectName = 3

eTextDecorationDB_ObjectLink = 4

eTextDecorationDB_ObjectKind = 5

eTextDecorationDB_TimeoutValue = 6

eTextDecorationFind_MatchingString = 7

eTextDecorationValidation_Speclink = 8

eTextDecorationValidation_ErrorPosition = 9

eTextDecorationValidation_UnkownParam = 10

## 17.7.2.5 ENUMCodeGenErrorLevel

Valeurs d'énumération pour identifier la gravité des messages de génération de code.

### Membres

eCodeGenErrorLevel_Information = 0

eCodeGenErrorLevel_Warning = 1

eCodeGenErrorLevel_Error = 2

eCodeGenErrorLevel_Undefined = 3

## 17.7.2.6 ENUMComponentDatapointSide

Valeurs d'énumération pour indiquer le côté d'un datapoint sur son composant. Voir également `Component.GetRootDatapoint`.

### Membres

eDatapointSideInput = 0

eDatapointSideOutput = 1

## 17.7.2.7 ENUMComponentSubType

Valeurs d'énumération pour indiquer les sous-types de composant.

### Membres

eComponentSubType_None = 0

eComponentSubType_Text_EDI = 1

eComponentSubType_Text_Flex = 2

eComponentSubType_Text_CSVFLF = 3

## 17.7.2.8 ENUMComponentType

Valeurs d'énumération pour indiquer les types de composant.

### Membres

eComponentType_Unknown = 0

eComponentType_XML = 1

eComponentType_DB = 2

eComponentType_Text = 3

eComponentType_Excel = 4

eComponentType_WSDL = 5

eComponentType_XBRL = 6

eComponentType_Input = 7

eComponentType_JSON = 8

## 17.7.2.9 ENUMComponentUsageKind

Valeurs d'énumération pour indiquer le type d'utilisation du composant indiqué.

### Membres

eComponentUsageKind_Unknown = 0

eComponentUsageKind_Instance = 1

eComponentUsageKind_Input = 2

eComponentUsageKind_Output = 3

eComponentUsageKind_Variable = 4

eComponentUsageKind_String = 5

## 17.7.2.10 ENUMConnectionType

Valeurs d'énumération pour indiquer le type de connexion. Voir également `Connection.ConnectionType`.

### Membres

eConnectionTypeTargetDriven = 0

eConnectionTypeSourceDriven = 1

eConnectionTypeCopyAll = 2

## 17.7.2.11 ENUMDOMType

Valeurs d'énumération pour spécifier le type de DOM utilisé par le code de mappage C++ généré.

NOTE : La valeur `eDOMType_xerces` est obsolète. `eDOMType_xerces3` indique l'utilisation Xerces 3.x. Obsolète dans ce contexte signifie que cette valeur n'est pas prise en charge et ne devrait pas être utilisée.

### Membres

eDOMType_xerces = 1 (obsolete)

eDOMType_xerces3= 2

eDOMType_msxml6= 3

## 17.7.2.12 ENUMLibType

Valeurs d'énumération pour spécifier le type de bibliothèque utilisé par le code de mappage C++.

### Membres

eLibType_static = 0

eLibType_dll = 1

## 17.7.2.13 ENUMProgrammingLanguage

Enumeration values to select a programming language.

### Members

eUndefinedLanguage = -1

eJava = 0

eCpp = 1

eCSharp = 2

eXSLT = 3

eXSLT2 = 4

eXQuery = 5

## 17.7.2.14 ENUMProjectItemType

Énumération pour identifier les différents types d'items de projet qui peuvent être des enfants de `Project` ou du dossier similaire `ProjectItems`. Voir également `ProjectItem.Kind`.

### Membres

eProjectItemType_MappingFolder = 0

eProjectItemType_Mapping = 1

eProjectItemType_WebServiceFolder = 2

eProjectItemType_WebServiceRoot = 3

eProjectItemType_WebServiceService = 4

eProjectItemType_WebServicePort = 5

eProjectItemType_WebServiceOperation = 6

eProjectItemType_ExternalFolder = 7

eProjectItemType_LibraryFolder = 8

eProjectItemType_ResourceFolder = 9

eProjectItemType_VirtualFolder = 10

eProjectItemType_Count = 11

eProjectItemType_Invalid = -1

## 17.7.2.15 ENUMProjectType

Enumeration values to select a project type for generated C# and C++ mapping code.

### Members

eVisualStudio2010Project = 6

eVisualStudio2013Project = 7

eVisualStudio2015Project = 8

eVisualStudio2017Project = 9

eVisualStudio2019Project = 10

## 17.7.2.16 ENUMSearchDatapointFlags

Valeurs d'énumération utilisées comme bits indicateurs ; à être utilisé comme combinaison d'indicateurs lors de la recherche d'un datapoint. Voir aussi
`GetChild`.

### Membres

eSearchDatapointElement = 1

eSearchDatapointAttribute = 2

## 17.7.2.17 ENUMViewMode

Valeurs d'énumération pour sélectionner une vue MapForce.

## Membres

eMapForceView = 0

eXSLView = 1

eOutputView = 2

# 18 ActiveX Integration

The MapForce user interface and the functionality described in this section can be integrated into custom applications that can consume ActiveX controls. ActiveX technology enables a wide variety of languages to be used for integration, such as C++, C#, VB.NET, HTML. (Note that ActiveX components integrated in HTML must be run with Microsoft Internet Explorer versions and platforms that support ActiveX). All components are full OLE Controls. Integration into Java is provided through wrapper classes.

To integrate the ActiveX controls into your custom code, the MapForce Integration Package must be installed (see https://www.altova.com/components/download). Ensure that you install MapForce first, and then the MapForce Integration Package. Other prerequisites apply, depending on language and platform (see Prerequisites 1292).

You can flexibly choose between two different levels of integration: application level and document level.

Integration at application level means embedding the complete interface of MapForce (including its menus, toolbars, panes, etc) as an ActiveX control into your custom application. For example, in the most simple scenario, your custom application could consist of only one form that embeds the MapForce graphical user interface. This approach is easier to implement than integration at document level but may not be suitable if you need flexibility to configure the MapForce graphical user interface according to your custom requirements.

Integration at document level means embedding MapForce into your own application piece-by-piece. This includes implementing not only the main MapForce control but also the main document editor window, and, optionally, any additional windows. This approach provides greater flexibility to configure the GUI, but requires advanced interaction with ActiveX controls in your language of choice.

The sections Integration at the Application Level 1296 and Integration at Document Level 1299 describe the key steps at these respective levels. The ActiveX Integration Examples 1303 section provides examples in C#, HTML, and Java. Looking through these examples will help you to make the right decisions quickly. The Object Reference 1339 section describes all COM objects that can be used for integration, together with their properties and methods.

For information about using MapForce as a Visual Studio plug-in, see MapForce in Visual Studio 911.

# 18.1 Prerequisites

To integrate the MapForce ActiveX control into a custom application, the following must be installed on your computer:

- MapForce
- The MapForce Integration Package, available for download at https://www.altova.com/components/download

To integrate the 64-bit ActiveX control, install the 64-bit versions of MapForce and MapForce Integration Package. For applications developed under Microsoft .NET platform with Visual Studio, both the 32-bit and 64-bit versions of MapForce and MapForce Integration Package must be installed, as explained below.

## Microsoft .NET (C#, VB.NET) with Visual Studio

To integrate the MapForce ActiveX control into a 32-bit application developed under Microsoft .NET, the following must be installed on your computer:

- Microsoft .NET Framework 4.0 or later
- Visual Studio 2012/2013/2015/2017/2019/2022
- MapForce 32-bit and MapForce Integration Package 32-bit
- The ActiveX controls must be added to the Visual Studio toolbox (see Adding the ActiveX Controls to the Toolbox 1294).

If you want to integrate the 64-bit ActiveX control, the following prerequisites apply in addition to the ones above:

- MapForce 32-bit and MapForce Integration Package 32-bit must still be installed (this is required to provide the 32-bit ActiveX control to the Visual Studio designer, since Visual Studio runs on 32-bit)
- MapForce 64-bit and MapForce Integration Package 64-bit must be installed (provides the actual 64-bit ActiveX control to your custom application at runtime)
- In Visual Studio, create a 64-bit build configuration and build your application using this configuration. For an example, see Running the Sample C# Solution 1303.

## Java

To integrate the MapForce ActiveX control into Java application using the Eclipse development environment, the following must be installed on your computer:

- Java Runtime Environment (JRE) or Java Development Kit (JDK) 7 or later
- Eclipse
- MapForce and MapForce Integration Package

**Note:** To run the 64-bit version of the MapForce ActiveX control, use a 64-bit version of Eclipse, as well as the 64-bit version of MapForce and the MapForce Integration Package.

## MapForce integration and deployment on client computers

If you create a .NET application and intend to distribute it to other clients, you will need to install the following on the client computer(s):

- MapForce
- The MapForce Integration Package
- The custom integration code or application.

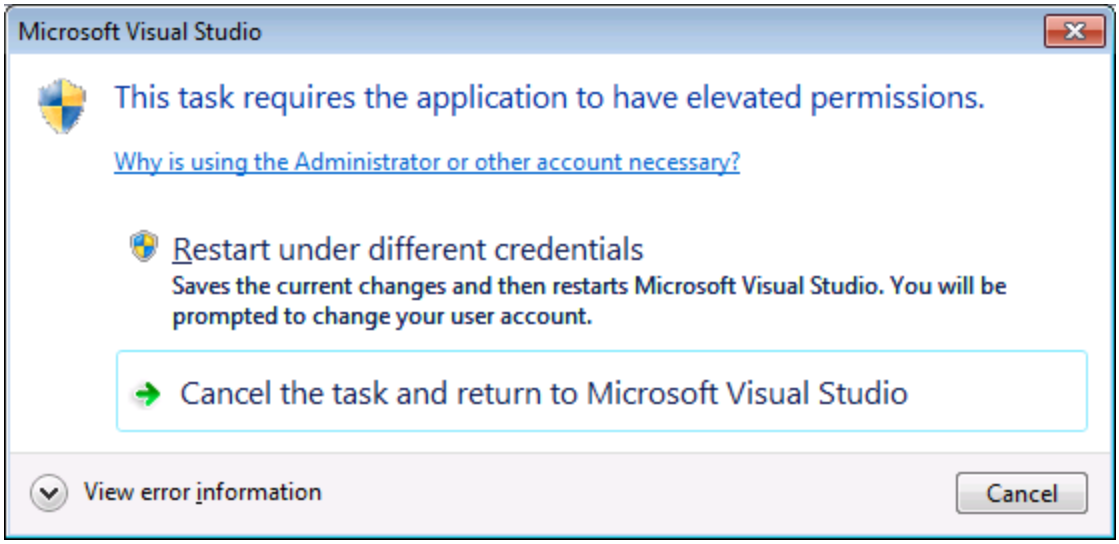
## 18.2 Adding the ActiveX Controls to the Toolbox

To use the MapForce ActiveX controls in an application developed with Visual Studio, the controls must first be added to the Visual Studio Toolbox, as follows:

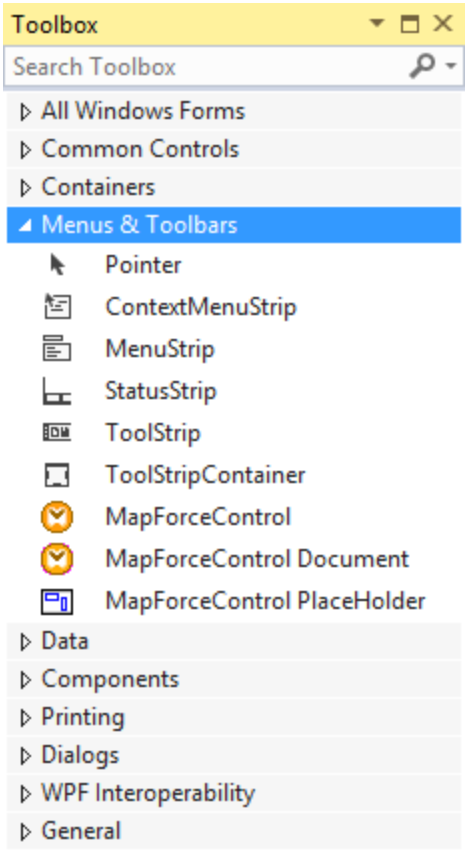
1. On the **Tools** menu of Visual Studio, click **Choose Toolbox Items**.
2. On the **COM Components** tab, select the check boxes next to the MapForceControl, MapForceControl Document, and MapForceControl Placeholder.

In case the controls above are not available, follow the steps below:

1. On the **COM Components** tab, click **Browse**, and select the **MapForceControl.ocx** file from the MapForce installation folder. Remember that the MapForce Integration Package must be installed; otherwise, this file is not available, see Prerequisites[1292].
2. If prompted to restart Visual Studio with elevated permissions, click **Restart under different credentials**.




If the steps above were successful, the MapForce ActiveX controls become available in the Visual Studio Toolbox.




**Note:** For an application-level integration, only the **MapForceControl** ActiveX control is used (see Integration at Application Level 1296). The **MapForceControl Document** and **MapForceControl Placeholder** controls are used for document-level integration (see Integration at Document Level 1299).

## 18.3 Integration at Application Level

Integration at application level allows you to embed the complete interface of MapForce into a window of your application. With this type of integration, you get the whole user interface of MapForce, including all menus, toolbars, the status bar, document windows, and helper windows. Customization of the application's user interface is restricted to what MapForce provides. This includes rearrangement and resizing of helper windows and customization of menus and toolbars.

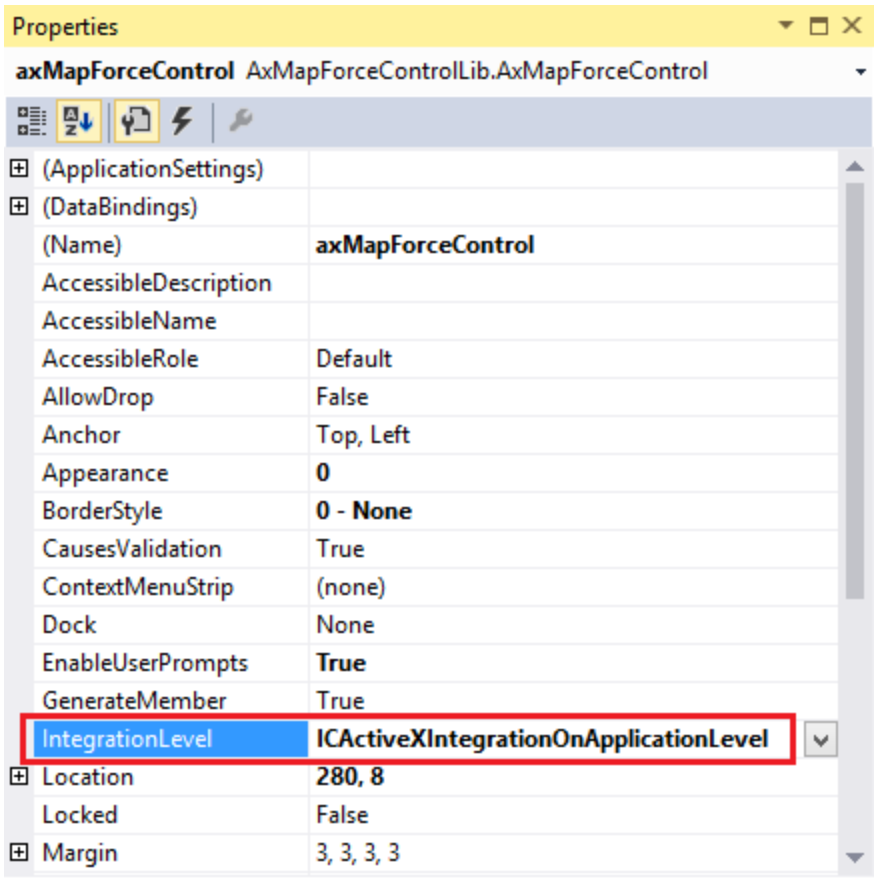
The only ActiveX control you need to integrate is `MapForceControl`[1342]. Do not instantiate or access `MapForceControlDocument`[1350] or `MapForceControlPlaceHolder`[1356] ActiveX controls when integrating at application-level.

If you have any initialization to do or if you want to automate some behaviour of MapForce, use the properties, methods, and events described for `MapForceControl`[1342]. Consider using `MapForceControl.Application`[1343] for more complex access to MapForce functionality.

For an example that shows how the MapForce application can be embedded in an HTML page, see HTML Integration at Application Level[1310].

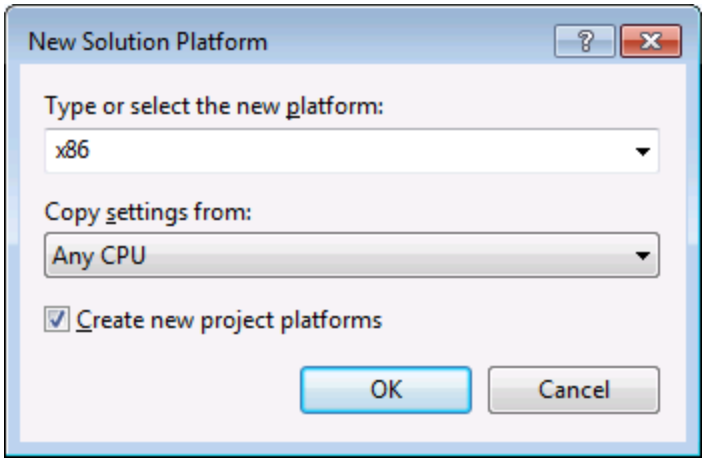
In C# or VB.NET with Visual Studio, the steps to create a basic, one-form application which integrates the MapForce ActiveX controls at application level are as follows:

1. Check that all prerequisites are met (see Prerequisites[1292]).
2. Create a new Visual Studio Windows Forms project with a new empty form.
3. If you have not done that already, add the ActiveX controls to the toolbox (see Adding the ActiveX Controls to the Toolbox[1294]).
4. Drag the **MapForceControl** from the toolbox onto your new form.
5. Select the **MapForceControl** on the form, and, in the Properties window, set the **IntegrationLevel** property to **ICActiveXIntegrationOnApplicationLevel**.




6. Create a build platform configuration that matches the platform under which you want to build (x86, x64). Here is how you can create the build configuration:

   a. Right-click the solution in Visual Studio, and select **Configuration Manager**.
   b. Under **Active solution platform**, select **New...** and then select the x86 or x64 configuration (in this example, **x86**).



You are now ready to build and run the solution in Visual Studio. Remember to build using the configuration that matches your target platform (x86, x64).

# 18.4 Integration at Document Level

Compared to integration at application level, integration at document level is a more complex, yet more flexible way to embed MapForce functionality into your application by means of ActiveX controls. With this approach, your code can access selectively the following parts of the MapForce user interface:

- Document editing window
- Project window
- Libraries window
- Overview window
- Messages window

As mentioned in Integration at Application Level[1296], for an ActiveX integration at application level, only one control is required, namely the **MapForceControl**. However, for an ActiveX integration at document level, MapForce functionality is provided by the following ActiveX controls:

- MapForceControl[1342]
- MapForceControl Document[1350]
- MapForceControl Placeholder[1356]

These controls are supplied by the **MapForceControl.ocx** file available in the application installation folder of MapForce. When you develop the ActiveX integration with Visual Studio, you will need to add these controls to the Visual Studio toolbox (see Adding the ActiveX Controls to the Toolbox[1294]).

The basic steps to integrate the ActiveX controls at document level into your application are as follows:

1. First, instantiate **MapForceControl** in your application. Instantiating this control is mandatory; it enables support for the **MapForceControl Document** and **MapForceControl Placeholder** controls mentioned above. It is important to set the `IntegrationLevel`[1344] property to **ICActiveXIntegrationOnDocumentLevel** (or "1"). To hide the control from the user, set its **Visible** property to **False**.

**Note:** When integrating at document level, do not use the **Open** method of the **MapForceControl**; this might lead to unexpected results. Use the corresponding open methods of **MapForceControl Document** and **MapForceControl PlaceHolder** instead.

2. Create at least one instance of **MapForceControl Document** in your application. This control supplies the document editing window of MapForce to your application and can be instantiated multiple times if necessary.

   Use the method **Open** to load any existing file. To access document-related functionality, use the **Path** and **Save** or methods and properties accessible via the property **Document**.

**Note:** The control does not support a read-only mode. The value of the property **ReadOnly** is ignored.

3. Optionally, add to your application the **MapForceControl Placeholder** control for each additional window (other than the document window) that must be available to your application.

   Instances of **MapForceControl PlaceHolder** allow you to selectively embed additional windows of MapForce into your application. The window kind (for example, Project window) is defined by the

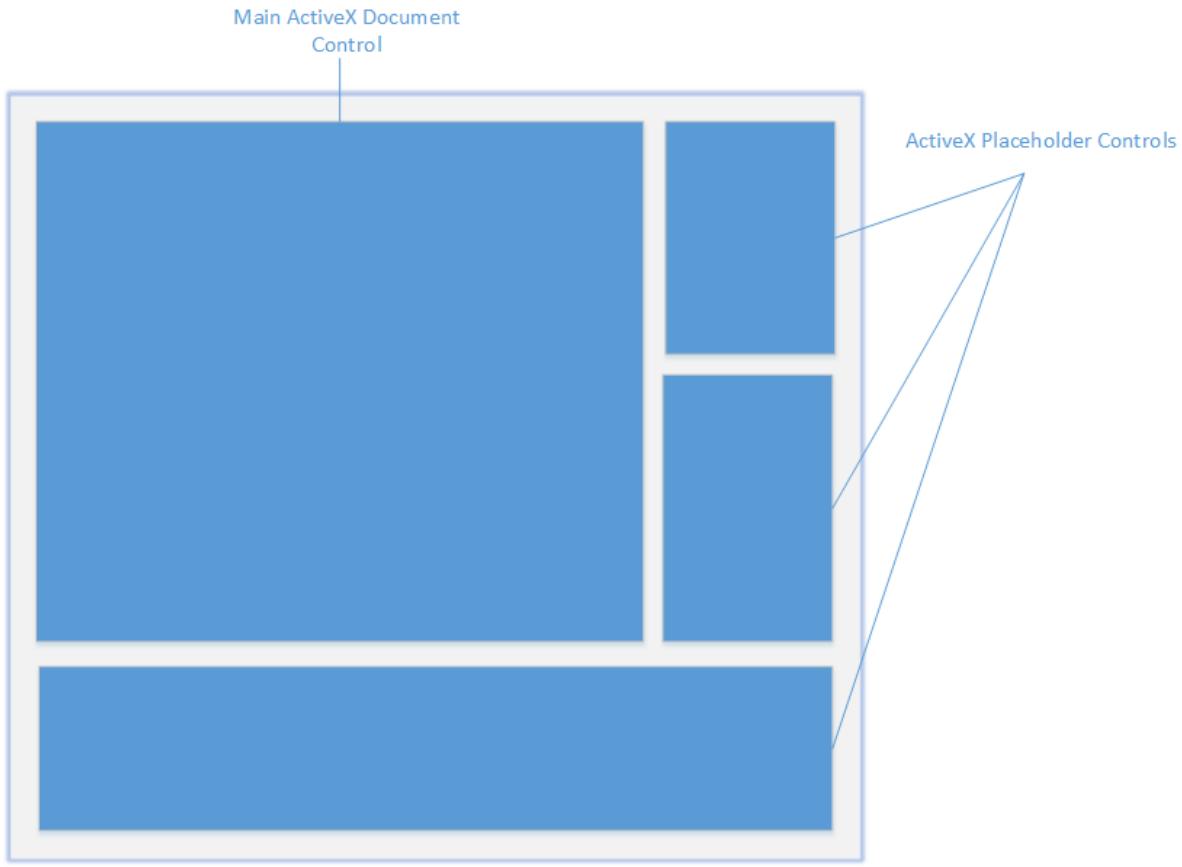
property **PlaceholderWindowID**. Therefore, to set the window kind, set the property **PlaceholderWindowID**. For valid window identifiers, see MapForceControlPlaceholderWindow[1359].

**Note:** Use only one **MapForceControl PlaceHolder** for each window identifier.

For placeholder controls that select the MapForce project window, additional methods are available. Use **OpenProject** to load a MapForce project. Use the property Project and the methods and properties from the MapForce automation interface to perform any other project related operations.

For example, in C# or VB.NET with Visual Studio, the steps to create a basic, one-form application which integrates the MapForce ActiveX controls at document level could be similar to those listed below. Note that your application may be more complex if necessary; however, the instructions below are important to understand the minimum requirements for an ActiveX integration at document level.

1. Create a new Visual Studio Windows Forms project with a new empty form.
2. If you have not done that already, add the ActiveX controls to the toolbox (see Adding the ActiveX Controls to the Toolbox[1294]).
3. Drag the **MapForceControl**[1342] from the toolbox onto your new form.
4. Set the **IntegrationLevel** property of the **MapForceControl** to **ICActiveXIntegrationOnDocumentLevel**, and the **Visible** property to **False**. You can do this either from code or from the **Properties** window.
5. Drag the **MapForceControl Document**[1350] from the toolbox onto the form. This control provides the main document window of MapForce to your application, so you may need to resize it to a reasonable size for a document.
6. Optionally, add one or more **MapForceControl Placeholder**[1356] controls to the form (one for each additional window type that your application needs, for example, the **Project** window). You will typically want to place such additional placeholder controls either below or to the right or left of the main document control, for example:




7. Set the **PlaceholderWindowID** property of each **MapForceControl Placeholder** control to a valid window identifier. For the list of valid values, see MapForceControlPlaceholderWindow[1359].
8. Add commands to your application (at minimum, you will need to open, save and close documents), as shown below.

## Querying MapForce Commands

When you integrate at document level, no MapForce menu or toolbar is available to your application. Instead, you can retrieve the required commands, view their status, and execute them programmatically, as follows:

- To retrieve all available commands, use the `CommandsList`[1343] property of the **MapForceControl**.
- To retrieve commands organized according to their menu structure, use the `MainMenu`[1344] property.
- To retrieve commands organized by the toolbar in which they appear, use the `Toolbars`[1345] property.
- To send commands to MapForce, use the `Exec`[1346] method.
- To query if a command is currently enabled or disabled, use the `QueryStatus`[1347] method.

This enables you to flexibly integrate MapForce commands into your application's menus and toolbars.

Your installation of MapForce also provides you with command label images used within MapForce. See the folder **<ApplicationFolder>\Examples\ActiveX\Images** of your MapForce installation for icons in GIF format. The file names correspond to the command names as they are listed in the Command Reference[1330] section.

## General considerations

To automate the behaviour of MapForce, use the properties, methods, and events described for the **MapForceControl**[1342], **MapForceControl Document**[1350], and **MapForceControl Placeholder**[1356].

For more complex access to MapForce functionality, consider using the following properties:

- `MapForceControl.Application`[1343]
- `MapForceControlDocument.Document`[1351]
- `MapForceControlPlaceHolder.Project`[1357]

These properties give you access to the MapForce automation interface (MapForceAPI)

**Note:** To open a document, always use `MapForceControlDocument.Open`[1353] or `MapForceControlDocument.New`[1352] on the appropriate document control. To open a project, always use `MapForceControlPlaceHolder.OpenProject`[1357] on a placeholder control embedding a MapForce project window.

For examples that show how to instantiate and access the necessary controls in different programming environments, see ActiveX Integration Examples[1303].

# 18.5 ActiveX Integration Examples

This section contains examples of MapForce document-level integration using different container environments and programming languages. Source code for all examples is available in the folder `<ApplicationFolder>\Examples\ActiveX` of your MapForce installation.

## 18.5.1 C#

A basic ActiveX integration example solution for C# and Visual Studio is available in the folder `<ApplicationFolder>\Examples\ActiveX\C#`. Before you compile the source code and run the sample, make sure that all prerequisites are met (see Running the Sample C# Solution 1303).

### 18.5.1.1 Running the Sample C# Solution

The sample Visual Studio solution available in the folder **<ApplicationFolder>\Examples\ActiveX\C#** illustrates how to consume the MapForce ActiveX controls. Before attempting to build and run this solution, note the following steps:

#### Step 1: Check the prerequisites

Visual Studio 2010 or later is required to open the sample solution. For the complete list of prerequisites, see Prerequisites 1292.

#### Step 2: Copy the sample to a directory where you have write permissions

To avoid running Visual Studio as an Administrator, copy the source code to a directory where you have write permissions, instead of running it from the default location.

#### Step 3: Check and set all required control properties

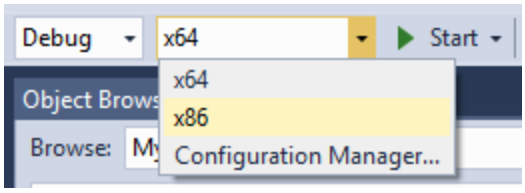
The sample application contains one instance of MapForceControlDocument 1350 and several instances of MapForceControlPlaceHolder 1356 controls. Double-check that the following properties of these controls are set as shown in the table below:

| Control name | Property | Property value |
|---|---|---|
| axMapForceControl | IntegrationLevel | ICActiveXIntegrationOnDocumentLevel |
| axMapForceControlLibrary | PlaceholderWindowID | 0 |
| axMapForceControlOutput | PlaceholderWindowID | 2 |
| axMapForceControlPreview | PlaceholderWindowID | 1 |

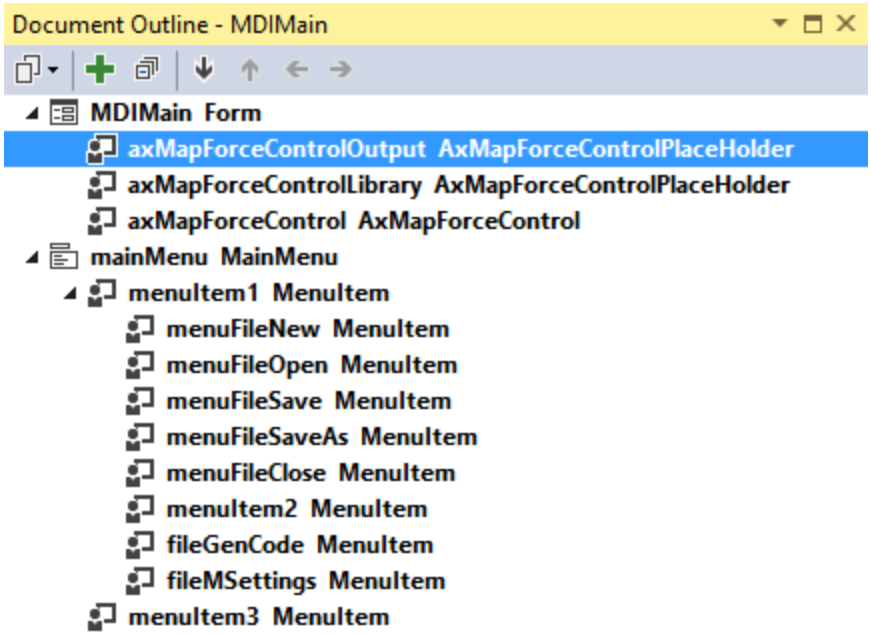
Here is how you can view or set the properties of an ActiveX control:

1. Open the **MDIMain.cs** form in the designer window.

**Note:** On 64-bit Windows, it may be necessary to change the build configuration of the Visual Studio solution to "x86" **before** opening the designer window. If you need to build the sample as a 64-bit application, see Prerequisites 1292.

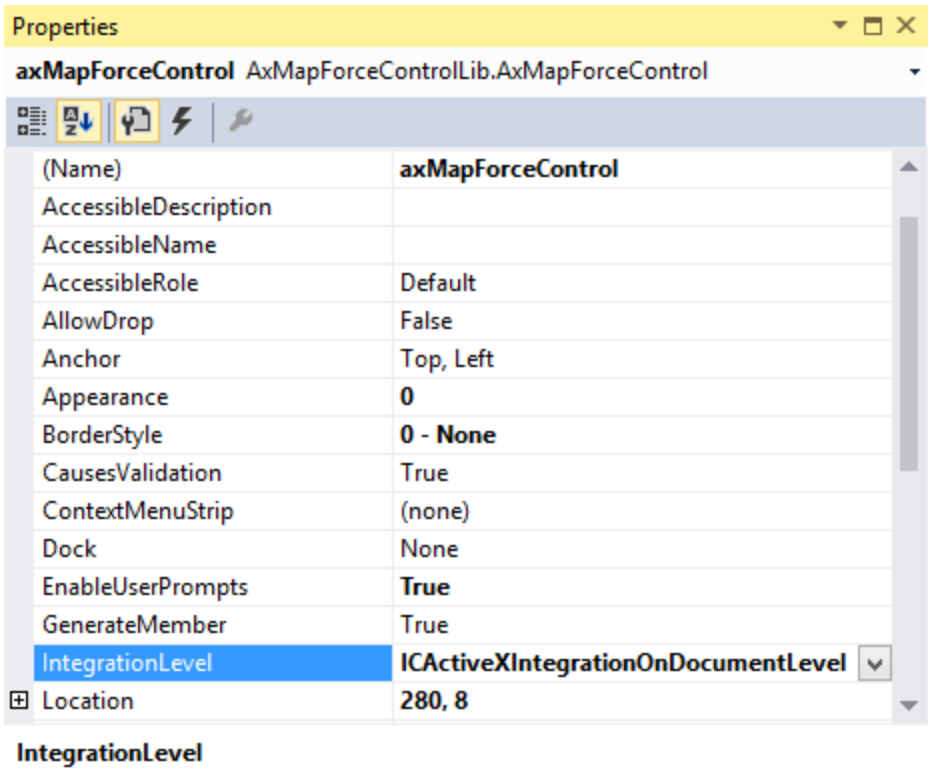



2. Open the **Document Outline** window of Visual Studio (On the **View** menu, click **Other Windows | Document Outline**).




3. Click an ActiveX control in the **Document Outline** window, and edit its required property in the **Properties** window, for example:

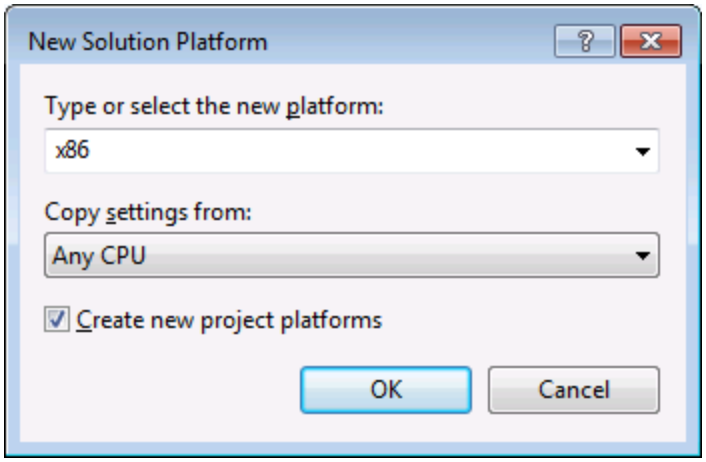
 




## Step 4: Set the build platform

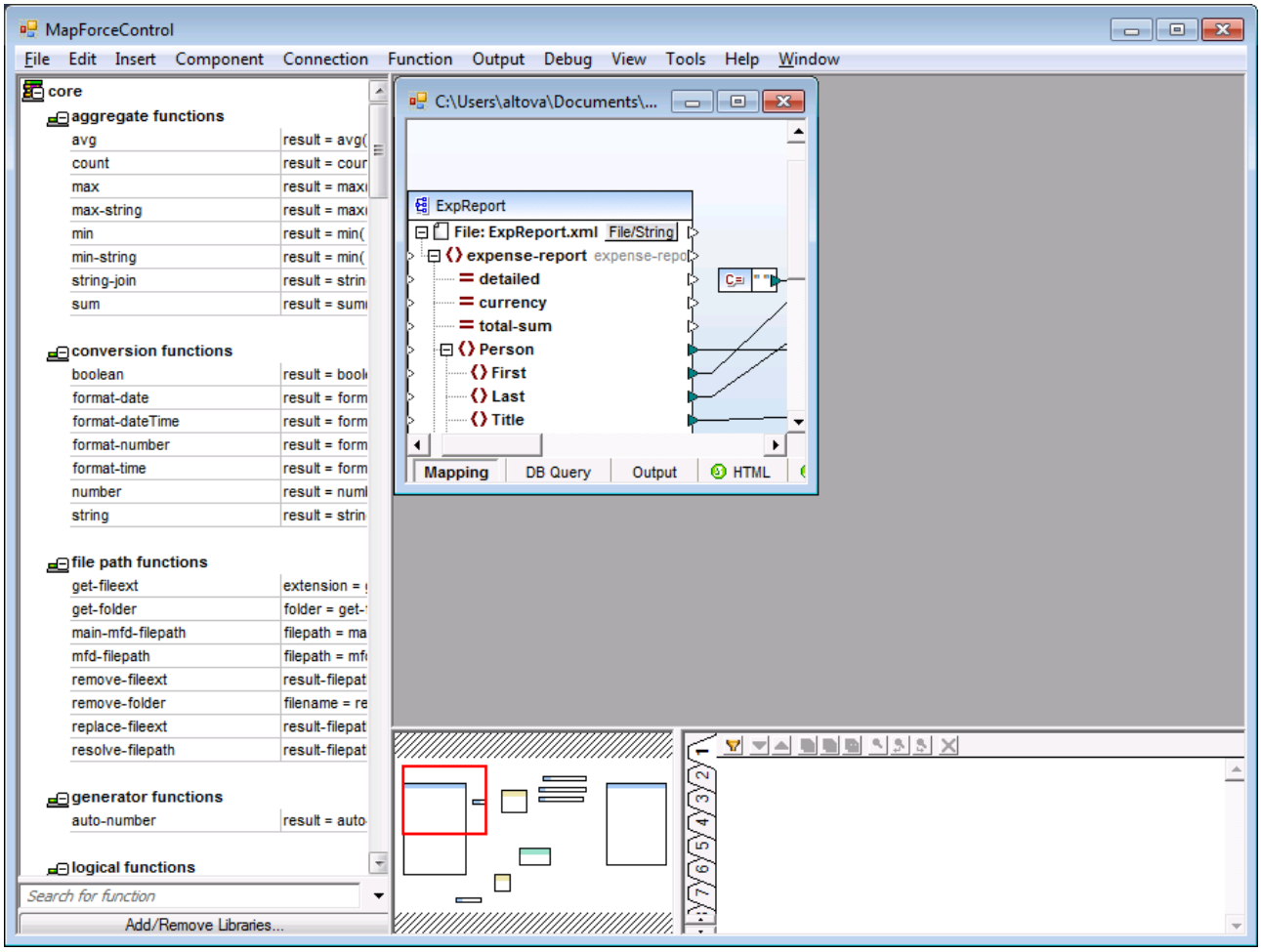
- Create a build platform configuration that matches the platform under which you want to build (x86, x64). Here is how you can create the build configuration:

  a. Right-click the solution in Visual Studio, and select **Configuration Manager**.
  b. Under **Active solution platform**, select **New...** and then select the x86 or x64 configuration (in this example, **x86**).




You are now ready to build and run the solution in Visual Studio. Remember to build using the configuration that matches your target platform (x86, x64); otherwise, runtime errors might occur.

On running the sample, the main MDI Frame window is displayed. Use **File | Open** to open a mapping file (for example, `MarketingExpenses.mfd`, which is in the MapForce examples folder). The file is loaded and displayed in a new document child window:




After you load the document, you can execute commands against the active document using the menu. Context menus are also available. You can also load additional documents. Save any modifications using the **File | Save** command.

## 18.5.1.2 Retrieving Command Information

The MapForceControl gives access to all commands of MapForce through its `CommandsList` , `MainMenu`, and `Toolbars` properties. The example project available in the folder **<ApplicationFolder>\Examples\ActiveX\C#** uses the `MainMenu` property to create the MapForce menu structure dynamically.

The code that gets the menu commands can be found in the `MDIMain` method in `MDIMain.cs` file:

```
public MDIMain()
{
```

```
    // ...

    // Get the MainMenu property of the control and create the menu structure from it.
    MFLib.MapForceCommand objCommand = this.axMapForceControl.MainMenu;
    InsertMenuStructure(mainMenu, objCommand);
}
```

In the code listing above, `mainMenu` is the existing static menu of the main MDI Frame window. If you open the **MDIMain.cs** form in the Visual Studio Designer, you will notice that this menu contains two menu items: **File** and **Window**.


*MDIMain.cs*

The method `InsertMenuStructure` takes as parameters the `mainMenu` and the `objCommand` objects (the former is the existing static menu, while the latter contains the full menu structure retrieved from the MapForce ActiveX control). The retrieved MapForce menu structure is then merged into the existing static menu. Note that the menus **File**, **Project**, and **Window** are not added dynamically. This is intentional, because these menus deal with actively open documents, and they would require code which is beyond the scope of this example. The basic file management commands (create, open, save, bring into focus) are handled by the existing static menus **File** and **Window**. All other menus are inserted dynamically based on the information taken from the `MainMenu` property of the ActiveX control. The new menus are inserted after "File" but before "Window", i.e. starting at menu index 1.

The method `InsertMenuStructure` iterates through all top-level menus found in `MapForceCommand` object and adds a new menu item for each. Since each top-level menu has its own child menu items, a call to the method `InsertMenuCommand` takes place for each encountered child menu item. Furthermore, since each child menu item can have its own children menu items, and so on, the `InsertMenuCommand` method recurses into itself until no more child menu items exist.

The commands added dynamically are instances of the class `CustomMenuItem`, which is defined in `CustomMenuItem.cs`. This class is derived from `System.Windows.Forms.MenuItem` class and has an additional member to store the MapForce command ID.

```
public class CustomMenuItem : System.Windows.Forms.MenuItem
{
   public int m_MapForceCmdID;
}
```

All dynamically added commands (except those that are containers for other commands) get the same event handler `AltovaMenuItem_Click` which does the processing of the command:

```
private void AltovaMenuItem_Click(object sender, EventArgs e)
{
  if(sender.GetType() == System.Type.GetType("MapForceApplication.CustomMenuItem"))
  {
    CustomMenuItem   customItem = (CustomMenuItem)sender;
    ProcessCommand(customItem.m_MapForceCmdID);
  }
}
```

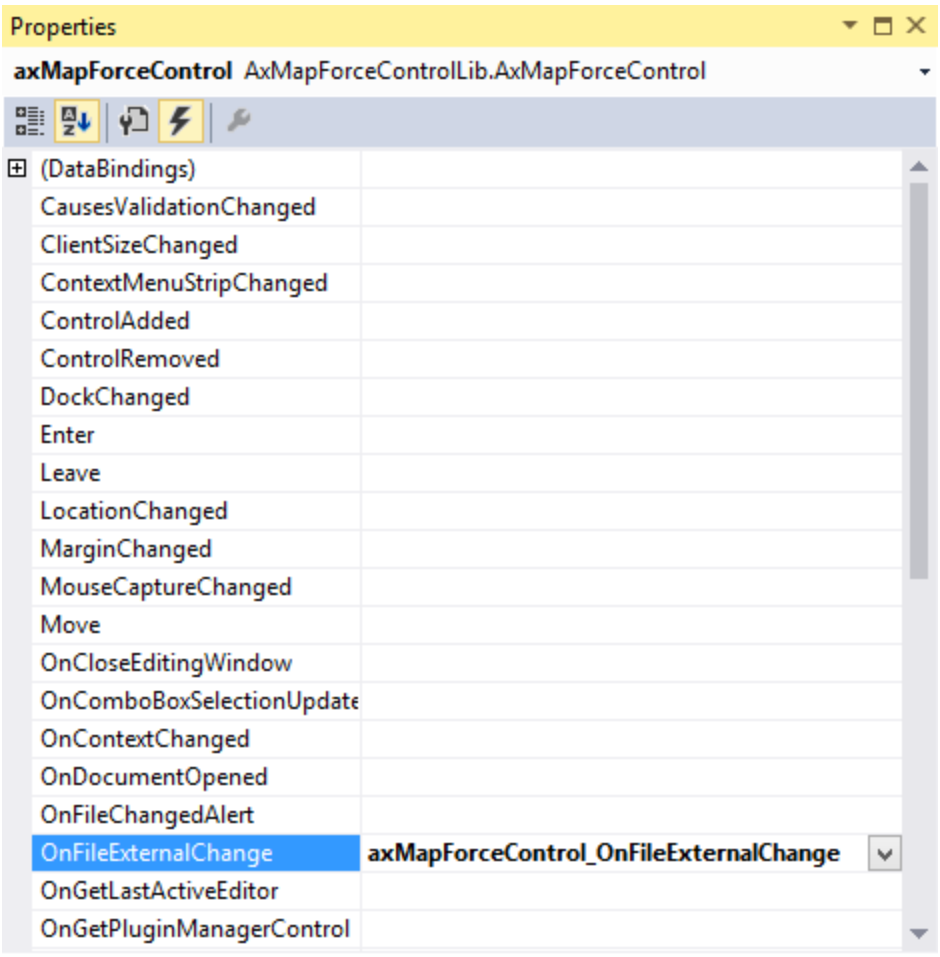
If the command is a container for other commands (that is, if it has child commands), it gets the event handler `AltovaSubMenu_Popup`. This handler queries the status of each child command and enables or disables it as required. This ensures that each command is enabled only when that is meaningful (for example, the **File | Save** menu item should be disabled if there is no active document open).

The method `ProcessCommand` delegates the execution either to the MapForceControl itself or to any active MapForce document loaded in a `MapForceControlDocument` control. This is necessary because the MapForceControl has no way to know which document is currently active in the hosting application.

```
private void ProcessCommand(int nID)
{
   MapForceDoc docMapForce = GetCurrentMapForceDoc();

   if(docMapForce != null)
      docMapForce.axMapForceControlDoc.Exec(nID);
   else
      axMapForceControl.Exec(nID);
}
```

### 18.5.1.3 Handling Events

Because all events in the MapForce library are based on connection points, you can use the C# delegate mechanism to provide the custom event handlers. You will always find a complete list of events on the property page of each control of the MapForce library. The image below shows the events of the main MapForceControl:




As you can see, the example project only overrides the `OnFileExternalChange` event. The creation of the C# delegate is done for you by the C# Framework. All you need to do is fill in the empty event handler.

For example, the handler implementation shown below turns off any file reloading and displays a message box to inform the user that a file loaded by the MapForceControl has been changed from outside:

```
private void axMapForceControl_OnFileExternalChange(object sender,
AxMapForceControlLib._DMapForceControlEvents_OnFileExternalChangeEvent e)
{
    MessageBox.Show("Attention: The file " + e.strPath + " has been changed from
outside\nbut reloading is turned off in the sample application!");

    // This turns off any file reloading:
    e.varRet = false;
}
```

## 18.5.2 HTML

The code listings in this section show how to integrate the MapForceControl at application level and document level. Source code for all examples is available in the folder `<ApplicationFolder>\Examples\ActiveX\HTML` of your MapForce installation.

**Note:** ActiveX controls in an HTML page are supported only by Internet Explorer when it runs as a 32-bit application. When Internet Explorer 10 or 11 runs in 64-bit mode, it does not load ActiveX controls. The default browser security settings will normally block ActiveX, so you will need to explicitly allow blocked content to run on the page when prompted by Internet Explorer.

### 18.5.2.1 HTML Integration at Application Level

This example shows a simple integration of the MapForce control at application-level into a HTML page. The integration is described in the following sections:

- Instantiate a MapForceControl in HTML code.
- Implement buttons to load documents and automate code-generation tasks.
- Define actions for some application events.

The code for this example is available at the following location in your MapForce installation: `<ApplicationFolder>\Examples\ActiveX\HTML\MapForceActiveX_ApplicationLevel.htm`.

#### 18.5.2.1.1 Instantiate the Control

The HTML `Object` tag is used to create an instance of the MapForceControl. The `Classid` is that of MapForceControl. Width and height specify the window size. No additional parameters are necessary, since application-level is the default.

```
<OBJECT id="objMapForceControl"
        Classid="clsid:A38637E9-5759-4456-A167-F01160CC22C1"
        width="800"
        height="500"
        VIEWASTEXT>
</OBJECT>
```

#### 18.5.2.1.2 Add Button to Open Default Document

As a simple example of how to automate some tasks, we add a button to the page:

```
<input type="button" value="Open Marketing Expenses" onclick="BtnOpenMEFile()">
```

When clicked, a predefined document will be opened in the MapForceControl. The `MakeAbsolutePath` method creates an absolute path using the location of the script as a base path.

```
function BtnOpenMEFile()
{
   var strPath = MakeAbsolutePath("MarketingExpenses.mfd");
   var objDoc = objMapForceControl.Open(strPath );

   if (objDoc == null)
      alert("Unable to locate MarketingExpenses.mfd at: " + objMapForceControl.BaseHref);
}
```

### 18.5.2.1.3 Add Buttons for Code Generation

Although code-generation for the active document is available via menus, we want to have buttons that will generate code without asking the user for the location of the output. The method is similar to that used in the previous section.

First come the buttons:

```
<input type="button" value="Generate XSLT" onclick="BtnGenerate( 0 )">
<input type="button" value="Generate Java" onclick="BtnGenerate( 1 )">
<input type="button" value="Generate C++" onclick="BtnGenerate( 2 )">
<input type="button" value="Generate C#" onclick="BtnGenerate( 3 )">
```

Then we provide the script that will generate the code into sub-folders of the currently defined default output folders.

```
// ------------------------------------------------------------------------
// generate code for active document into language-specific subfolders of
// the current default output dicrectory. No user intercation necessary.
function BtnGenerate(languageID)
{
   // get top-level object of automation interface
   var objApp = objMapForceControl.Application;

   // get the active document
   var objDocument = objApp.ActiveDocument;

   // retrive object to set the generation output path
   var objOptions = objApp.Options;

   if ( objDocument == null )
      alert( "no active document found" );
   else
   {
      objOptions.XSLTDefaultOutputDirectory = objOptions.CodeDefaultOutputDirectory =
GetDefaultOutputDirectory();

      if (languageID == 0)
      {
         objOptions.XSLTDefaultOutputDirectory = objOptions.XSLTDefaultOutputDirectory +
"\\XSLTGen";
```

```
         objDocument .GenerateXSLT();
      }
      else if (languageID == 1)
      {
         objOptions.CodeDefaultOutputDirectory = objOptions.CodeDefaultOutputDirectory +
"/JavaCode";
         objDocument .GenerateJavaCode();
      }
      else if (languageID == 2)
      {
         objOptions.CodeDefaultOutputDirectory = objOptions.CodeDefaultOutputDirectory +
"/CPPCode";
         objDocument .GenerateCppCode();
      }
      else if (languageID == 3)
      {
         objOptions.CodeDefaultOutputDirectory = objOptions.CodeDefaultOutputDirectory +
"/CSharpCode";
         objDocument .GenerateCHashCode();
      }
   }
}
```

#### 18.5.2.1.4 Connect to Custom Events

The example implements two event callbacks for MapForceControl custom events to show the principle:

```
<!-- ------------------------------------------------------------ -->
<!--  custom event 'OnDocumentOpened" of MapForceControl object  -->
<SCRIPT FOR="objMapForceControl" event="OnDocumentOpened( objDocument )"
LANGUAGE="javascript">
   // alert("Document '" + objDocument.Name + "' opend!");
</SCRIPT>

<!-- ------------------------------------------------------------ -->
<!--  custom event 'OnDocumentClosed" of MapForceControl object  -->
<SCRIPT FOR="objMapForceControl" event="OnDocumentClosed( objDocument )"
LANGUAGE="javascript">
   // alert("Document '" + objDocument.Name + "' closed!");
</SCRIPT>
```

### 18.5.2.2 HTML Integration at Document Level

This example shows an integration of the MapForce control at document-level into a HTML page. The following topics are covered:

- Instantiate a MapForceControl ActiveX control object in HTML code
- Instantiate a MapForceControlDocument ActiveX control to allow editing a MapForce file
- Instantiate one MapForceControlPlaceHolder for a MapForceControl project window
- Instantiate one MapForceControlPlaceHolder to alternatively host one of the MapForce helper windows

- Create a simple custom toolbar for some heavy-used MapForce commands
- Add some more buttons that use the COM automation interface of MapForce
- Use event handlers to update command buttons

This example is available in its entirety in the file `MapForceActiveX_DocumentLevel.htm` within the `<ApplicationFolder>\Examples\ActiveX\HTML\` folder of your MapForce installation.

#### 18.5.2.2.1 Instantiate the MapForceControl

The HTML `OBJECT` tag is used to create an instance of the MapForceControl. The Classid is that of MapForceControl. Width and height are set to 0 since we use this control as manager control without use for its user interface. The integration level is specified as a parameter within the `OBJECT` tag.

```
<object id="objMapForceX" classid="clsid:A38637E9-5759-4456-A167-F01160CC22C1" width="0"
height="0" VIEWASTEXT>
   <param name="IntegrationLevel" value="1">
</object>
```

#### 18.5.2.2.2 Create Editor Window

The HTML `OBJECT` tag is used to embed an editing window. The additional custom parameter specifies that the control is to be initialized with a new empty document.

```
<object id="objDoc1" classid="clsid:DFBB0871-DAFE-4502-BB66-08CEB7DF5255" width="600"
height="500" VIEWASTEXT>
   <param name="NewDocument">
</object>
```

#### 18.5.2.2.3 Create Project Window

The HTML `OBJECT` tag is used to create a MapForceControlPlaceHolder window. The parameter defines the placeholder to show the MapForce project window.

```
<!-- ------------------------------------------------- -->
<!-- create project window placeholder control.        -->
<!-- initialize it with a project.                     -->
<object id="objProjectWindow" classid="clsid:FDEC3B04-05F2-427d-988C-F03A85DE53C2"
width="200" height="200" VIEWASTEXT>
   <param name="PlaceholderWindowID" value="3">
</object>
```

#### 18.5.2.2.4 Create Placeholder for Helper Windows

The **MapForceControlPlaceHolder** control is required to host an application helper window, see also Integration at Document Level[1299]. In the code listing below, the HTML `object` tag is used to instantiate a control that will host the **Libraries** window by default (`PlaceholderWindowID` is 0).

```
<!-- create helper window placeholder control.           -->
<!-- the editor with focus will automatically direct its -->
<!-- output to the appropriate helper window.            -->
<object id="objPlaceholderWindow" Classid="clsid:FDEC3B04-05F2-427d-988C-F03A85DE53C2"
width="200" height="200" VIEWASTEXT>
   <param name="PlaceholderWindowID" value="0">
   <param name="FileName" value="">
</object>
```

The example HTML page includes a few buttons which call the `BtnHelperWindow` method when clicked. The `BtnHelperWindow` method reassigns the `PlaceholderWindowID` of the control, and thus cause the ActiveX object to display a different helper window.

```
// specify which of the helper windows shall be shown in the placeholder control.
function BtnHelperWindow(i_ePlaceholderWindowID)
{
   objPlaceholderWindow.PlaceholderWindowID = i_ePlaceholderWindowID;
}
```

For the list of possible values of `PlaceholderWindowID`, see `MapForceControlPlaceholderWindow`[1359].

#### 18.5.2.2.5 Create a Custom Toolbar

The example HTML page also includes a custom toolbar (intended as a replica of the MapForce menu). The custom toolbar consists of buttons with images of MapForce commands, for example:

```
<button id="btnInsertXML" title="Insert XML Schema/File" onclick="BtnDoCommand(32393)">
   <img src="..\Images\ID_INSERT_XSD.gif" width="16" height="16" />
</button>
<button id="btnInsertDB" title="Insert Database" onclick="BtnDoCommand(32389)">
   <img src="..\Images\ID_INSERT_DATABASE.gif" width="16" height="16" />
</button>
<button id="btnInsertEDI" title="Insert EDI" onclick="BtnDoCommand(32390)">
   <img src="..\Images\ID_INSERT_EDI.gif" width="16" height="16" />
</button>
<button id="btnInsertText" title="Insert Text file" onclick="BtnDoCommand(32392)">
   <img src="..\Images\ID_INSERT_TXT.gif" width="16" height="16" />
</button>
```

The names of button images correspond to the command ID numbers, see Command Reference[1330]. On clicking the button, the corresponding command ID is sent to the main control and executed:

```
// perform any command specified by cmdID.
// command routing includes application, active document and view.
function BtnDoCommand(cmdID)
{
   objMapForceX.Exec( cmdID );
   msgtext.innerText = "Command " + cmdID + " performed.";
}
```

### 18.5.2.2.6 Create More Buttons

In the example, we add some more buttons to show some automation code.

```
<!-- add some buttons associated with above editor.               -->
<!-- generation of code is now implemented using the MapForce automation  -->
<!-- interface to select a target folder without prompting the user.      -->
<p>
   <input type="button" value="New File" onclick="BtnNewFile(objDoc1)">
   <input type="button" value="Save File" onclick="BtnSaveFile(objDoc1)">
   <input type="text" title="Path" id="strPath" width="150">
   <input type="button" value="Open MarketingExpenses" onclick="BtnOpenMEFile(objDoc1)">
   <input type="button" value="Open MapForce Sample Project"
onclick="BtnOpenProjectFile(objDoc1)">
</p>
```

The corresponding JavaScript looks like this:

```
// ---------------------------------------------------------------------
// open a new empty document in the specified document control window.
function BtnNewFile(objDocCtrl)
{
   objDocCtrl.OpenDocument("");
   objDocCtrl.setActive();
}

// ---------------------------------------------------------------------
// Saves the current file in the specified document control window.
function BtnSaveFile(objDocCtrl)
{
   if(objDocCtrl.Path.length > 0)
objDocCtrl.SaveDocument();
   else
   {
if(strPath.value.length > 0)
{
   objDocCtrl.Path = strPath.value;
   objDocCtrl.SaveDocument();
}
else
{
   alert("Please set path for the document first!");
```

```
   strPath.focus();
}
   }

   objDocCtrl.setActive();
}

// ------------------------------------------------------
// open a document in the specified document control window.
function BtnOpenMEFile(objDocCtrl)
{
   // do not use MapForceX.Application.OpenDocument(...) to open a document,
   // since then MapForceControl wouldn't know a control window to show
   // the document in. Instead:

   var strPath = MakeAbsolutePath("MarketingExpenses.mfd");
   var objDoc = objDocCtrl.OpenDocument(strPath);

   if (objDoc != null)
   {
objDocCtrl.setActive();
msgtext.innerText = "Opened mapping: " + strPath;
   }
   else
alert("Unable to open " + strPath);
}
```

### 18.5.2.2.7 Create Event Handler to Update Button Status

Availability of a command may vary with every mouse click or keystroke. The custom event `OnUpdateCmdUI` of MapForceControl gives us an opportunity to update the enabled/disabled state of buttons associated with MapForce commands. The method `MapForceControl.QueryStatus`[1347] is used to query whether a command is enabled or not.

```
<!--  custom event 'OnUpdateCmdUI" of MapForceControl object  -->
function objMapForceX::OnUpdateCmdUI()
{
   if ( document.readyState == "complete" )// 'complete'
   {
      // update status of buttons
      // set activity status of simulated toolbar
      GenerateXSLT.disabled =  ! (objDoc1.QueryStatus(32360) & 0x02);   // not enabled
      GenerateJava.disabled =  ! (objDoc1.QueryStatus(32358) & 0x02);   // not enabled
      GenerateCpp.disabled =   ! (objDoc1.QueryStatus(32356) & 0x02);   // not enabled
      GenerateCSharp.disabled = ! (objDoc1.QueryStatus(32357) & 0x02);   // not enabled

      btnInsertXML.disabled = ! (objDoc1.QueryStatus(32393) & 0x02);
      btnInsertDB.disabled = ! (objDoc1.QueryStatus(32389) & 0x02);
      btnInsertEDI.disabled = ! (objDoc1.QueryStatus(32390) & 0x02);
      btnInsertText.disabled = ! (objDoc1.QueryStatus(32392) & 0x02);

      btnInsertConstant.disabled = ! (objDoc1.QueryStatus(32388) & 0x02);
```

```
      btnInsertFilter.disabled = ! (objDoc1.QueryStatus(32391) & 0x02);
      btnInsertIFELSE.disabled = ! (objDoc1.QueryStatus(32394) & 0x02);
      btnInsertException.disabled = ! (objDoc1.QueryStatus(32311) & 0x02);

      btnFuncUserDef.disabled = ! (objDoc1.QueryStatus(32380) & 0x02);
      btnFuncUserDefSel.disabled = ! (objDoc1.QueryStatus(32381) & 0x02);
      btnFuncSettings.disabled = ! (objDoc1.QueryStatus(32387) & 0x02);
      btnInsertInput.disabled = ! (objDoc1.QueryStatus(32383) & 0x02);

      btnGenXSLT.disabled = ! (objDoc1.QueryStatus(32360) & 0x02);
      btnGenXSLT2.disabled = ! (objDoc1.QueryStatus(32361) & 0x02);
      btnGenXQuery.disabled = ! (objDoc1.QueryStatus(32359) & 0x02);
      btnGenCPP.disabled = ! (objDoc1.QueryStatus(32356) & 0x02);
      btnGenCSharp.disabled = ! (objDoc1.QueryStatus(32357) & 0x02);
      btnGenJava.disabled = ! (objDoc1.QueryStatus(32358) & 0x02);
   }
}
```

## 18.5.3 Java

MapForce ActiveX components can be accessed from Java code. Java integration is provided by the libraries listed below. These libraries are available in the folder `<ApplicationFolder>\Examples\JavaAPI` of your MapForce installation, after you have installed both MapForce and the MapForce Integration Package (see also Prerequisites[1292]).

- `AltovaAutomation.dll`: a JNI wrapper for Altova automation servers (in case of the 32-bit installation of MapForce)
- `AltovaAutomation_x64.dll`: a JNI wrapper for Altova automation servers (in case of the 64-bit installation of MapForce)
- `AltovaAutomation.jar`: Java classes to access Altova automation servers
- `MapForceActiveX.jar`: Java classes that wrap the MapForce ActiveX interface
- `MapForceActiveX_JavaDoc.zip`: a Javadoc file containing help documentation for the Java interface

**Note:** In order to use the Java ActiveX integration, the .dll and .jar files must be included in the Java class search path.

### Example Java project

An example Java project is supplied with your product installation. You can test the Java project and modify and use it as you like. For more details, see Example Java Project[1319].

### Rules for mapping the ActiveX Control names to Java

For the documentation of ActiveX controls, see Object Reference[1339]. Note that the object naming conventions are slightly different in Java compared to other languages. Namely, the rules for mapping between the ActiveX controls and the Java wrapper are as follows:

- ***Classes and class names***
  For every component of the MapForce ActiveX interface a Java class exists with the name of the component.

- ***Method names***
  Method names on the Java interface are the same as used on the COM interfaces but start with a small letter to conform to Java naming conventions. To access COM properties, Java methods that prefix the property name with get and set can be used. If a property does not support write-access, no setter method is available. Example: For the `IntegrationLevel` property of the MapForceControl, the Java methods `getIntegrationLevel` and `setIntegrationLevel` are available.

- ***Enumerations***
  For every enumeration defined in the ActiveX interface, a Java enumeration is defined with the same name and values.

- ***Events and event handlers***
  For every interface in the automation interface that supports events, a Java interface with the same name plus 'Event' is available. To simplify the overloading of single events, a Java class with default implementations for all events is provided. The name of this Java class is the name of the event interface plus '`DefaultHandler`'. For example:
  `MapForceControl`: Java class to access the application
  `MapForceControlEvents`: Events interface for the MapForceControl
  `MapForceControlEventsDefaultHandler`: Default handler for `MapForceControlEvents`

## Exceptions to mapping rules

There are some exceptions to the rules listed above. These are listed below:

| Interface | Changes in Java class |
|---|---|
| `MapForceControlDocument`, method `New` | Renamed to `newDocument` |
| `MapForceControlDocument`, method `OpenDocument` | Removed. Use the `Open` method |
| `MapForceControlDocument`, method `NewDocument` | Removed. Use the `newDocument` method |
| `MapForceControlDocument`, method `SaveDocument` | Removed. Use the `Save` method |

## This section

This section shows how some basic MapForce ActiveX functionality can be accessed from Java code. It is organized into the following sub-sections:

## 18.5.3.1 Example Java Project

The MapForce installation package contains an example Java project, located in the ActiveX Examples folder of the application folder: `<ApplicationFolder>\Examples\ActiveX\Java\`.

The Java example shows how to integrate the MapForceControl in a common desktop application created with Java. You can test it directly from the command line using the batch file `BuildAndRun.bat`, or you can compile and run the example project from within Eclipse. See below for instructions on how to use these procedures.

### File list

The Java examples folder contains all the files required to run the example project. These files are listed below:

| | |
|---|---|
| `.classpath` | Eclipse project helper file |
| `.project` | Eclipse project file |
| `AltovaAutomation.dll` | Java-COM bridge: DLL part (for the 32-bit installation) |
| `AltovaAutomation_x64.dll` | Java-COM bridge: DLL part (for the 64-bit installation) |
| `AltovaAutomation.jar` | Java-COM bridge: Java library part |
| `BuildAndRun.bat` | Batch file to compile and run example code from the command line prompt. Expects folder where Java Virtual Machine resides as parameter. |
| `MapForceActiveX.jar` | Java classes of the MapForce ActiveX control |
| `MapForceActiveX_JavaDoc.zip` | Javadoc file containing help documentation for the Java API |
| `MapForceContainer.java` | Java example source code |
| `MapForceContainerEventHandler.java` | Java example source code |
| `MapForceTable.java` | Java example source code |

### What the example does

The example places one MapForce document editor window, the MapForce project window, the MapForce library window and the MapForce validation window in an AWT frame window. It reads out the main menu defined for MapForce and creates an AWT menu with the same structure. You can use this menu or the project window to open and work with files in the document editor.

You can modify the example in any way you like.

The following specific features are described in code listings:

- Creating the ActiveX Controls 1321: Starts MapForce, which is registered as an automation server, or activates MapForce if it is already running.
- Loading Data in the Controls 1321: Locates one of the example documents installed with MapForce and opens it.

- Basic Event Handling[1322]: Changes the view of all open documents to Text View. The code also shows how to iterate through open documents.
- Menus[1322]: Validates the active document and shows the result in a message box. The code shows how to use output parameters.
- UI Update Event Handling[1324]: Shows how to handle MapForce events.
- Creating a MapForce Mapping Table[1325]: Shows how to create a MapForce mapping table and prepare it for modal activation.

## Updating the path to the Examples folder

Before running the provided sample, you may need to edit the **MapForceContainer.java** file. Namely, check that the following path refers to the actual folder where the MapForce example files are stored on your operating system:

```
// Locate samples installed with the product.
final String strExamplesFolder = System.getenv( "USERPROFILE" ) + "\\Documents\\Altova\
\MapForce2024\\MapForceExamples\\";
```

## Running the example from the command line

To run the example from the command line:

1. Check that all prerequisites are met (see Prerequisites[1292]).
2. Open a command prompt window, change the current directory to the sample Java project folder, and type:

   ```
   buildAndRun.bat "<Path-to-the-Java-bin-folder>"
   ```

3. Press **Enter**.

The Java source in `MapForceContainer.java` will be compiled and then executed.

## Compiling and running the example in Eclipse

To import the sample Java project into Eclipse:

1. Check that all prerequisites are met (see Prerequisites[1292]).
2. On the **File** menu, click **Import**.
3. Select **Existing Projects into Workspace**, and browse for the Eclipse project file located at `<ApplicationFolder>\Examples\ActiveX\Java\`. Since you may not have write-access in this folder, it is recommended to select the **Copy projects into workspace** check box on the Import dialog box.

To run the example application, right-click the project in Package Explorer and select the command **Run as | Java Application**.

Help for Java API classes is available through comments in code as well as the Javadoc view of Eclipse. To enable the Javadoc view in Eclipse, select the menu command **Window | Show View | JavaDoc**.

### 18.5.3.2 Creating the ActiveX Controls

The code listing below show how ActiveX controls can be created. The constructors will create the Java wrapper objects. Adding these Canvas-derived objects to a panel or to a frame will trigger the creation of the wrapped ActiveX object.

```
01   /**
02    * MapForce manager control - always needed
03    */
04   public static MapForceControl         mapForceControl = null;
05
06   /**
07    * MapForceDocument editing control
08    */
09   public static MapForceControlDocument     mapForceDocument = null;
10
11   /**
12    * Tool windows - MapForce place-holder controls
13    */
14   private static MapForceControlPlaceHolder   mapForceProjectToolWindow = null;
15   private static MapForceControlPlaceHolder   mapForceValidationToolWindow = null;
16   private static MapForceControlPlaceHolder   mapForceLibraryToolWindow = null;
17
18   // Create the MapForce ActiveX control; the parameter determines that we want
     // to place document controls and place-holder controls individually.
19   // It gives us full control over the menu, as well.
20      mapForceControl = new MapForceControl(
        ICActiveXIntegrationLevel.ICActiveXIntegrationOnDocumentLevel.getValue(), false );
21
22      mapForceDocument = new MapForceControlDocument();
23      frame.add( mapForceDocument, BorderLayout.CENTER );
24
25
26   // Create a project window and open the sample project in it
27      mapForceProjectToolWindow = new MapForceControlPlaceHolder(
        MapForceControlPlaceholderWindow.MapForceXProjectWindow.getValue(),
        strExamplesFolder + "MapForceExamples.mfp" ) ;
28      mapForceProjectToolWindow.setPreferredSize( new Dimension( 200, 200 ) );
```

### 18.5.3.3 Loading Data in the Controls

The code listing below show how data can be loaded in the ActiveX controls.

```
1    // Locate samples installed with the product.
2       final String strExamplesFolder = System.getenv( "USERPROFILE" ) +
        "\\Documents\\Altova\\MapForce2024\\MapForceExamples\\";
3       mapForceProjectToolWindow = new
MapForceControlPlaceHolder( MapForceControlPlaceholderWindow.MapForceXProjectWindow.getValu
e(), strExamplesFolder + "MapForceExamples.mfp" ) ;
```

### 18.5.3.4 Basic Event Handling

The code listing below shows how basic events can be handled. When calling the MapForceControl's `open` method, or when trying to open a file via the menu or Project tree, the `onOpenedOrFocused` event is sent to the attached event handler. The basic handling for this event is opening the file by calling the MapForceDocumentControl's `open` method.

```
01       // Open the Marketing file when button is pressed
02       btnMarkExp.addActionListener( new ActionListener() {
03         public void actionPerformed(ActionEvent e) {
04           try {
05             // Instruct the Document control to open the file - avoid calling the open
method of MapForceControl (see help)
06             mapForceDocument.open( strExamplesFolder + "MarketingExpenses.mfd" );
07             mapForceDocument.requestFocusInWindow();
08           } catch (AutomationException e1) {
09             e1.printStackTrace();
10           }
11         }
12       } );
13       public void onOpenedOrFocused( String i_strFileName, boolean
i_bOpenWithThisControl, boolean i_bFileAlreadyOpened ) throws AutomationException
14   {
15     // Handle the New/Open events coming from the Project tree or from the menus
16     if ( !i_bFileAlreadyOpened )
17     {
18       // This is basically an SDI interface, so open the file in the already existing
document control
19       try {
20         MapForceContainer.mapForceDocument.open( i_strFileName );
21         MapForceContainer.mapForceDocument.requestFocusInWindow();
22       } catch (Exception e) {
23         e.printStackTrace();
24       }
25     }
26   }
```

### 18.5.3.5 Menus

The code listing below shows how menu items can be created. Each `MapForceCommand` object gets a corresponding `MenuItem` object, with the `ActionCommand` set to the ID of the command. The actions generated by all menu items are handled by the same function, which can perform specific handlings (like reinterpreting the closing mechanism) or can delegate the execution to the `MapForceControl` object by calling its `exec` method. The `menuMap` object that is filled during menu creation is used later (see section UI Update Event Handling 1324).

```
01
02       // Load the file menu when the button is pressed
03       btnMenu.addActionListener( new ActionListener() {
04         public void actionPerformed(ActionEvent e) {
05           try {
06             // Create the menubar that will be attached to the frame
```

```
07              MenuBar mb = new MenuBar();
08              // Load the main menu's first item - the File menu
09              MapForceCommand xmlSpyMenu =
mapForceControl.getMainMenu().getSubCommands().getItem( 0 );
10              // Create Java menu items from the Commands objects
11              Menu fileMenu = new Menu();
12              handlerObject.fillMenu( fileMenu, xmlSpyMenu.getSubCommands() );
13              fileMenu.setLabel( xmlSpyMenu.getLabel().replace( "&", "" ) );
14              mb.add( fileMenu );
15              frame.setMenuBar( mb );
16              frame.validate();
17            } catch (AutomationException e1) {
18              e1.printStackTrace();
19            }
20            // Disable the button when the action has been performed
21            ((AbstractButton) e.getSource()).setEnabled( false );
22          }
23        } ) ;
24    /**
25     * Populates a menu with the commands and submenus contained in an MapForceCommands
object
26     */
27    public void fillMenu(Menu newMenu, MapForceCommands mapForceMenu) throws
AutomationException
28    {
29      // For each command/submenu in the mapForceMenu
30      for ( int i = 0 ; i < mapForceMenu.getCount() ; ++i  )
31      {
32        MapForceCommand mapForceCommand = mapForceMenu.getItem( i );
33        if ( mapForceCommand.getIsSeparator() )
34          newMenu.addSeparator();
35        else
36        {
37          MapForceCommands subCommands = mapForceCommand.getSubCommands();
38          // Is it a command (leaf), or a submenu?
39          if ( subCommands.isNull() || subCommands.getCount() == 0 )
40          {
41            // Command -> add it to the menu, set its ActionCommand to its ID and store it
in the menuMap
42            MenuItem mi = new MenuItem( mapForceCommand.getLabel().replace( "&", "" ) );
43            mi.setActionCommand( "" + mapForceCommand.getID() );
44            mi.addActionListener( this );
45            newMenu.add( mi );
46            menuMap.put( mapForceCommand.getID(), mi );
47          }
48          else
49          {
50            // Submenu -> create submenu and repeat recursively
51            Menu newSubMenu = new Menu();
52            fillMenu( newSubMenu, subCommands );
53            newSubMenu.setLabel( mapForceCommand.getLabel().replace( "&", "" ) );
54            newMenu.add( newSubMenu );
55          }
56        }
57      }
58    }
59    /**
60     * Action handler for the menu items
```

```
61    * Called when the user selects a menu item; the item's action command corresponds to
the command table for MapForce
62    */
63   public void actionPerformed( ActionEvent e )
64   {
65     try
66     {
67       int iCmd = Integer.parseInt( e.getActionCommand() );
68       // Handle explicitly the Close commands
69       switch ( iCmd )
70       {
71         case 57602:       // Close
72         case 34050:       // Close All
73           MapForceContainer.initMapForceDocument();
74           break;
75         default:
76           MapForceContainer.mapForceControl.exec( iCmd );
77           break;
78       }
79     }
80     catch ( Exception ex )
81     {
82       ex.printStackTrace();
83     }
84
85   }
```

## 18.5.3.6 UI Update Event Handling

The code listing below shows how a UI-Update event handler can be created.

```
01 /**
02    * Call-back from the MapForceControl.
03    * Called to enable/disable commands
04    */
05   @Override
06   public void onUpdateCmdUI() throws AutomationException
07   {
08     // A command should be enabled if the result of queryStatus contains the Supported
(1) and Enabled (2) flags
09     for ( java.util.Map.Entry<Integer, MenuItem> pair : menuMap.entrySet() )
10
      pair.getValue().setEnabled( MapForceContainer.mapForceControl.queryStatus( pair.getKe
y() ) > 2 );
11   }
12  /**
13    * Call-back from the MapForceControl.
14    * Usually called while enabling/disabling commands due to UI updates
15    */
16   @Override
17   public boolean onIsActiveEditor( String i_strFilePath ) throws AutomationException
18   {
19     try {
20       return
MapForceContainer.mapForceDocument.getDocument().getFullName().equalsIgnoreCase( i_strFileP
ath );
```

```
21      } catch ( Exception e ) {
22        return false;
23      }
24    }
```

## 18.5.3.7 Listing the Properties of a MapForce Mapping

The listing below shows how a Mapping object in MapForce can be loaded as a table and prepared for modal activation.

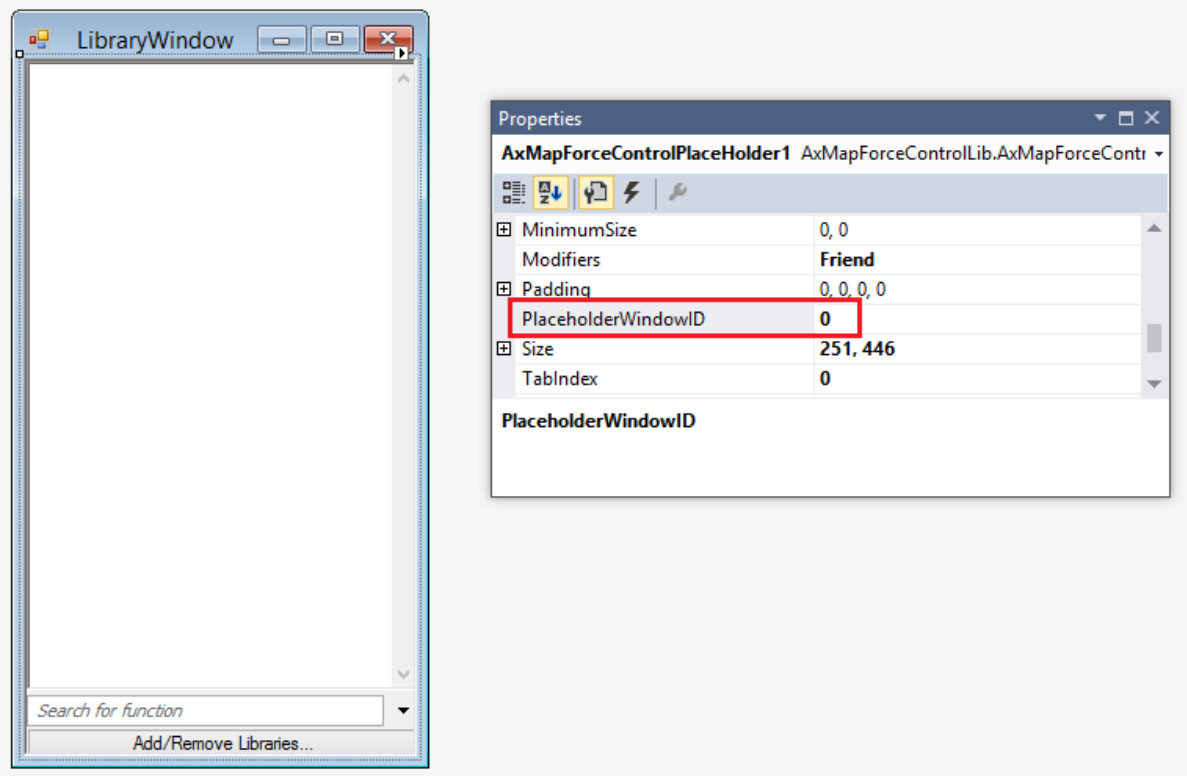
```
01 //access MapForce Java-COM bridge
02 import com.altova.automation.MapForce.*;
03 import com.altova.automation.MapForce.Component;
04 import com.altova.automation.MapForce.Enums.ENUMComponentUsageKind;
05
06 //access AWT and Swing components
07 import java.awt.*;
08 import javax.swing.*;
09 import javax.swing.table.*;
10
11
12 /**
13  * A simple example of a table control loading the structure from a Mapping object.
14  * The class receives an Mapping object, loads its components in a JTable, and prepares
15  * for modal activation.
16  *
17  * Feel free to modify and extend this sample.
18  *
19  * @author Altova GmbH
20  */
21 class MapForceTable extends JDialog
22 {
23   /**
24    * The table control
25    */
26   private JTable myTable;
27
28   /**
29    * Constructor that prepares the modal dialog containing the filled table control
30    * @param mapping The data to be displayed in the table
31    * @param parent  Parent frame
32    */
33   public MapForceTable( Mapping mapping, Frame parent )
34   {
35     // Construct the modal dialog
36     super( parent, "MapForce component table", true );
37     // Build up the tree
38     fillTable( mapping );
39     // Arrange controls in the dialog
40     setContentPane( new JScrollPane( myTable ) );
41   }
42
43   /**
44    * Loads the components of a Mapping object in the table
45    * @param mapping Source data
```

```
46    */
47   private void fillTable( Mapping mapping)
48   {
49     try
50     {
51       // count how many Instance components do we have
52       int size = 0;
53       for (Component comp : mapping.getComponents())
54         if ( comp.getUsageKind() ==
ENUMComponentUsageKind.eComponentUsageKind_Instance )
55           ++size;
56
57       // Prepare data
58       final String[] columnNames = { "Component", "Has inputs", "Has outputs", "Input
file", "Output file", "Schema" };
59       final Object[][] data = new Object[size ][ 7 ] ;
60       int index = 0 ;
61       for (Component comp : mapping.getComponents())
62         if ( comp.getUsageKind() ==
ENUMComponentUsageKind.eComponentUsageKind_Instance )
63         {
64           int i = 0;
65           data[ index ][ i++ ] = comp.getName() ;
66           data[ index ][ i++ ] = new Boolean( comp.getHasIncomingConnections() );
67           data[ index ][ i++ ] = new Boolean( comp.getHasOutgoingConnections() );
68           data[ index ][ i++ ] = comp.getInputInstanceFile();
69           data[ index ][ i++ ] = comp.getOutputInstanceFile();
70           data[ index++ ][ i ] = comp.getSchema() ;
71         }
72
73       // Set up table
74       myTable = new JTable( new AbstractTableModel() {
75           public String getColumnName(int col) { return columnNames[col]; }
76           public int getRowCount() { return data.length; }
77           public int getColumnCount() { return columnNames.length; }
78           public Object getValueAt(int row, int col) { return data[row][col]; }
79           public boolean isCellEditable(int row, int col) { return false; }
80           public Class getColumnClass(int c) { return getValueAt(0, c).getClass(); }

81       } );
82
83       // Set width
84       for( index = 0 ; index < columnNames.length ; ++index )
85         myTable.getColumnModel().getColumn( index ).setMinWidth( 80 );
86       myTable.getColumnModel().getColumn( 5 ).setMinWidth( 400 );
87     }
88     catch (Exception e)
89     {
90       e.printStackTrace();
91     }
92   }
93
94 }
```

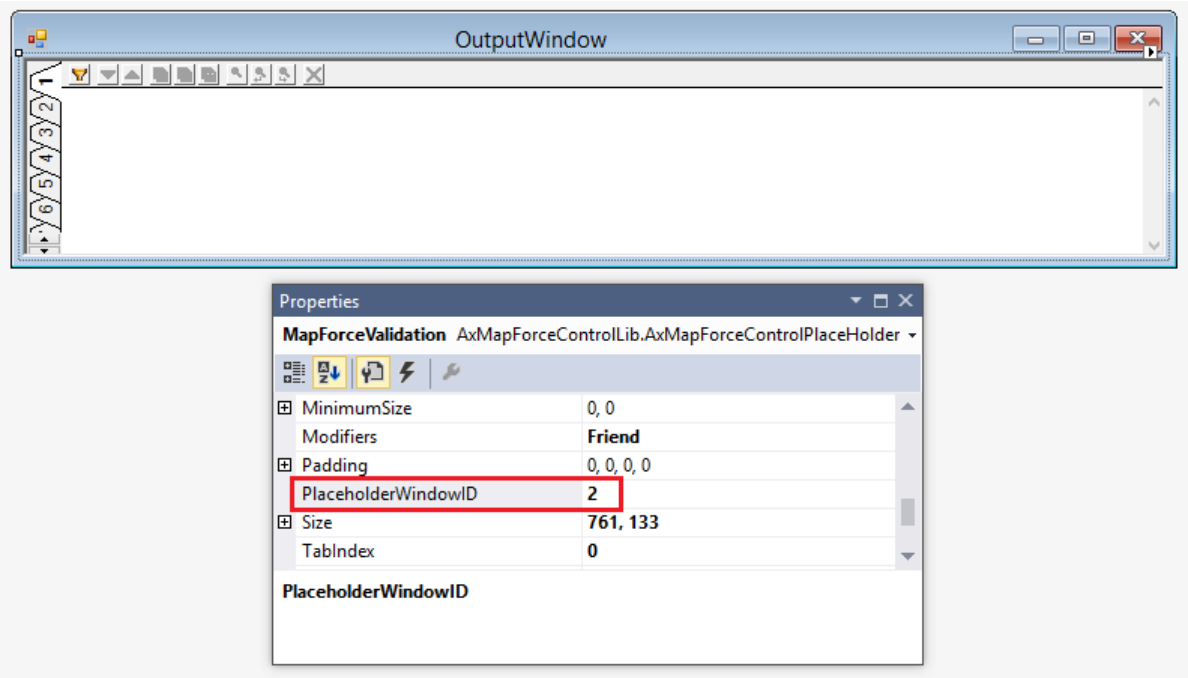
## 18.5.4 VB.NET

Source code which illustrates integration of MapForceControl into a VB.NET application can be found in the folder `<ApplicationFolder>\Examples\ActiveX\VB.NET` of your MapForce installation. The solution consists of three windows, as follows:

1. **MainWindow.vb** - the main document window, which also includes a basic application menu.
2. **LibraryWindow.vb** - the Library window. The contents of this window is populated by a Placeholder control which has the **PlaceholderWindowID** property set to 0 (this value instructs the control to display specifically the Library window).




3. **OutputWindow.vb** - the Messages (Output) window. The contents of this window is populated by a Placeholder control which has the **PlaceholderWindowID** property set to 2 (this value instructs the control to display specifically the Output window).




Before attempting to build and run this solution, note the following steps:

## Step 1: Check the prerequisites

For the list of prerequisites, see Prerequisites 1292.

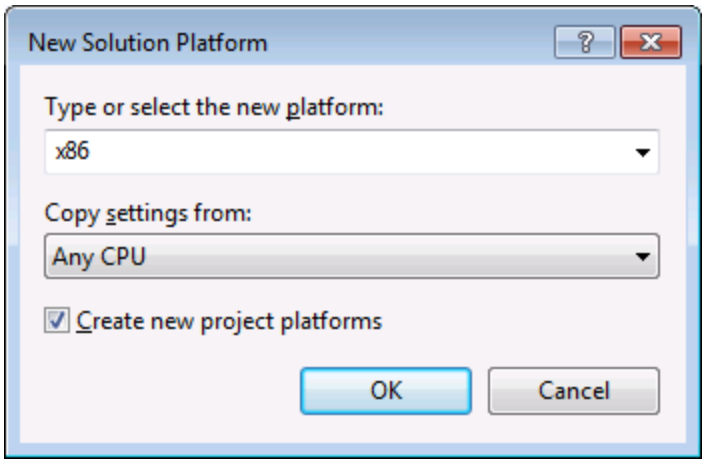
## Step 2: Copy the sample to a directory where you have write permissions

To avoid running Visual Studio as an Administrator, copy the source code to a directory where you have write permissions, instead of running it from the default location.

## Step 3: Set the build platform

- Create a build platform configuration that matches the platform under which you want to build (x86, x64). Here is how you can create the build configuration:

  a. Right-click the solution in Visual Studio, and select **Configuration Manager**.
  b. Under **Active solution platform**, select **New...** and then select the x86 or x64 configuration (in this example, **x86**).




You are now ready to build and run the solution in Visual Studio. Remember to build using the configuration that matches your target platform (x86, x64); otherwise, runtime errors might occur.

## 18.6 Command Reference

This section lists the names and identifiers of all menu commands that are available within MapForce. Every sub-section lists the commands from the corresponding top-level menu of MapForce. The command tables are organized as follows:

- The "Menu Item" column shows the command's menu text as it appears in MapForce, to make it easier for you to identify the functionality behind the command.
- The "Command Name" column specifies the string that can be used to get an icon with the same name from **ActiveX\Images** folder of the MapForce installation directory.
- The "ID" column shows the numeric identifier of the column that must be supplied as argument to methods which execute or query this command.

To execute a command, use the `MapForceControl.Exec`[1346] or the `MapForceControlDocument.Exec`[1352] methods. To query the status of a command, use the `MapForceControl.QueryStatus`[1347] or `MapForceControlDocument.QueryStatus`[1353] methods.

Depending on the edition of MapForce you have installed, some of these commands might not be supported.

### 18.6.1 "File" Menu

The "File" menu has the following commands:

| Menu item | Command name | ID |
|---|---|---|
| New... | ID_FILE_NEW | 57600 |
| Open... | ID_FILE_OPEN | 57601 |
| Save | ID_FILE_SAVE | 57603 |
| Save As... | ID_FILE_SAVE_AS | 57604 |
| Save All | ID_FILE_SAVEALL | 32377 |
| Reload | IDC_FILE_RELOAD | 32467 |
| Close | ID_WINDOW_CLOSE | 32453 |
| Close All | ID_WINDOW_CLOSEALL | 32454 |
| Print... | ID_FILE_PRINT | 57607 |
| Print Preview | ID_FILE_PRINT_PREVIEW | 57609 |
| Print Setup... | ID_FILE_PRINT_SETUP | 57606 |
| Validate Mapping | ID_MAPPING_VALIDATE | 32347 |
| Mapping Settings | ID_MAPPING_SETTINGS | 32396 |
| Generate Code in Selected Language | ID_FILE_GENERATE_SELECTED_CODE | 32362 |

| Menu item | Command name | ID |
|---|---|---|
| XSLT 1.0 | ID_FILE_GENERATEXSLT | 32360 |
| XSLT 2.0 | ID_FILE_GENERATEXSLT2 | 32361 |
| XQuery | ID_FILE_GENERATEXQUERY | 32359 |
| Java | ID_FILE_GENERATEJAVACODE | 32358 |
| C# (Sharp) | ID_FILE_GENERATECSCODE | 32357 |
| C++ | ID_FILE_GENERATECPPCODE | 32356 |
| Compile to MapForce Server Execution File... | ID_FILE_CREATE_SERVER_EXECUTION_FILE | 32517 |
| Deploy to FlowForce Server... | ID_FILE_DEPLOY_MAPPING | 32506 |
| Generate Documentation... | ID_FILE_GENERATE_DOCUMENTATION | 32468 |
| Recent File | ID_FILE_MRU_FILE1 | 57616 |
| Exit | ID_APP_EXIT | 57665 |

## 18.6.2 "Edit" Menu

The "Edit" menu has the following commands:

| Menu item | Command name | ID |
|---|---|---|
| Undo | ID_EDIT_UNDO | 57643 |
| Redo | ID_EDIT_REDO | 57644 |
| Find... | ID_EDIT_FIND | 57636 |
| Find Next | ID_EDIT_FINDNEXT | 32349 |
| Find Previous | ID_EDIT_FINDPREV | 32350 |
| Cut | ID_EDIT_CUT | 57635 |
| Copy | ID_EDIT_COPY | 57634 |
| Paste | ID_EDIT_PASTE | 57637 |
| Delete | ID_EDIT_CLEAR | 57632 |
| Select All | ID_EDIT_SELECT_ALL | 57642 |

## 18.6.3 "Insert" Menu

The "Insert" menu has the following commands:

| Menu item | Command name | ID |
|---|---|---|
| XML Schema/File... | ID_INSERT_XSD | 32393 |
| Database... | ID_INSERT_DATABASE | 32389 |
| EDI... | ID_INSERT_EDI | 32390 |
| Text File... | ID_INSERT_TXT | 32392 |
| Web Service Function... | ID_INSERT_WEBSERVICE_FUNCTION | 32319 |
| Excel 2007+ File... | ID_INSERT_EXCEL | 32376 |
| XBRL Document... | ID_INSERT_XBRL | 32469 |
| JSON Schema/File... | ID_INSERT_JSON | 32531 |
| Insert Input... | ID_FUNCTION_INSERT_INPUT | 32383 |
| Insert Output... | ID_FUNCTION_INSERT_OUTPUT | 32402 |
| Constant... | ID_INSERT_CONSTANT | 32388 |
| Variable... | ID_INSERT_VARIABLE | 32500 |
| Join | ID_INSERT_JOIN | 32581 |
| Sort: Nodes/Rows | ID_INSERT_SORT | 32444 |
| Filter: Nodes/Rows | ID_INSERT_FILTER | 32391 |
| SQL-WHERE/ORDER | ID_INSERT_SQLWHERE_CONDITION | 32351 |
| Value-Map | ID_INSERT_VALUEMAP | 32354 |
| IF-Else Condition | ID_INSRT_CONDITION | 32394 |
| Exception | ID_INSERT_EXCEPTION | 32311 |

## 18.6.4 "Project" Menu

The "Project" menu has the following commands:

| Menu item | Command name | ID |
|---|---|---|
| Reload Project | ID_PROJECT_RELOAD | 32476 |

| Menu item | Command name | ID |
|---|---|---|
| Close Project | ID_FILE_CLOSEPROJECT | 32355 |
| Save Project | ID_FILE_SAVEPROJECT | 32378 |
| Add Files to Project... | ID_PROJECT_ADDFILESTOPROJECT | 32420 |
| Add Active File to Project | ID_PROJECT_ADDACTIVEFILETOPROJECT | 32419 |
| Create Folder... | ID_PROJECT_CREATE_FOLDER | 32310 |
| Open Mapping | ID_PROJECT_OPEN_MAPPING | 32307 |
| Create Mapping for Operation... | ID_PROJECT_CREATE_MAPPING_FOR_OPERATION | 32399 |
| Add Mapping File for Operation... | ID_PROJECT_ADD_MAPPING | 32309 |
| Insert Web Service... | ID_PROJECT_INSERT_WEBSERVICE | 32306 |
| Open File in XMLSpy | ID_PROJECT_OPEN_IN_XMLSPY | 32305 |
| Generate Code for Entire Project | ID_PROJECT_GENERATE_ALL | 32303 |
| XSLT 1.0 | ID_PROJECT_GENERATEXSLTCODE_ENTIRE | 32408 |
| XSLT 2.0 | ID_PROJECT_GENERATEXSLT2CODE_ENTIRE | 32409 |
| XQuery | ID_PROJECT_GENERATEXQUERYCODE_ENTIRE | 32410 |
| Java | ID_PROJECT_GENERATEJAVACODE_ENTIRE | 32411 |
| C# (Sharp) | ID_PROJECT_GENERATECSCODE_ENTIRE | 32412 |
| C++ | ID_PROJECT_GENERATECPPCODE_ENTIRE | 32413 |
| Properties | ID_PROJECT_PROPERTIES | 32404 |
| Recent Project | ID_FILE_MRU_PROJECT1 | 32364 |

## 18.6.5 "Component" Menu

The "Component" menu has the following commands:

| Menu item | Command name | ID |
|---|---|---|
| Change Root Element... | ID_COMPONENT_CHANGEROOTELEMENT | 32334 |
| Edit Schema Definition in XMLSpy | ID_COMPONENT_EDIT_SCHEMA | 32337 |

| Menu item | Command name | ID |
|---|---|---|
| Edit FlexText Configuration | ID_COMPONENT_EDIT_MFT | 32301 |
| Add/Remove/Edit Database Objects... | ID_COMPONENT_SELECTTABLES | 32346 |
| Create Mapping to EDI X12 997 | ID_COMPONENT_CREATE_MAPPING_TO_997 | 32483 |
| Create Mapping to EDI X12 999 | ID_COMPONENT_CREATE_MAPPING_TO_999 | 32484 |
| Refresh | IDC_COMMAND_REFRESH_COMPONENT | 32373 |
| Add Duplicate Input Before | ID_COMPONENT_CREATE_DUPLICATE_ICON_BEFORE | 32503 |
| Add Duplicate Input After | ID_COMPONENT_CREATE_DUPLICATE_ICON | 32335 |
| Remove Duplicate | ID_COMPONENT_REMOVE_DUPLICATE_ICON | 32339 |
| Add Comment Before | ID_COMPONENT_ADD_COMMENT_BEFORE | 32518 |
| Add Comment After | ID_COMPONENT_ADD_COMMENT_AFTER | 32519 |
| Add Processing Instruction Before... | ID_COMPONENT_ADD_PI_BEFORE | 32520 |
| Add Processing Instruction After... | ID_COMPONENT_ADD_PI_AFTER | 32521 |
| Edit Processing Instruction Name... | ID_COMPONENT_EDIT_PI | 32524 |
| Delete Comment/Processing Instruction | ID_COMPONENT_REMOVE_COMMENT_PI | 32522 |
| Write Content as CDATA Section | ID_COMPONENT_TOGGLE_CDATA | 32525 |
| Database Table Actions | ID_POPUP_DATABASETABLEACTIONS | 32400 |
| Query Database... | ID_QUERY_DATABASE | 32341 |
| Align Tree Left | ID_COMPONENT_LEFTALIGNTREE | 32338 |
| Align Tree Right | ID_COMPONENT_RIGHTALIGNTREE | 32340 |
| Properties | ID_COMPONENT_PROPERTIES | 32336 |

## 18.6.6 "Connection" Menu

The "Connection" menu has the following commands:

| Menu item | Command name | ID |
|---|---|---|
| Auto Connect Matching Children | ID_CONNECTION_AUTOCONNECTCHILDREN | 32342 |

| Menu item | Command name | ID |
|---|---|---|
| Settings for Connect Matching Children | ID_CONNECTION_SETTINGS | 32344 |
| Connect Matching Children... | ID_CONNECTION_MAPCHILDELEMENTS | 32343 |
| Target Driven (Standard) | ID_POPUP_NORMALCONNECTION | 32401 |
| Copy-All (Copy Child Items) | ID_POPUP_NORMALWITHCHILDREN_CONNECTION | 32460 |
| Source Driven (Mixed Content) | ID_POPUP_ORDERBYSOURCECONNECTION | 32403 |
| Properties | ID_POPUP_CONNECTION_SETTINGS | 32398 |

## 18.6.7 "Function" Menu

The "Function" menu has the following commands:

| Menu item | Command name | ID |
|---|---|---|
| Create User-Defined Function... | ID_FUNCTION_CREATE_EMPTY | 32380 |
| Create User-Defined Function from Selection... | ID_FUNCTION_CREATE_FROM_SELECTION | 32381 |
| Function Settings | ID_FUNCTION_SETTINGS | 32387 |
| Remove Function | ID_FUNCTION_REMOVE | 32385 |
| Insert Input... | ID_FUNCTION_INSERT_INPUT | 32383 |
| Insert Output... | ID_FUNCTION_INSERT_OUTPUT | 32402 |

## 18.6.8 "Output" Menu

The "Output" menu has the following commands:

| Menu item | Command name | ID |
|---|---|---|
| XSLT 1.0 | ID_SELECT_LANGUAGE_XSLT | 32433 |
| XSLT 2.0 | ID_SELECT_LANGUAGE_XSLT2 | 32434 |
| XQuery | ID_SELECT_LANGUAGE_XQUERY | 32432 |
| Java | ID_SELECT_LANGUAGE_JAVA | 32431 |
| C# (Sharp) | ID_SELECT_LANGUAGE_CSHARP | 32430 |
| C++ | ID_SELECT_LANGUAGE_CPP | 32429 |

| Menu item | Command name | ID |
|---|---|---|
| Built-In Execution Engine | ID_SELECT_LANGUAGE_BUILTIN | 32490 |
| Validate Output File | ID_XML_VALIDATE | 32458 |
| Save Output File... | IDC_FILE_SAVEGENERATEDOUTPUT | 32321 |
| Save All Output Files... | IDC_FILE_SAVEALLGENERATEDOUTPUT | 32374 |
| Regenerate Output | ID_REGENERATE_PREVIEW_OUTPUT | 32480 |
| Run SQL-Script | ID_TRANSFORM_RUN_SQL | 32442 |
| Insert/Remove Bookmark | ID_TOGGLE_BOOKMARK | 32317 |
| Next Bookmark | ID_GOTONEXTBOOKMARK | 32315 |
| Previous Bookmark | ID_GOTOPREVBOOKMARK | 32314 |
| Remove All Bookmarks | ID_REMOVEALLBOOKMARKS | 32313 |
| Pretty-Print XML Text | ID_PRETTY_PRINT_OUTPUT | 32363 |
| Text View Settings | ID_TEXTVIEWSETTINGSDIALOG | 32472 |

## 18.6.9 "Debug" Menu

The "Debug" menu has the following commands:

| Menu item | Command name | ID |
|---|---|---|
| Start Debugging | ID_DEBUG_START | 32540 |
| Stop Debugging | ID_DEBUG_STOP | 32541 |
| Step Into | ID_DEBUG_STEP_INTO | 32545 |
| Step Over | ID_DEBUG_STEP_OVER | 32551 |
| Step Out | ID_DEBUG_STEP_OUT | 32552 |
| Minimal Step | ID_DEBUG_STEP_NEXT_TRACE | 32554 |

## 18.6.10 "View" Menu

The "View" menu has the following commands:

| Menu item | Command name | ID |
|---|---|---|
| Show Annotations | ID_SHOW_ANNOTATION | 32435 |

| Menu item | Command name | ID |
|---|---|---|
| Show Types | ID_SHOW_TYPES | 32437 |
| Show Library in Function Header | ID_VIEW_SHOWLIBRARYINFUNCTIONHEADER | 32448 |
| Show Tips | ID_SHOW_TIPS | 32436 |
| XBRL Display Options | ID_VIEW_XBRL_DISPLAY_OPTIONS | 32473 |
| Show Selected Component Connectors | ID_VIEW_AUTOHIGHLIGHTCOMPONENTCONNECTIONS | 32443 |
| Show Connectors from Source to Target | ID_VIEW_RECURSIVEAUTOHIGHLIGHT | 32447 |
| Zoom... | ID_VIEW_ZOOM | 32451 |
| Back | ID_CMD_BACK | 32479 |
| Forward | ID_CMD_FORWARD | 32478 |
| Status Bar | ID_VIEW_STATUS_BAR | 59393 |
| Library Window | ID_VIEW_LIBRARY_WINDOW | 32445 |
| Messages | ID_VIEW_VALIDATION_OUTPUT | 32450 |
| Overview | ID_VIEW_OVERVIEW_WINDOW | 32446 |
| Project Window | ID_VIEW_PROJECT_WINDOW | 32302 |
| Values | ID_DEBUG_VIEW_VALUES_WINDOW | 32544 |
| Context | ID_DEBUG_VIEW_CONTEXT_WINDOW | 32546 |
| Breakpoints | ID_DEBUG_VIEW_DEBUGPOINTS_WINDOW | 32547 |

## 18.6.11 "Tools" Menu

The "Tools" menu has the following commands:

| Menu item | Command name | ID |
|---|---|---|
| Global Resources | IDC_GLOBALRESOURCES | 37401 |
| <plugin not loaded> | IDC_GLOBALRESOURCES_SUBMENUENTRY1 | 37408 |
| Create Reversed Mapping | ID_CREATE_REVERSED_MAPPING | 32489 |
| Customize... | IDC_APP_TOOLS_CUSTOMIZE | 32959 |
| Options... | ID_TOOLS_OPTIONS | 32441 |

## 18.6.12 "Window" Menu

The "Window" menu has the following commands:

| Menu item | Command name | ID |
|---|---|---|
| Cascade | ID_WINDOW_CASCADE | 57650 |
| Tile Horizontal | ID_WINDOW_TILE_HORZ | 57651 |
| Tile Vertical | ID_WINDOW_TILE_VERT | 57652 |

## 18.6.13 "Help" Menu

The "Help" menu has the following commands:

| Menu item | Command name | ID |
|---|---|---|
| Table of Contents... | IDC_HELP_CONTENTS | 32966 |
| Index... | IDC_HELP_INDEX | 32967 |
| Search... | IDC_HELP_SEARCH | 32969 |
| Software Activation... | IDC_ACTIVATION | 32970 |
| Order Form... | IDC_OPEN_ORDER_PAGE | 32971 |
| Registration... | IDC_REGISTRATION | 32972 |
| Check for Updates... | IDC_CHECK_FOR_UPDATES | 32973 |
| MapForce Product Comparison... | IDC_PRODUCT_COMPARISON | 32955 |
| Support Center... | IDC_OPEN_SUPPORT_PAGE | 32961 |
| FAQ on the Web... | IDC_OPEN_FAQ_PAGE | 32962 |
| Download Components and Free Tools... | IDC_OPEN_COMPONENTS_PAGE | 32963 |
| MapForce on the Internet.. | IDC_OPEN_HOME_PAGE | 32964 |
| MapForce Training... | IDC_OPEN_TRAINING_PAGE | 32965 |
| About MapForce... | ID_APP_ABOUT | 57664 |

# 18.7 Object Reference

**Objects:**
MapForceCommand 1339
MapForceCommands 1341
MapForceControl 1342
MapForceControlDocument 1350
MapForceControlPlaceHolder 1356

To give access to standard MapForce functionality, objects of the **MapForce automation interface** can be accessed as well. See MapForceControl.Application 1343, MapForceControlDocument.Document 1351 and MapForceControlPlaceHolder.Project 1357 for more information.

## 18.7.1 MapForceCommand

**Properties:**
ID 1340
Label 1340
Name 1340
IsSeparator 1340
ToolTip 1341
StatusText 1341
Accelerator 1339
SubCommands 1341

**Description:**
A command object can be one of the following: an executable command, a command container (for example, a menu, submenu, or toolbar), or a menu separator. To determine what kind of information is stored in the current `Command` object, query its `ID`, `IsSeparator`, and `SubCommands` properties, as follows.

| The Command object is... | When... |
|---|---|
| An executable command | • `ID` is greater than zero<br>• `IsSeparator` is false<br>• `SubCommands` is empty |
| A command container | • `ID` is zero<br>• `IsSeparator` is false<br>• `SubCommands` contains a collection of `Command` objects. |
| Separator | • `ID` is zero<br>• `IsSeparator` is true |

### 18.7.1.1 Accelerator

***Property:*** `Accelerator` as string

**Description:**
Returns the accelerator key defined for the command. If the command has no accelerator key assigned, this property returns the empty string. The string representation of the accelerator key has the following format:

```
[ALT+][CTRL+][SHIFT+]key
```

Where `key` is converted using the Windows Platform SDK function `GetKeyNameText`.

### 18.7.1.2 ID

***Property:*** `ID` as long

**Description:**
This property gets the unique identifier of the command. A command's ID is required to execute the command (using `Exec`[1346]) or query its status (using `QueryStatus`[1347]). If the command is a container for other commands (for example, a top-level menu), or a separator, the ID is 0.

### 18.7.1.3 IsSeparator

***Property:*** `IsSeparator` as boolean

**Description:**
The property returns `true` if the command object is a menu separator; `false` otherwise. See also Command[1339].

### 18.7.1.4 Label

***Property:*** `Label` as string

**Description:**
This property gets the text of the command as it is displayed in the graphical user interface of MapForce. If the command is a separator, "Label" is an empty string. This property may also return an empty string for some toolbar commands that do not have any GUI text associated with them.

### 18.7.1.5 Name

***Property:*** `Name` as string

**Description:**
This property gets the unique name of the command. This value can be used to get the icon file of the command, where it is available. The available icon files can be found in the folder **<ApplicationFolder>\Examples\ActiveX\Images** of your MapForce installation.

### 18.7.1.6 StatusText

***Property:*** `Label` as string

**Description:**
The status text is the text shown in the status bar of MapForce when the command is selected. It applies only to command objects that are not separators or containers of other commands; otherwise, the property is an empty string.

### 18.7.1.7 SubCommands

***Property:*** `SubCommands` as `Commands` 1341

**Description:**
The `SubCommands` property gets the collection of `Command` 1339 objects that are sub-commands of the current command. The property is applicable only to commands that are containers for other commands (menus, submenus, or toolbars). Such container commands have the `ID` set to 0, and the `IsSeparator` property set to `false`.

### 18.7.1.8 ToolTip

***Property:*** `ToolTip` as string

**Description:**
This property gets the text that is shown as a tool-tip for each command. If the command does not have a tooltip text, the property returns an empty string.

## 18.7.2 MapForceCommands

**Properties:**
`Count` 1341
`Item` 1342

**Description:**
Collection of `Command` 1339 objects to get access to command labels and IDs of the MapForceControl. Those commands can be executed with the `Exec` 1346 method and their status can be queried with `QueryStatus` 1347.

### 18.7.2.1 Count

***Property:*** `Count` as long

**Description:**
Number of `Command` 1339 objects on this level of the collection.

### 18.7.2.2 Item

***Property:*** `Item` (n as long) as `Command` 1339

**Description:**
Gets the command with the index n in this collection. Index is 1-based.

## 18.7.3 MapForceControl

**Properties:**
`IntegrationLevel` 1344
`Appearance` 1343
`Application` 1343
`BorderStyle` 1343
`CommandsList` 1343
`EnableUserPrompts` 1344
`MainMenu` 1344
`Toolbars` 1345

**Methods:**
`Open` 1346
`Exec` 1346
`QueryStatus` 1347

**Events:**
`OnUpdateCmdUI` 1349
`OnOpenedOrFocused` 1349
`OnCloseEditingWindow` 1347
`OnFileChangedAlert` 1348
`OnContextChanged` 1348
`OnDocumentOpened` 1348
`OnValidationWindowUpdated` 1349

This object is a complete ActiveX control and should only be visible if the MapForce library is used in the Application Level mode.

```
CLSID: A38637E9-5759-4456-A167-F01160CC22C1
ProgID: Altova.MapForceControl
```

### 18.7.3.1 Properties

The following properties are defined:

`IntegrationLevel` 1344
`EnableUserPrompts` 1344
`Appearance` 1343
`BorderStyle` 1343

Command related properties:

CommandsList[1343]
MainMenu[1344]
Toolbars[1345]

Access to MapForceAPI:
Application[1343]

### 18.7.3.1.1 Appearance

***Property:*** `Appearance` as short

***Dispatch Id:*** *-520*

**Description:**
A value not equal to `0` displays a client edge around the control. Default value is `0`.

### 18.7.3.1.2 Application

***Property:*** `Application` as Application

***Dispatch Id:*** *1*

**Description:**
The `Application` property gives access to the `Application` object of the complete MapForce automation server API. The property is read-only.

### 18.7.3.1.3 BorderStyle

***Property:*** `BorderStyle` as short

***Dispatch Id:*** *-504*

**Description:**
A value of `1` displays the control with a thin border. Default value is `0`.

### 18.7.3.1.4 CommandsList

***Property:*** `CommandList` as `Commands`[1341] (read-only)

***Dispatch Id:*** *1004*

**Description:**
This property returns a flat list of all commands defined available with MapForceControl. To get commands organized according to their menu structure, use `MainMenu`[1344]. To get toolbar commands, use `Toolbars`[1345].

```
public void GetAllMapForceCommands()
{
    // Get all commands from the MapForce ActiveX control assigned to the current form
    MapForceControlLib.MapForceCommands commands = this.axMapForceControl1.CommandList;
    // Iterate through all commands
    for (int i = 0; i < commands.Count; i++)
    {
      // Get each command by index and output it to the console
      MapForceControlLib.MapForceCommand cmd = axMapForceControl1.CommandList[i];
      Console.WriteLine("{0} {1} {2}", cmd.ID, cmd.Name, cmd.Label.Replace("&", ""));
    }
}
```

*C# example*

### 18.7.3.1.5 EnableUserPrompts

***Property:*** `EnableUserPrompts` as boolean

***Dispatch Id:*** *1006*

**Description:**
Setting this property to *false*, disables user prompts in the control. The default value is *true*.

### 18.7.3.1.6 IntegrationLevel

***Property:*** `IntegrationLevel` as `ICActiveXIntegrationLevel` 1358

***Dispatch Id:*** *1000*

**Description:**
The `IntegrationLevel` property determines the operation mode of the control. See also Integration at Application Level 1296 and Integration at Document Level 1299 for more information.

**Note:** It is important to set this property immediately after the creation of the `MapForceControl` object.

### 18.7.3.1.7 MainMenu

***Property:*** `MainMenu` as Command 1339 (read-only)

***Dispatch Id:*** *1003*

**Description:**
This property provides information about the structure and commands available in the MapForceControl main menu, as a `Command` object. The `Command` object contains all available submenus of MapForce (for example "File", "Edit", "View" etc.). To access the submenu objects, use the `SubCommands` property of the `MainMenu` property. Each submenu is also a `Command` object. For each submenu, you can then further iterate through their

SubCommands property in order to get their corresponding child commands and separators (this technique may be used, for example, to create the application menu programmatically). Note that some menu commands act as containers ("parents") for other menu commands, in which case they also have a SubCommands property. To get the structure of all menu commands programmatically, you will need a recursive function, as illustrated for C# in Retrieving Command Information[1306].

```
public void GetMapForceMenus()
{
    // Get the main menu from the MapForce ActiveX control assigned to the current form
    MapForceControlLib.MapForceCommand mainMenu = this.axMapForceControl1.MainMenu;

    // Loop through entries of the main menu (e.g. File, Edit, etc.)
    for (int i = 0; i < mainMenu.SubCommands.Count; i++)
    {
      MapForceControlLib.MapForceCommand menu = mainMenu.SubCommands[i];
      Console.WriteLine("{0} menu has {1} children items (including separators)",
menu.Label.Replace("&", ""), menu.SubCommands.Count);
    }
}
```

*C# example*

### 18.7.3.1.8 Toolbars

***Property:*** Toolbars as Commands[1341] (read-only)

***Dispatch Id:*** *1005*

**Description:**
This property provides information about the structure of MapForceControl toolbars, as a Command object. The Command object contains all available toolbars of MapForce. To access the toolbars, use the SubCommands property of the Toolbars property. Each toolbar is also a Command object. For each toolbar, you can then further iterate through their SubCommands property in order to get their commands (this technique may be used, for example, to create the application's toolbars programmatically).

```
public void GetMapForceToolbars()
{
    // Get the application toolbars from the MapForce ActiveX control assigned to the
current form
    MapForceControlLib.MapForceCommands toolbars = this.axMapForceControl1.Toolbars;

    // Iterate through all toolbars
    for (int i = 0; i < toolbars.Count; i++)
    {
      MapForceControlLib.MapForceCommand toolbar = toolbars[i];
      Console.WriteLine();
      Console.WriteLine("The toolbar \"{0}\" has the following commands:",
toolbar.Label);

      // Iterate through all commands of this toolbar
```

```
      for (int j = 0; j < toolbar.SubCommands.Count; j++)
      {
         MapForceControlLib.MapForceCommand cmd = toolbar.SubCommands[j];
         // Output only command objects that are not separators
         if (!cmd.IsSeparator)
         {
            Console.WriteLine("{0}, {1}, {2}", cmd.ID, cmd.Name, cmd.Label.Replace("&",
"", ""));
         }
      }
   }
}
```

*C# example*

## 18.7.3.2 Methods

The following methods are defined:

Open 1346
Exec 1346
QueryStatus 1347

### 18.7.3.2.1 Exec

***Method:*** `Exec` (`nCmdID` as long) as boolean

***Dispatch Id:*** *6*

**Description:**
This method calls the MapForce command with the ID `nCmdID`. If the command can be executed, the method returns `true`. To get a list of all available commands, use `CommandsList` 1343. To retrieve the status of any command, use `QueryStatus` 1347.

### 18.7.3.2.2 Open

***Method:*** `Open` (`strFilePath` as string) as boolean

***Dispatch Id:*** *5*

**Description:**
The result of the method depends on the extension passed in the argument `strFilePath`. If the file extension is `.sps`, a new document is opened. If the file extension is `.svp`, the corresponding project is opened. If a different file extension is passed into the method, the control tries to load the file as a new component into the active document.

Do not use this method to load documents or projects when using the control in document-level integration mode. Instead, use `MapForceControlDocument.Open` 1353 and `MapForceControlPlaceHolder.OpenProject` 1357.

#### 18.7.3.2.3 QueryStatus

***Method:*** `QueryStatus` (`nCmdID` as long) as long

***Dispatch Id:*** *7*

**Description:**
`QueryStatus` returns the enabled/disabled and checked/unchecked status of the command specified by `nCmdID`. The status is returned as a bit mask.

| Bit | Value | Name | Meaning |
|---|---|---|---|
| 0 | 1 | Supported | Set if the command is supported. |
| 1 | 2 | Enabled | Set if the command is enabled (can be executed). |
| 2 | 4 | Checked | Set if the command is checked. |

This means that if `QueryStatus` returns `0` the command ID is not recognized as a valid MapForce command. If `QueryStatus` returns a value of `1` or `5`, the command is disabled.

### 18.7.3.3 Events

The MapForceControl ActiveX control provides the following connection point events:

`OnUpdateCmdUI` 1349
`OnOpenedOrFocused` 1349
`OnCloseEditingWindow` 1347
`OnFileChangedAlert` 1348
`OnContextChanged` 1348

`OnDocumentOpened` 1348
`OnValidationWindowUpdated` 1349

#### 18.7.3.3.1 OnCloseEditingWindow

***Event:*** `OnCloseEditingWindow` (i_strFilePath as String) as boolean

***Dispatch Id:*** *1002*

**Description:**
This event is triggered when MapForce needs to close an already open document. As an answer to this event, clients should close the editor window associated with *i_strFilePath*. Returning *true* from this event indicates that the client has closed the document. Clients can return *false* if no specific handling is required and MapForceControl should try to close the editor and destroy the associated document control.

#### 18.7.3.3.2 OnContextChanged

***Event:*** `OnContextChanged` (`i_strContextName` as String, `i_bActive` as bool) as bool

***Dispatch Id:*** *1004*

**Description:**
This event is not used in MapForce

#### 18.7.3.3.3 OnDocumentOpened

***Event:*** `OnDocumentOpened` (`objDocument` as Document)

***Dispatch Id:*** *1*

**Description:**
This event is triggered whenever a document is opened. The argument `objDocument` is a `Document` object from the MapForce automation interface and can be used to query for more details about the document, or perform additional operations. When integrating on document-level, it is often better to use the event `MapForceControlDocument.OnDocumentOpened` 1355 instead.

#### 18.7.3.3.4 OnFileChangedAlert

***Event:*** `OnFileChangedAlert` (`i_strFilePath` as String) as bool

***Dispatch Id:*** *1001*

**Description:**
This event is triggered when a file loaded with MapForceControl is changed on the hard disk by another application. Clients should return true, if they handled the event, or false, if MapForce should handle it in its customary way, i.e. prompting the user for reload.

#### 18.7.3.3.5 OnLicenseProblem

***Event:*** `OnLicenseProblem` (`i_strLicenseProblemText` as String)

***Dispatch Id:*** *1005*

**Description:**
This event is triggered when MapForceControl detects that no valid license is available for this control. In case of restricted user licenses this can happen some time after the control has been initialized. Integrators should use this event to disable access to this control's functionality. After returning from this event, the control will block access to its functionality (e.g. show empty windows in its controls and return errors on requests).

#### 18.7.3.3.6 OnOpenedOrFocused

***Event:*** `OnOpenedOrFocused` (`i_strFilePath` as String, `i_bOpenWithThisControl` as bool)

***Dispatch Id:*** *1000*

**Description:**
When integrating at application level, this event informs clients that a document has been opened, or made active by MapForce.

When integrating at document level, this event instructs the client to open the file `i_strFilePath` in a document window. If the file is already open, the corresponding document window should be made the active window.

if `i_bOpenWithThisControl` is `true`, the document must be opened with MapForceControl, since internal access is required. Otherwise, the file can be opened with different editors.

#### 18.7.3.3.7 OnToolWindowUpdated

***Event:*** `OnToolWindowUpdated(pToolWnd` as long `)`

***Dispatch Id:*** *1006*

**Description:**
This event is triggered when the tool window is updated.

#### 18.7.3.3.8 OnUpdateCmdUI

***Event:*** `OnUpdateCmdUI ()`

***Dispatch Id:*** *1003*

**Description:**
Called frequently to give integrators a good opportunity to check status of MapForce commands using `MapForceControl.QueryStatus`[1347]. Do not perform long operations in this callback.

#### 18.7.3.3.9 OnValidationWindowUpdated

***Event:*** `OnValidationWindowUpdated` ()

***Dispatch Id:*** *3*

**Description:**
This event is triggered whenever the validation output window is updated with new information.

## 18.7.4 MapForceControlDocument

**Properties:**
Appearance 1351
BorderStyle 1351
Document 1351
IsModified 1351
Path 1351
ReadOnly 1352

**Methods:**
Exec 1352
New 1352
Open 1353
QueryStatus 1353
Reload 1353
Save 1353
SaveAs 1354

**Events:**
OnDocumentOpened 1355
OnDocumentClosed 1354
OnModifiedFlagChanged 1355
OnContextChanged 1354
OnFileChangedAlert 1355
OnActivate 1354

If the MapForceControl is integrated in the Document Level mode each document is displayed in an own object of type `MapForceControlDocument`. The `MapForceControlDocument` contains only one document at the time but can be reused to display different files one after another.

This object is a complete ActiveX control.

```
CLSID: DFBB0871-DAFE-4502-BB66-08CEB7DF5255
ProgID: Altova.MapForceControlDocument
```

### 18.7.4.1 Properties

The following properties are defined:

ReadOnly 1352
IsModified 1351
Path 1351
Appearance 1351
BorderStyle 1351

Access to MapForceAPI:
Document 1351

#### 18.7.4.1.1 Appearance

***Property:*** `Appearance` as short

***Dispatch Id:*** *-520*

**Description:**
A value not equal to `0` displays a client edge around the document control. Default value is `0`.

#### 18.7.4.1.2 BorderStyle

***Property:*** `BorderStyle` as short

***Dispatch Id:*** *-504*

**Description:**
A value of `1` displays the control with a thin border. Default value is `0`.

#### 18.7.4.1.3 Document

***Property:*** `Document` as Document

***Dispatch Id:*** *1*

**Description:**
The `Document` property gives access to the `Document` object of the MapForce automation server API. This interface provides additional functionality which can be used with the document loaded in the control. The property is read-only.

#### 18.7.4.1.4 IsModified

***Property:*** `IsModified` as boolean (read-only)

***Dispatch Id:*** *1006*

**Description:**
`IsModified` is *true* if the document content has changed since the last open, reload or save operation. It is *false*, otherwise.

#### 18.7.4.1.5 Path

***Property:*** `Path` as string

***Dispatch Id:*** *1005*

**Description:**
Sets or gets the full path name of the document loaded into the control.

#### 18.7.4.1.6 ReadOnly

***Property:*** `ReadOnly` as boolean

***Dispatch Id:*** *1007*

**Description:**
Using this property you can turn on and off the read-only mode of the document. If `ReadOnly` is `true` it is not possible to do any modifications.

### 18.7.4.2 Methods

The following methods are defined:

Document handling:
`New` 1352
`Open` 1353
`Reload` 1353
`Save` 1353
`SaveAs` 1354

Command Handling:
`Exec` 1352
`QueryStatus` 1353

#### 18.7.4.2.1 Exec

***Method:*** `Exec` (`nCmdID` as long) as boolean

***Dispatch Id:*** *8*

**Description:**
`Exec` calls the MapForce command with the ID `nCmdID`. If the command can be executed, the method returns `true`. This method should be called only if there is currently an active document available in the application.

To get commands organized according to their menu structure, use the `MainMenu` 1344 property of MapForceControl. To get toolbar commands, use the `Toolbars` 1345 property of the MapForceControl.

#### 18.7.4.2.2 New

***Method:*** `New` () as boolean

***Dispatch Id:*** *1000*

**Description:**
This method initializes a new document inside the control.

### 18.7.4.2.3 Open

***Method:*** `Open` (`strFileName` as string) as boolean

***Dispatch Id:*** *1001*

**Description:**
`Open` loads the file `strFileName` as the new document into the control.

### 18.7.4.2.4 QueryStatus

***Method:*** `QueryStatus` (`nCmdID` as long) as long

***Dispatch Id:*** *9*

**Description:**
`QueryStatus` returns the enabled/disabled and checked/unchecked status of the command specified by `nCmdID`. The status is returned as a bit mask.

| Bit | Value | Name | Meaning |
|---|---|---|---|
| 0 | 1 | Supported | Set if the command is supported. |
| 1 | 2 | Enabled | Set if the command is enabled (can be executed). |
| 2 | 4 | Checked | Set if the command is checked. |

This means that if `QueryStatus` returns `0` the command ID is not recognized as a valid MapForce command. If `QueryStatus` returns a value of `1` or `5` the command is disabled. The client should call the `QueryStatus` method of the document control if there is currently an active document available in the application.

### 18.7.4.2.5 Reload

***Method:*** `Reload` () as boolean

***Dispatch Id:*** *1002*

**Description:**
`Reload` updates the document content from the file system.

### 18.7.4.2.6 Save

***Method:*** `Save` () as boolean

***Dispatch Id:*** *1003*

**Description:**
Save saves the current document at the location Path[1351].

#### 18.7.4.2.7 SaveAs

***Method:*** SaveAs (strFileName as string) as boolean

***Dispatch Id:*** *1004*

**Description:**
SaveAs sets Path[1351] to *strFileName* and then saves the document to this location.

### 18.7.4.3 Events

The MapForceControlDocument ActiveX control provides following connection point events:

OnDocumentOpened[1355]
OnDocumentClosed[1354]
OnModifiedFlagChanged[1355]
OnContextChanged[1354]
OnFileChangedAlert[1355]
OnActivate[1354]
OnSetEditorTitle[1356]

#### 18.7.4.3.1 OnActivate

***Event:*** OnActivate ()

***Dispatch Id:*** *1005*

**Description:**
This event is triggered when the document control is activated, has the focus, and is ready for user input.

#### 18.7.4.3.2 OnContextChanged

***Event:*** OnContextChanged (i_strContextName as String, i_bActive as bool) as bool

***Dispatch Id:*** *1004*

**Description:** None

#### 18.7.4.3.3 OnDocumentClosed

***Event:*** OnDocumentClosed (objDocument as Document)

***Dispatch Id:*** *1001*

**Description:**
This event is triggered whenever the document loaded into this control is closed. The argument `objDocument` is a `Document` object from the MapForce automation interface and should be used with care.

#### 18.7.4.3.4 OnDocumentOpened

***Event:*** `OnDocumentOpened` (`objDocument` as Document)

***Dispatch Id:*** *1000*

**Description:**
This event is triggered whenever a document is opened in this control. The argument `objDocument` is a `Document` object from the MapForce automation interface, and can be used to query for more details about the document, or perform additional operations.

#### 18.7.4.3.5 OnDocumentSaveAs

***Event:*** `OnContextDocumentSaveAs` (`i_strFileName` as String)

***Dispatch Id:*** *1007*

**Description:**
This event is triggered when this document gets internally saved under a new name.

#### 18.7.4.3.6 OnFileChangedAlert

***Event:*** `OnFileChangedAlert` () as bool

***Dispatch Id:*** *1003*

**Description:**
This event is triggered when the file loaded into this document control is changed on the hard disk by another application. Clients should return true, if they handled the event, or false, if MapForce should handle it in its customary way, i.e. prompting the user for reload.

#### 18.7.4.3.7 OnModifiedFlagChanged

***Event:*** `OnModifiedFlagChanged` (`i_bIsModified` as boolean)

***Dispatch Id:*** *1002*

**Description:**
This event gets triggered whenever the document changes between modified and unmodified state. The parameter *i_bIsModifed* is *true* if the document contents differs from the original content, and *false*, otherwise.

#### 18.7.4.3.8 OnSetEditorTitle

***Event:*** OnSetEditorTitle ()

***Dispatch Id:*** *1006*

**Description:**
This event is being raised when the contained document is being internally renamed.

## 18.7.5 MapForceControlPlaceHolder

**Properties available for all kinds of placeholder windows:**
`PlaceholderWindowID` 1357

**Properties for project placeholder window:**
`Project` 1357

**Methods for project placeholder window:**
`OpenProject` 1357
`CloseProject` 1357

The `MapForceControlPlaceHolder` control is used to show the additional MapForce windows like Overview, Library or Project window. It is used like any other ActiveX control and can be placed anywhere in the client application.

```
CLSID: FDEC3B04-05F2-427d-988C-F03A85DE53C2
ProgID: Altova.MapForceControlPlaceHolder
```

### 18.7.5.1 Properties

The following properties are defined:

`PlaceholderWindowID` 1357

Access to MapForceAPI:
`Project` 1357

#### 18.7.5.1.1 Label

***Property:*** `Label` as String (read-only)

***Dispatch Id:*** *1001*

**Description:**
This property gives access to the title of the placeholder. The property is read-only.

#### 18.7.5.1.2 PlaceholderWindowID

***Property:*** `PlaceholderWindowID as MapForceControlPlaceholderWindow`[1359]

***Dispatch Id:*** *1*

**Description:**
This property specifies which MapForce window should be displayed in the client area of the control. The `PlaceholderWindowID` can be set at any time to any valid value of the `MapForceControlPlaceholderWindow`[1359] enumeration. The control changes its state immediately and shows the new MapForce window.

#### 18.7.5.1.3 Project

***Property:*** `Project` as Project (read-only)

***Dispatch Id:*** *2*

**Description:**
The `Project` property gives access to the `Project` object of the MapForce automation server API. This interface provides additional functionality which can be used with the project loaded into the control. The property will return a valid project interface only if the placeholder window has `PlaceholderWindowID`[1357] with a value of `MapForceXProjectWindow (=3)`. The property is read-only.

### 18.7.5.2 Methods

The following method is defined:

`OpenProject`[1357]
`CloseProject`[1357]

#### 18.7.5.2.1 OpenProject

***Method:*** `OpenProject` (`strFileName` as `string`) as `boolean`
Dispatch Id: *3*
**Description:**
`OpenProject` loads the file `strFileName` as the new project into the control. The method will fail if the placeholder window has a **`PlaceholderWindowID`**[1357] different to `MapForceXProjectWindow (=3)`.

#### 18.7.5.2.2 CloseProject

**Method:** `CloseProject ()`

**Dispatch Id:** *4*

**Description:**
`CloseProject` closes the project loaded by the control. The method will fail if the placeholder window has a `PlaceholderWindowID`[1357] different to `MapForceXProjectWindow (=3)`.

### 18.7.5.3 Events

The MapForceControlPlaceholder ActiveX control provides following connection point events:

`OnModifiedFlagChanged`[1358]

#### 18.7.5.3.1 OnModifiedFlagChanged

***Event:*** `OnModifiedFlagChanged` (`i_bIsModified` as boolean)

***Dispatch Id:*** *1*

**Description:**
This event gets triggered only for placeholder controls with a `PlaceholderWindowID`[1357] of `MapForceXProjectWindow (=3)`. The event is fired whenever the project content changes between modified and unmodified state. The parameter *i_bIsModifed* is *true* if the project contents differs from the original content, and *false*, otherwise.

#### 18.7.5.3.2 OnSetLabel

***Event:*** `OnSetLabel(i_strNewLabel` as string`)`

***Dispatch Id:*** *1000*

**Description:**
Raised when the title of the placeholder window is changed.

## 18.7.6 Enumerations

The following enumerations are defined:

`ICActiveXIntegrationLevel`[1358]
`MapForceControlPlaceholderWindow`[1359]

### 18.7.6.1 ICActiveXIntegrationLevel

Possible values for the IntegrationLevel[1344] property of the MapForceControl.

```
ICActiveXIntegrationOnApplicationLevel  = 0
ICActiveXIntegrationOnDocumentLevel     = 1
```

## 18.7.6.2 MapForceControlPlaceholderWindow

This enumeration contains the list of the supported additional MapForce windows.

```
MapForceXNoWindow          = -1
MapForceXLibraryWindow             = 0
MapForceXOverviewWindow            = 1
MapForceXValidationWindow          = 2
MapForceXProjectWindow             = 3
MapForceXDebuggerValuesWindow      = 4
MapForceXDebuggerContextWindow     = 5
MapForceXDebuggerPointsWindow      = 6
```

# 19 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 :

# 19.1 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.

MapForce est disponible en option en tant que plug-in pour les environnements de développement intégrés suivants :

- Visual Studio 2012/2013/2015/2017/2019/2022. Voir MapForce Plug-in pour Visual Studio 911.
- Eclipse 2024-03 (4.31), 2023-12 (4.30), 2023-09 (4.29), 2023-06 (4.28). Voir MapForce Plug-in pour Eclipse 914.

MapForce peut être intégré aux produits Microsoft Office comme suit :

- Il peut mapper des données de ou vers les bases de données Access. Pour des versions prises en charge, voir Bases de données et MapForce 158
- Il peut générer de la documentation de mappage dans Word 2000 ou versions ultérieures. Voir Générer et personnaliser la documentation de mappage 823.

## 19.1.1 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 Professional 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.

| Format de mappage | | XSLT 1.0 | XSLT 2.0 | XSLT 3.0 | XQuery | C++ | C# | Java | BUILT-IN |
|---|---|---|---|---|---|---|---|---|---|
| XML[1] | | ● | ● | ● | ● | ● | ● | ● | ● |
| CSV et texte | | | | | | ● | ● | ● | ● |
| Fichiers binaires | | | | | | | | | ● |
| Bases de données[2] | ADO | | | | | ● | ● | | ● |
| | ADO.NET | | | | | | ● | | ● |
| | JDBC | | | | | | | ● | ● |

| Format de mappage | | XSLT 1.0 | XSLT 2.0 | XSLT 3.0 | XQuery | C++ | C# | Java | BUILT-IN |
|---|---|---|---|---|---|---|---|---|---|
| | Native SQLite | | | | | | | | ● |
| | Native PostgreSQL | | | | | | | | ● |
| | ODBC | | | | | ● | ● | ● | ● |

Notes de bas de page :

1. XML avec un traitement de signatures numériques est uniquement pris en charge par MapForce Enterprise Edition en utilisant BUILT-IN en tant que langage de transformation.
2. Les limites s'appliquent au type de base de données et à l'environnement cible. Pour plus d'informations, voir Mappages de base de données dans des environnements d'exécution divers 159.

## 19.1.2 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 Professional Edition**.

| Fonction | XSLT 1.0 | XSLT 2.0 | XSLT 3.0 | XQuery | C++ | C# | Java | BUILT-IN |
|---|---|---|---|---|---|---|---|---|
| Fournir des paramètres au mappage 398 | ● | ● | ● | ● | ● | ● | ● | ● |
| Fournir les noms de fichier d'entrée dynamiquement depuis le mappage 787 | ● | ● | ● | ● | ● | ● | ● | ● |
| Fournir des noms de fichier de caractère générique en tant qu'entrée de mappage 787 1 | | ● | ● | ● | ● | ● | ● | ● |
| Générer les noms de fichier de sortie dynamiquement depuis le mappage 787 | | ● | ● | | ● | ● | ● | ● |

| Fonction | XSLT 1.0 | XSLT 2.0 | XSLT 3.0 | XQuery | C++ | C# | Java | BUILT-IN |
|---|---|---|---|---|---|---|---|---|
| Retourner des valeurs de string depuis le mappage 409 | ● | ● | ● | | ● | ● | ● | ● |
| Variables 413 | | ● | ● | ● | ● | ● | ● | ● |
| Composants de tri 438 | | ● | ● | ● | | | | ● |
| Fonctions de regroupement 597 | | ● | ● | | ● | ● | ● | ● |
| Filtres 444 | ● | ● | ● | ● | ● | ● | ● | ● |
| Composants Join 426 | | | | | | | | ● |
| Composants Value-Map 451 | ● | ● | ● | ● | ● | ● | ● | ● |
| Valeurs par défaut et fonctions de nœud 476 | | | | | | | | ● |
| Exceptions de mappage 463 | | ● | ● | ● | ● | ● | ● | ● |
| Parsage de string et sérialisation 794 2 | | | | | | | | ● |
| Noms de nœuds dynamiques 766 | | ● | ● | ● | ● | ● | ● | ● |
| Insertions bulk de base de données 280 | | | | | | | | ● |
| Base de données SQL SELECT sans paramètres d'entrée 264 | | | | | ● | ● | ● | ● |
| Base de données SQL SELECT avec paramètres d'entrée 264 | | | | | | | | ● |
| Procédures stockées dans | | | | | | | | ● |

| Fonction | XSLT 1.0 | XSLT 2.0 | XSLT 3.0 | XQuery | C++ | C# | Java | BUILT-IN |
|---|---|---|---|---|---|---|---|---|
| base de données 321 | | | | | | | | |
| Gestion d'exception de base de données 293 3 | | | | | ● | ● | ● | ● |
| Traçage de base de données et logging d'erreurs 256 | | | | | | | | ● |
| Générer des fichiers d'exécution MapForce Server 864 | | | | | | | | ● |
| Déployer des mappages sur FlowForce Server 867 | | | | | | | | ● |
| Lire des données depuis des fichiers binaires 666 | | | | | | | | ● |
| Écrire des données depuis des fichiers binaires 668 | | | | | | | | ● |

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.
2. Pour JSON, le parsage et la sérialisation sont en plus pris en charge dans et C#.
3. La gestion d'exception de base de données est possible quand le langage de mappage est pris en charge par le pilote de base de données connecté actuellement, tel qu'indiqué dans la table précédente.

## 19.2 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.

### 19.2.1 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 :



#### 19.2.1.1 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, ` ` .

## 19.2.1.2 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`.

## 19.2.1.3 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[1371]. Pour

utiliser une collation spécifique, fournir son URI tel que donné dans la liste des collations prises en charge[1371]. 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`.

## 19.2.2 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[1373] 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()`.

### 19.2.2.1 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 1375

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 1378
- Géolocalisation 1395
- Liée à l'image 1404
- Numérique 1409

- Séquence 1431
- String 1440
- Divers 1446

### 19.2.2.1.1 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 1375 ]

## 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 1375 ]

### 19.2.2.1.2 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** 1378 ]

## 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 1378 ]

## 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 1378 ]

## 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 1378 ]

## 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*) **as**xs: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*) **as**xs: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*) **as**xs: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 1378 ]

## 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 1378 ]

## 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 1378 ]

## 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 1378 ]

## 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 1378 ]

## 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 1378 ]

## 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 1378 ]

## 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 1378 ]

### 19.2.2.1.3 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`[1404] 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`)<br>**`D°M'S.SS"N/S D°M'S.SS"E/W`**<br>*Exemple :* **`33°55'11.11"N 22°44'66.66"W`** |

| **2** |
|---|
| Degrés décimaux, avec orientation suffixée (`N/S, E/O`)<br>**`D.DDN/S D.DDE/W`**<br>*Exemple :* **`33.33N 22.22W`** |

| **3** |
|---|
| Degrés, minutes, secondes décimales, avec signe préfixé (`+/-`); le signe plus (**`N/E`**) est optionnel<br>**`+/-D°M'S.SS" +/-D°M'S.SS"`**<br>*Exemple :* **`33°55'11.11" -22°44'66.66"`** |

| **4** |
|---|
| Degrés décimaux, avec signe préfixé (`+/-`); le signe plus (**`N/E`**) est optionnel<br>**`+/-D.DD +/-D.DD`**<br>*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`[1404] et l'attribut `@Geolocation`[1404] 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`[1404] et l'attribut `@Geolocation`[1404] 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`[1404] et l'attribut `@Geolocation`[1404] 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 1404 et l'attribut @Geolocation 1404 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[1404] et l'attribut @Geolocation[1404] 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 1395 ]

## 19.2.2.1.4 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*

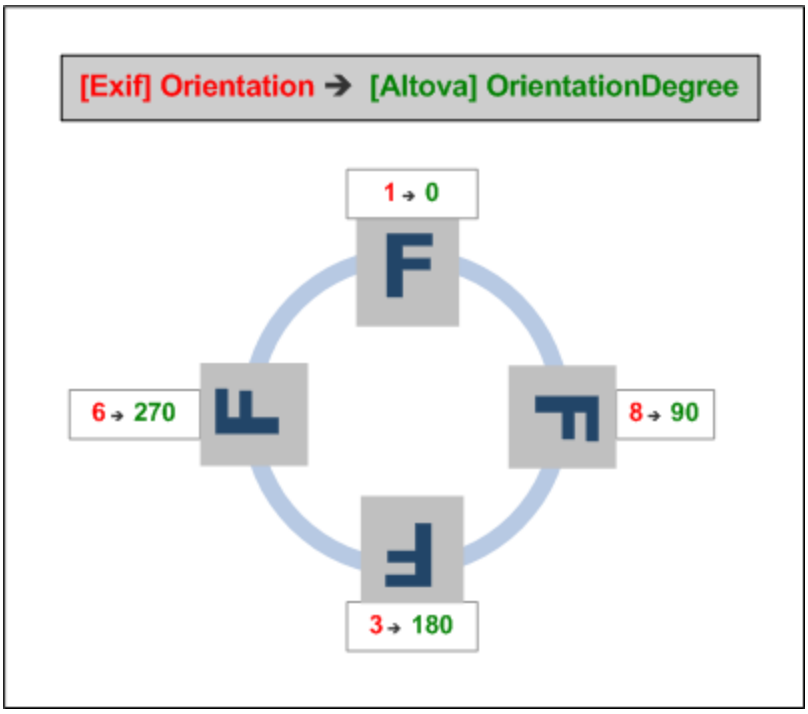
La machine Altova XPath/XQuery génère l'attribut **Geolocation** personnalisable depuis les onglets standard de métadonnées Exif. **Geolocation** est une concaténation de quatre onglets Exif : `GPSLatitude`, `GPSLatitudeRef`, `GPSLongitude`, `GPSLongitudeRef`, avec des unités ajoutées (voir table ci-dessous).

| 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 1404 ]

### 19.2.2.1.5 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 1409 ]

## 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 1409 ]

## Fonctions de formatage de numéro

[ Top 1409 ]

### 19.2.2.1.6 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 |

### 19.2.2.1.7 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)

### 19.2.2.1.8 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"

### 19.2.2.1.9 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 1446 ]

## 19.2.2.2 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 et .NET, et pour les scripts MSXSL pour XSLT 1449. 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
- Fonctions d'extension .NET
- Scripts MSXSL pour XSLT 1449

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.

## 19.2.2.2.1 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.

# 19.3 Données techniques

Cette section contient des informations utiles concernant certains aspects techniques de votre logiciel. Cette information est organisée dans les sections suivantes :



## 19.3.1 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.

## 19.3.2 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.

## 19.3.3 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.

## 19.3.4 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.

# 19.4 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.

## 19.4.1 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.

## 19.4.2 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.

## 19.4.3 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

# H



# I

# J



# L

# M

# N

# O



# P

# Q



# R

# S

# T

# U



# V



# W

# X



# Y



# Z