Altova MapForce 2025 Basic Edition

Benutzer- und Referenzhandbuch

Altova MapForce 2025 Basic Edition
Benutzer- und Referenzhandbuch

All rights reserved. No parts of this work may be reproduced in any form or by any means - graphic, electronic, or mechanical, including photocopying, recording, taping, or information storage and retrieval systems - without the written permission of the publisher.

Products that are referred to in this document may be either trademarks and/or registered trademarks of the respective owners. The publisher and the author make no claim to these trademarks.

While every precaution has been taken in the preparation of this document, the publisher and the author assume no responsibility for errors or omissions, or for damages resulting from the use of information contained in this document or from the use of programs and source code that may accompany it. In no event shall the publisher and the author be liable for any loss of profit or any other commercial damage caused or alleged to have been caused directly or indirectly by this document.

Published: 2025

© 2019-2025 Altova GmbH

Inhaltsverzeichnis

1 Einführung 9

1.1 Neue Funktionen 10
- 1.1.1 Version 2025 10
- 1.1.2 Version 2024 10
- 1.1.3 Version 2023 11
- 1.1.4 Version 2022 12
- 1.1.5 Version 2021 13

1.2 Was ist MapForce? 14
- 1.2.1 Mapping: Quellen und Ziele 15
- 1.2.2 Mapping-Szenarien 16
- 1.2.3 Transformationssprachen 16
- 1.2.4 Integration mit Altova-Produkten 18

1.3 Übersicht über die Benutzeroberfläche 19
- 1.3.1 Leisten 20
- 1.3.2 Fenster 20
- 1.3.3 Fenster "Meldungen" 24
- 1.3.4 Bereiche 25

2 Mapping-Grundlagen 29

2.1 Komponenten 31
- 2.1.1 Hinzufügen von Komponenten 35
- 2.1.2 Komponentengrundlagen 38
- 2.1.3 Dateipfade 40

2.2 Verbindungen 45
- 2.2.1 Verbindungsarten 49
- 2.2.2 Verbindungseinstellungen 56
- 2.2.3 Kontextmenü für Verbindungen 57
- 2.2.4 Fehlerhafte Verbindungen 59
- 2.2.5 Beibehalten von Verbindungen nach Löschen von Komponenten 61

2.3 Allgemeine Verfahren und Funktionalitäten ... 63
2.3.1 Validierung ... 63
2.3.2 Codegenerierung ... 65
2.3.3 Funktionalitäten der Textansicht ... 67
2.3.4 Suchen in der Textansicht ... 71
2.3.5 Mapping-Einstellungen ... 74
2.4 Grundlegende Regeln und Strategien ... 77
2.4.1 Sequenzen ... 78
2.4.2 Kontext und Verarbeitungsreihenfolge ... 79
2.4.3 Parent-Kontext ... 83
2.4.4 Prioritätskontext ... 87
2.4.5 Mehrere Zielkomponenten ... 92

3 Tutorials 96

3.1 Eine Quellkomponente auf eine Zielkomponente ... 97
3.1.1 Erstellen und Speichern eines Designs ... 98
3.1.2 Hinzufügen der Quellkomponente ... 99
3.1.3 Hinzufügen der Zielkomponente ... 101
3.1.4 Verbinden von Quell- und Zielkomponente ... 102
3.1.5 Vorschau auf das Mapping-Ergebnis ... 104
3.2 Mehrere Quellkomponenten auf eine Zielkomponente ... 106
3.2.1 Vorbereiten des Mapping-Designs ... 107
3.2.2 Hinzufügen einer zweiten Quellkomponente ... 107
3.2.3 Konfigurieren der Ausgabe ... 108
3.2.4 Verbinden der zweiten Quellkomponente mit der Zielkomponente ... 109
3.3 Verkettetes Mapping ... 111
3.3.1 Vorbereiten des Mapping-Designs ... 111
3.3.2 Konfigurieren der zweiten Zieldatei ... 112
3.3.3 Ziehen der Verbindungen ... 113
3.3.4 Filtern der Daten ... 114
3.3.5 Anzeige einer Vorschau und Speichern der Ausgabe ... 117
3.4 Mehrere Quellkomponenten auf mehrere Zielkomponenten ... 120
3.4.1 Konfigurieren des Input ... 122
3.4.2 Konfigurieren des Ausgabeteils 1 ... 123

3.4.3 Konfigurieren des Ausgabeteils 2 ... 126

4 Strukturkomponenten 129

4.1 XML und XML-Schema ... 130
4.1.1 XML-Komponenteneinstellungen ... 131
4.1.2 Abgeleitete Typen ... 136
4.1.3 NULL-Werte ... 138
4.1.4 Kommentare und Processing Instructions ... 140
4.1.5 CDATA-Abschnitte ... 140
4.1.6 Wildcards - xs:any / xs:anyAttribute ... 142
4.1.7 Benutzerdefinierte Namespaces ... 144
4.1.8 Schema-Manager ... 147

5 Transformationskomponenten 163

5.1 Einfache Input-Komponente ... 164
5.1.1 Hinzufügen von einfachen Input-Komponenten ... 165
5.1.2 Einstellungen für einfache Input-Komponenten ... 166
5.1.3 Erstellen eines Input-Standardwerts ... 167
5.1.4 Beispiel: Verwenden von Dateinamen als Mapping-Parameter ... 168
5.2 Einfache Output-Komponente ... 171
5.2.1 Hinzufügen einfacher Output-Komponenten ... 172
5.2.2 Beispiel: Vorschau auf die Funktionsausgabe ... 173
5.3 Variablen ... 175
5.3.1 Hinzufügen einer Variablen ... 177
5.3.2 Geltungsbereich und Kontext von Variablen ... 181
5.3.3 Beispiel: Filtern und Nummerieren von Nodes ... 184
5.3.4 Beispiel: Unterteilen von Datensätzen in Gruppen und Untergruppen ... 185
5.4 Sortieren von Komponenten ... 188
5.4.1 Sortieren nach mehreren Schlüsseln ... 190
5.4.2 Sortieren mit Variablen ... 192
5.5 Filter und Bedingungen ... 194
5.5.1 Beispiel: Filtern von Nodes ... 196
5.5.2 Beispiel: Rückgabe eines Werts auf Basis einer Bedingung ... 198

5.6 Wertezuordnungen ... 200
5.6.1 Beispiel: Ersetzen von Wochentagen ... 205
5.6.2 Beispiel: Ersetzen von Stellenbezeichnungen ... 208
5.7 Funktionen ... 212
5.7.1 Grundlegendes zu Funktionen ... 212
5.7.2 Verwalten von Funktionsbibliotheken ... 215
5.7.3 Benutzerdefinierte Funktionen ... 219
5.7.4 Spezielle benutzerdefinierte Funktionen ... 239
5.7.5 Regular Expressions ... 247
5.7.6 Referenz Funktionsbibliothek ... 250

6 Komplexe Mappings 404

6.1 Mappen von Node-Namen ... 405
6.1.1 Zugriff auf Node-Namen ... 406
6.1.2 Zugriff auf Nodes eines bestimmten Typs ... 414
6.1.3 Beispiel: Mappen von Elementnamen auf Attributwerte ... 418
6.2 Stapel-Verarbeitung von Dateien ... 422
6.2.1 Beispiel: Aufteilen einer XML-Datei in mehrere ... 424

7 Automatisieren mit Altova-Produkten 427

7.1 Automatisierung mit RaptorXML Server ... 428
7.2 MapForce-Befehlszeilenschnittstelle ... 429

8 Globale Altova-Ressourcen 432

8.1 Einrichten globaler Ressourcen Teil 1 ... 433
8.2 Einrichten globaler Ressourcen Teil 2 ... 435
8.3 XML-Dateien als globale Ressourcen ... 438
8.4 Ordner als globale Ressourcen ... 440

9 Kataloge in MapForce 442

9.1 Funktionsweise von Katalogen ... 443
9.2 Katalogstruktur in MapForce ... 445

9.3 Anpassen Ihrer Kataloge ... 447
9.4 Umgebungsvariablen ... 449

10 ActiveX Integration 451

10.1 Voraussetzungen ... 452
10.2 Hinzufügen der ActiveX Controls zur Toolbox ... 454
10.3 Integration auf Applikationsebene ... 455
10.4 Integration auf Dokumentebene ... 457
10.5 Beispiele zur ActiveX-Integration ... 461
10.5.1 C# ... 461
10.5.2 Java ... 462
10.6 Befehlsreferenz ... 467
10.7 Objektreferenz ... 468
10.7.1 MapForceCommand ... 468
10.7.2 MapForceCommands ... 470
10.7.3 MapForceControl ... 471
10.7.4 MapForceControlDocument ... 478
10.7.5 MapForceControlPlaceHolder ... 484
10.7.6 Enumerationen ... 487

11 Menübefehle 488

11.1 Datei ... 489
11.2 Bearbeiten ... 492
11.3 Einfügen ... 493
11.4 Komponente ... 496
11.5 Verbindung ... 498
11.6 Funktion ... 499
11.7 Ausgabe ... 500
11.8 Ansicht ... 502
11.9 Extras ... 504
11.9.1 Anpassen von Menüs ... 505
11.9.2 Anpassen von Tastaturkürzeln ... 506
11.9.3 Optionen ... 508

11.10 Fenster ... 517
11.11 Hilfe ... 519

12 Anhänge 524

12.1 Anmerkungen zur Unterstützung ... 525
12.1.1 Unterstützte Quellen und Ziele ... 525
12.1.2 Unterstützte Funktionalitäten im generierten Code ... 525
12.2 Informationen zu den Prozessoren ... 527
12.2.1 Informationen zum XSLT- und XQuery-Prozessor ... 527
12.2.2 XSLT- und XPath/XQuery-Funktionen ... 533
12.3 Technische Daten ... 633
12.3.1 OS- und Arbeitsspeicheranforderungen ... 633
12.3.2 Altova-Prozessoren ... 633
12.3.3 Unicode-Unterstützung ... 634
12.3.4 Internet-Verwendung ... 634
12.4 Lizenzinformationen ... 636
12.4.1 Electronic Software Distribution ... 636
12.4.2 Software-Aktivierung und Lizenzüberwachung ... 637
12.4.3 Altova Endbenutzer-Lizenzvereinbarung ... 638
12.4.4 Verpacken von Lizenzdateien in den MapForce Installer ... 638

Index 639

1 Einführung

Altova MapForce 2025 Basic Edition ist ein leistungsstarkes Datentransformations- und ETL-Tool zur Integration von Daten. MapForce ist eine 32/64-Bit Windows-Applikation, die auf Windows 10, Windows 11 und Windows Server 2016 oder höher läuft. 64-Bit-Unterstützung steht für die Enterprise und die Professional Edition zur Verfügung.

Sie können mit MapForce Daten aus jedem und in jedes beliebige Format konvertieren. MapForce verfügt über eine grafische Benutzeroberfläche 19, die eine Reihe von Optionen zum Verwalten, Visualisieren, Bearbeiten und Ausführen einzelner Mappings und komplexer Mapping-Projekte enthält. MapForce bietet für die Datentransformation eine umfangreiche Bilbliothek von Datenverarbeitungs- und -konvertierungsfunktionen 212, mit denen Daten den Anforderungen Ihres Datenintegrationsprojekts gemäß gefiltert und bearbeitet werden können.

Nachdem Sie Ihr Mapping erstellt haben, können Sie in einem separaten Fenster eine Vorschau auf die Ausgabe anzeigen und die Ausgabe im gewünschten Ordner speichern. Zusätzlich dazu können Sie Code für die externe Ausführung generieren 65.

Durch die Integration von MapForce mit anderen Altova-Produkten können Sie MapForce-Funktionalitäten noch erweitern.

- Ihre Mappings können auch auf MapForce Server ausgeführt werden. Dadurch können Geschäftsvorgänge, für die Daten regelmäßig transformiert werden müssen, automatisiert werden. MapForce Server beinhaltet einen leistungsstarken Datentransformationsprozessor und kann beliebige Datenkonvertierungen durchführen. Beachten Sie, dass es sich hierbei um einen plattformübergreifenden Server handelt, der auf Windows-, macOS und Linux-Systemen zur Verfügung steht.
- RaptorXML Server ist ein ultraschneller Prozessor, der Ihre Instanzen validiert.
- Mit Hilfe von FlowForce Server können Sie Aufgaben automatisieren und Ihre Mappings in Form geplanter Aufträge ausführen.
- StyleVision Server generiert Ausgabedokumente in HTML, RTF, PDF und Word.
- Mit Hilfe von StyleVision können Sie StyleVision Power Stylesheets erstellen, anhand derer StyleVision Server Ausgabedokumente in mehreren Formaten generieren kann.
- DatabaseSpy ist ein vielseitiges Tool, mit dem Sie Datenbanken erstellen, bearbeiten und abfragen können.
- XMLSpy ist besonders gut geeignet, um Ihre Mapping-Dateien zu bearbeiten. Über einige MapForce-Dialogfelder können Dateien direkt in XMLSpy geöffnet werden.
- Sie können MapForce auch als Plug-in von Microsoft Visual Studio und Eclipse verwenden. Dadurch können Sie direkt von der Entwicklungsumgebung Ihrer Wahl aus auf die MapForce-Funktionalitäten zugreifen.

Letzte Aktualisierung: 17.03.2025

1.1 Neue Funktionen

In diesem Abschnitt finden Sie eine Beschreibung der neuen Funktionalitäten in den einzelnen MapForce Release-Versionen. Nähere Informationen dazu finden Sie im jeweiligen Unterabschnitt.

1.1.1 Version 2025

Version 2025 Release 2

- Unterstützung für Shopify/GraphQL APIs (*Enterprise Edition*).
- Unterstützung für MySQL 9, DB2 12.1 und SQLite 3.47.2 (*Professional und Enterprise Edition*).
- Interne Aktualisierungen und Verbesserungen

Version 2025

- Unterstützung für die OpenAPI-Spezifikationen 2.0, 3.0 und 3.1 für HTTP APIs (*Enterprise Edition*).
- Im PDF Extractor können Sie nun auf der Benutzeroberfläche und zur Laufzeit nach Text suchen (*Enterprise Edition*).
- In Antwort auf empfangene EDIFACT-Nachrichten gesendete EDIFACT CONTRL-Nachrichten können nun automatisch generiert werden (*Enterprise Edition*).
- Unterstützung für UN/EDIFACT 2023A (*Enterprise Edition*).
- Unterstützung für SWIFT 2024 (*Enterprise Edition*).
- Unterstützung für Azure CosmosDB (*Enterprise Edition*).
- Interne Aktualisierungen und Verbesserungen

1.1.2 Version 2024

Version 2024 Release 2

- Über das Projekt **`MapForceExamples`** können nun verschiedene Video-Tutorials aufgerufen werden (*Professional und Enterprise Edition*). Des Weiteren können Sie auch Ihre eigenen Links zu externen Ressourcen hinzufügen.
- Wenn Sie Ihr Mapping auf FlowForce Server bereitstellen, können Sie die Mapping-Dateien für den späteren Abruf anhängen. Dadurch stellen Sie sicher, dass Ihre Mapping-Dateien nicht verloren gehen und Sie diese jederzeit wieder herunterladen können (*Professional und Enterprise Edition*).
- Für Datenbankkomponenten kann nun zur Laufzeit eine gemeinsame Datenbankverbindung verwendet werden (*Professional und Enterprise Edition*).
- Es kann nun anhand von Enumerationstypen eine Wertezuordnung in XML- (*alle Editionen*) und XBRL-Komponenten (*Enterprise Edition*) erstellt werden. Dank dieser Funktionalität lassen sich Enumerationswerte schneller und leichter zuordnen: Beide Seiten der Wertezuordnung werden mit allen Enumerationswerten vorausgefüllt. Sie müssen die entsprechenden Werte der Wertezuordnung nur mehr überprüfen und bearbeiten. Nähere Informationen dazu finden Sie unter Wertezuordnungen 204.
- Unterstützung für NET 8.0 bei der C#-Codegenerierung (*Professional und Enterprise Edition*). Nähere Informationen dazu finden Sie unter Codegenerierung 65.

 © 2019-2025 Altova GmbH

- Unterstützung für FORTRAS EDI-Nachrichten (*Enterprise Edition*).
- Unterstützung für PostgreSQL 16, MySQL 8.2, MySQL 8.3, MariaDB 11.2, SQLite 3.45 (*Professional und Enterprise Edition*).
- Interne Aktualisierungen und Verbesserungen

Version 2024

- Es gibt nun ein neues MapForce-Tool namens PDF Extractor (*Enterprise Edition*). Mit Hilfe von PDF Extractor können Sie PDF-Extraktionsvorlagen erstellen, die in MapForce importiert und als Quellkomponenten in Ihren Mappings verwendet werden können.
- In MapForce (*Enterprise Edition*) lassen sich nun auch KI-gestützte Mappings erstellen. Sie können in MapForce REST Webservice-Aufrufe an eine API wie OpenAI API, Azure OpenAI API, AWS AI Services, usw. erstellen.
- Unterstützung für SWIFT 2023 (*Enterprise Edition*).
- Es gibt nun eine neue **`sleep`**-Funktion, mit Hilfe derer Daten nach einer festgelegten Wartezeit übergeben werden können (*Professional und Enterprise Edition*).
- Es gibt nun native Unterstützung für MySQL und MariaDB (*Professional und Enterprise editions*).
- Die Funktion zum Verbinden identer Sub-Einträge wurde verbessert und um neue Übereinstimmungsoptionen erweitert. Nähere Informationen dazu finden Sie unter Verbindungen mit identen Sub-Einträgen 52.
- In OAuth-Anmeldeinformationen werden nun zusätzlich zum Grant Type *Autorisierungscode* die Grant Types *Client-Anmeldeinformationen* und *Passwort-Anmeldeinformationen des Ressourcenbesitzers* unterstützt (*Enterprise Edition*).
- Interne Aktualisierungen und Verbesserungen

1.1.3 Version 2023

Version 2023 Release 2

- In der XML-Deklaration von XML-Zieldateien kann nun das Attribut `standalone="yes"` generiert werden. Nähere Informationen dazu finden Sie unter XML-Komponenteneinstellungen 134.
- Das Hilfe-System 519 wurde umgestaltet. Standardmäßig wird die Online-Hilfe aufgerufen, doch haben Sie die Option, alternativ dazu standardmäßig das lokal installierte PDF-Benutzerhandbuch zu verwenden 508.
- Zu einem Mapping können nun Kommentare im Stil einer Notiz hinzugefügt werden. Nähere Informationen dazu finden Sie unter Kommentare 33.
- Es gibt neue Einstellungen zum Definieren der Netzwerk-Einstellungen 513.
- Unterstützung für VDA EDI-Nachrichten (*Enterprise Edition*).
- Interne Aktualisierungen und Verbesserungen

Version 2023

- Es werden nun die folgenden Designs unterstützt: *Klassisch*, *Hell* und *Dunkel*. Nähere Informationen dazu finden Sie unter Fenster 517.
- Interne Aktualisierungen und Verbesserungen
- Die Eclipse-Unterstützung wurde aktualisiert und auf die folgenden Versionen ausgeweitet: 2022-09, 2022-06, 2022-03, 2021-12 (*Professional und Enterprise Edition*).
- Unterstützung für ODETTE EDI-Nachrichten (*Enterprise Edition*).

- Unterstützung für die XII Transformation Registry 5 Specification (*Enterprise Edition*).
- Es besteht nun die Möglichkeit, datenbankbasierte UDF-Parameter 227 und Variablen 176 mit einer hierarchischen Struktur damit in Zusammenhang stehender Tabellen zu erstellen (*Professional und Enterprise Edition*).
- Es ist nun möglich, eine `application/x-www-form-urlencoded` Request-Struktur an einen REST-Service zu senden (*Enterprise Edition*).
- Unterstützung für das UN/EDIFACT D.21B- und D.22A-Verzeichnis (*Enterprise Edition*).
- Unterstützung für SQLite 3.39.2, MariaDB 10.9.2 und PostgreSQL 14.5 (*Professional und Enterprise Edition*).
- Unterstützung für den XML-Schema-Manager 147, ein Tool, mit dem Sie XML-Schemas installieren und zentral verwalten können, um diese in allen XML-Schema-fähigen Applikationen von Altova verwenden zu können.
- Unterstützung für mapbare EDI-Trennzeichen (*Enterprise Edition*). Diese Funktion wird derzeit für die folgenden EDI-Standards unterstützt: EDIFACT, X12 und NCPDP SCRIPT.

1.1.4 Version 2022

Version 2022 Release 2

- Interne Aktualisierungen und Optimierungen
- Die Eclipse-Unterstützung wurde aktualisiert und auf die folgenden Versionen ausgeweitet: 2021-12; 2021-09; 2021-06; 2021-03 (*Professional und Enterprise Edition*).
- Unterstützung für Visual Studio 2022 im MapForce Plug-in für Visual Studio und bei der Codegenerierung (*Professional und Enterprise Edition*).
- Unterstützung für NET 6.0 bei der Codegenerierung (*Professional und Enterprise Edition*).
- Unterstützung neuer Datenbankversionen: PostgreSQL 14, SQLite 3.37.2, MariaDB 10.6.5, MySQL 8.0.28, IBM DB2 11.5.7 (*Professional und Enterprise Edition*).
- Von Bildern kann im Fenster **Projekt** nun eine Vorschau angezeigt werden (*Professional und Enterprise Edition*).
- Es können nun EBA-konforme Filing Indicators für XBRL-Zielkomponenten erstellt werden (*Enterprise Edition*).

Version 2022

- Interne Aktualisierungen und Optimierungen
- Die Eclipse-Unterstützung wurde aktualisiert und auf die folgenden Versionen ausgeweitet: 2021-09; 2021-06; 2021-03; 2020-12 (*Professional und Enterprise Edition*).
- "Alles kopieren"-Verbindungen 54 unterstützen nun JSON. Diese Funktionalität steht nur für kompatible JSON-Typen zur Verfügung (*Enterprise Edition*).
- Es gibt ein neues StyleVision-Ausgabefenster namens *Text*. Wenn einer Komponente eine SPS-Datei zugewiesen wird, können Sie in MapForce eine Vorschau auf das neue reine Textausgabeformat anzeigen (*Professional und Enterprise Edition*).
- Unterstützung für JSON-Schema in Variablen 175 und Parametern von benutzerdefinierten Funktionen 225 (*Enterprise Edition*).
- Unterstützung für NoSQL-Datenbanken: MongoDB und CouchDB (*Enterprise Edition*).
- Es steht nun eine neue `bson`-Funktionsbibliothek zur Verfügung, mit Hilfe derer Sie einige der BSON-Typen erstellen und bearbeiten können (*Enterprise Edition*).
- Unterstützung für das EDIFACT D.20B- und D.21A-Verzeichnis.
- Unterstützung für SWIFT 2021.

 © 2019-2025 Altova GmbH

1.1.5 Version 2021

Version 2021 Release 3

- Unterstützung für den neuen JSON Schema Draft 2019-09 und Draft 2020-12 (*nur Enterprise Edition*).

Version 2021 Release 2

- Unterstützung von XSLT 3.0 als Mapping-Sprache. Siehe Generieren von XSLT-Code 65. Außerdem enthält MapForce nun vordefinierte Funktionen, die unterstützt werden, wenn als Mapping-Sprache XSLT 3.0 ausgewählt ist. Nähere Informationen dazu finden Sie unter Referenz Funktionsbibliothek 250.
- Interne Aktualisierungen und Verbesserungen

Version 2021

- Interne Aktualisierungen und Verbesserungen

1.2 Was ist MapForce?

Altova Website: Datenmapping-Tool

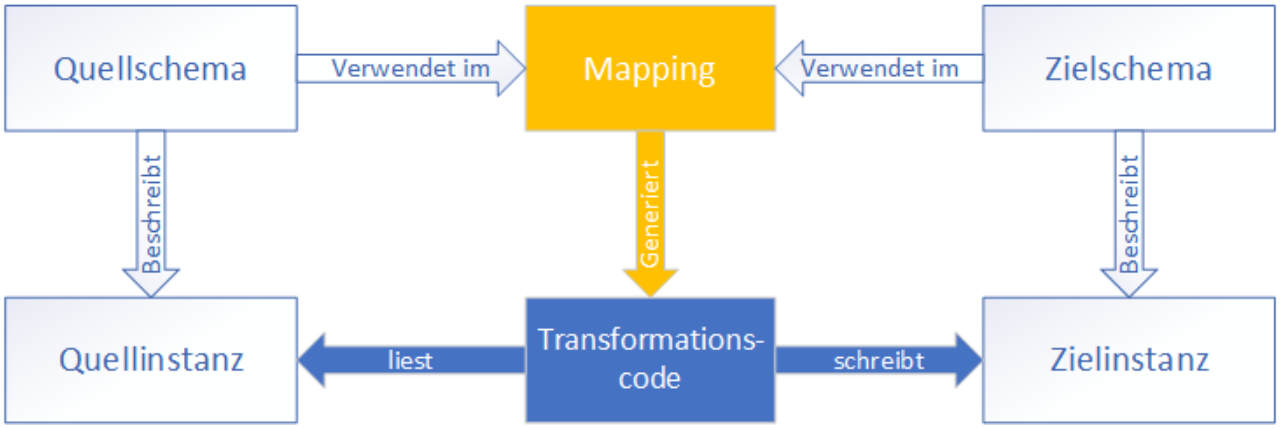
MapForce ist ein leistungsstarkes grafisches Tool zur Konvertierung und Integration beliebiger Datenformate. Eine vollständige Liste der verfügbaren Datenformate finden Sie unter Mapping: Quellen und Ziele 15. Ein Mapping besteht normalerweise aus einer oder mehreren Datenquell- und einer oder mehreren Datenzielkomponenten 31. Außerdem kann ein Mapping eine oder mehrere Transformationskomponenten 32 enthalten, die umfassende Datenverarbeitungs- und -filterungsoptionen bieten. Nähere Informationen über verschiedene Mapping-Szenarien finden Sie unter Mapping-Szenarien 16 und in den Tutorials 96.

Um ein Mapping durchführen zu können, müssen Sie eine Datenstruktur zur Beschreibung der Struktur der einzelnen Quell- und Zieldateien bereitstellen. So definiert etwa ein XML-Schema die Struktur eines XML-Dokuments. Das Mapping (der Quell- auf die Zieldaten) erfolgt über eine grafische Benutzeroberfläche mittels Drag-and-Drop. Für das Mapping muss kein Programmcode geschrieben werden, da der Code von MapForce für Sie generiert wird. Mit Hilfe dieses Codes können Sie Dokumente mit einer bestimmten Datenquellstruktur in Dokumente mit einer bestimmten Datenzielstruktur transformieren.

Alle Editionen von MapForce stehen als 32-Bit-Applikation zur Verfügung. Die MapForce Professional und die Enterprise Edition stehen zusätzlich dazu auch als 64-Bit-Applikation zur Verfügung.

Abstraktes Modell

Im unten gezeigten abstrakten Modell wird eines der grundlegende Szenarien einer Datentransformation in MapForce dargestellt. Im Quellschema ist die Struktur der Quellinstanzdatei beschrieben. Im Zielschema ist die Struktur der Zielinstanzdatei beschrieben. Quell- und Zielschema können je nach Bedarf dieselbe oder eine andere Struktur haben. Wenn Sie die Quell- und die Zielkomponente miteinander verbinden, wird Transformationscode (in der ausgewählten Transformationssprache 16) generiert, der Daten aus der Quellinstanzdatei ausliest und in die Zielinstanzdatei schreibt. Ein Beispiel für die Implementierung dieses Datentransformationsmodells finden Sie im Tutorial 1 97.



In realen Anwendungsszenarien können Sie eine beliebige Kombination aus Datenquellen (z.B. XML-, EDI- und Textdateien) verwenden und diese auf eine beliebige Kombination von Zielkomponenten (z.B. eine Datenbank und eine Excel-Datei) mappen.

 © 2019-2025 Altova GmbH

Konventionen

Die in diesem Handbuch dargestellten und referenzierten Mapping-Dateien befinden sich in den folgenden Ordnern:

- `C:\Benutzer\<Benutzername>\Dokumente\Altova\MapForce2025\MapForce Examples`
- `C:\Benutzer\<Benutzername>\Dokumente\Altova\MapForce2025\MapForceExamples\Tutorials.`
- `C:\Benutzer\<Benutzername>\Dokumente\Altova\MapForce2025\MapForceExamples\Tutorials\BasicTutorials`

In diesem Abschnitt

Dieser Abschnitt ist in die folgenden Kapitel gegliedert:

- Mapping: Quellen und Ziele 15
- Mapping-Szenarien 16
- Transformationssprachen 16
- Integration mit Altova-Produkten 18

1.2.1 Mapping: Quellen und Ziele

Mit den wichtigen Begriffen *Quelle* und *Ziel* werden in MapForce Datenstrukturen bezeichnet, von bzw. auf die Daten gemappt werden. Unten finden Sie eine Liste der Technologien, die als Mapping-Quellen und -Ziele verwendet werden können.

MapForce Basic Edition

- XML und XML-Schema

MapForce Professional Edition

- XML und XML-Schema
- Flat Files, einschließlich der Formate CSV (Comma Separated Values = kommagetrennte Werte) und FLF (Fixed-Length Field = Felder mit fester Länge)
- Datenbanken: Alle gebräuchlichen relationalen Datenbanken
- Binärdateien (BLOB-Rohinhalte)

MapForce Enterprise Edition

- XML und XML-Schema
- Flat Files, einschließlich der Formate CSV (Comma Separated Values = kommagetrennte Werte) und FLF (Fixed-Length Field = Felder mit fester Länge)
- Daten aus Altdatenbeständen aus Textdateien können mittels MapForce FlexText auf andere Formate gemappt und konvertiert werden
- SQL-Datenbanken: Alle gebräuchlichen relationalen Datenbanken
- NoSQL-Datenbanken
- Binärdateien (BLOB-Rohinhalte)
- EDI-Standards
- JSON-Dateien
- Microsoft Excel-Dateien ab Version 2007
- XBRL-Instanzdateien und -Taxonomien

- Protocol Buffer
- Auf in PDF Extractor erstellten PDF-Vorlagen basierende PDF-Dateien (können nur als Datenquelle verwendet werden)

1.2.2 Mapping-Szenarien

Altova Website: MapForce-Videodemos

Je nach Geschäftsanforderung können Ihre Mappings unterschiedlich komplex sein: So müssen Sie etwa ein Mapping konfigurieren, in dem Daten aus einer Quellkomponente ausgelesen und in mehrere Zielkomponenten geschrieben werden oder in dem Daten aus mehreren Quellkomponenten in einer Zielkomponente zusammengeführt werden. Als Quellen und Ziele können unterschiedliche Datenstrukturen verwendet werden, z.B. XML-Dateien, Datenbanken, EDI-Dateien, usw. Nähere Informationen zu unterstützten Datenformaten finden Sie unter Mapping: Quellen und Ziele 15.

Die folgenden Szenarien sind nur einige Beispiele für die Komplexität von Mapping-Designs:

- Mappen einer Quellkomponente auf eine Zielkomponente. Nähere Informationen dazu finden Sie im Tutorial 1 97.
- Zusammenführen mehrerer Datenquellen in einer Zielstruktur. Nähere Informationen dazu finden Sie im Tutorial 2 106.
- Mappen von Daten aus einer Quellkomponente auf die erste Zielkomponente, anschließende Filterung der Daten, sodass nur eine Teilmenge dieser Daten auf die zweite Zielkomponente gemappt wird. Siehe Tutorial 3 111.
- Mappen mehrere Quellkomponenten auf mehrere Zielkomponenten. Siehe Tutorial 4 120.

Unabhängig davon, mit welcher Technologie Sie arbeiten, ermittelt MapForce die Struktur Ihrer Daten normalerweise automatisch oder schlägt vor, dass Sie ein Schema für Ihre Daten bereitstellen. MapForce kann auch anhand einer Beispielinstanzdatei ein Schema generieren. Wenn Sie z.B. eine XML-Instanzdatei, aber keine Schemadefinition haben, kann MapForce eine für Sie generieren. Auf diese Art kann MapForce die Daten aus der XML-Datei für das Mappen auf andere Dateien oder Formate zur Verfügung stellen. Nähere Informationen über die Grundbegriffe und wichtigsten Funktionen von MapForce finden Sie unter Grundlegende Aufgaben 29 und Übersicht über die Benutzeroberfläche 19.

Projekte (Professional und Enterprise Edition)
Um Ihre Datenmapping-Designs leichter aufrufen und verwalten zu können, können Sie diese In Mapping-Projekten organisieren. Code kann nicht nur für einzelne Mappings in Ihrem Projekt, sondern auch für ganze Projekte generiert werden.

1.2.3 Transformationssprachen

Mit Hilfe einer Transformationssprache wird in MapForce Transformationscode zur Ausführung von Mappings generiert. Sie können eine Transformationssprache jederzeit auswählen/wechseln. Sie können Programmcode mit dem Menübefehl **Datei | Code generieren in** oder **Datei | Code in ausgewählter Sprache generieren** generieren und mit diesem Code Datentransformationscode außerhalb von MapForce ausführen. Nähere Informationen dazu finden Sie unter Codegenerierung 65.

Je nach MapForce Edition stehen die folgenden Sprachen für Ihre Datentransformationen zur Verfügung:

 © 2019-2025 Altova GmbH

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

Wenn Sie als Transformationssprache XSLT 1-3 oder XQuery auswählen, können Sie den Transformationscode in einem separaten Fenster von MapForce anzeigen.

Zur Auswahl der Transformationssprache haben Sie folgende Möglichkeiten:

- Klicken Sie im Menü **Ausgabe** auf den Namen der Sprache, die Sie für die Transformation verwenden möchten.
- Klicken Sie in der **Sprachauswahl**-Symbolleiste (*Abbildung unten*) auf den Namen der Sprache.

Wenn Sie die Transformationssprache des Mappings wechseln, kann es vorkommen, dass bestimmte MapForce-Funktionalitäten für diese Sprache nicht unterstützt werden. Nähere Informationen dazu finden Sie unter Anmerkungen zur Unterstützung 525.

MapForce validiert bei der Erstellung von Mappings bzw. bei der Erstellung der Mapping-Vorschau die Gültigkeit Ihrer Schemas und Transformationen. Wenn Validierungsfehler auftreten, werden diese von MapForce im Fenster "Meldungen" 24 angezeigt. Dies ist hilfreich, weil Sie die Fehler dadurch sofort überprüfen und korrigieren können.

BUILT-IN
Bei Auswahl der Option Built-In als Transformationssprache wird zum Ausführen des von Mappings der native MapForce-Transformationsprozessor verwendet. Bei Anzeige einer Vorschau auf ein Mapping, in dem als Transformationssprache Java, C# oder C++ ausgewählt ist, verwendet MapForce diese Option auch implizit.

Der Built-In-Prozessor verarbeitet Mappings, ohne dafür externe Prozessoren zu verwenden, was vor allem dann ratsam ist, wenn der Arbeitsspeicher knapp bemessen ist. Wenn kein Programmcode in einer bestimmten Sprache generiert werden muss, verwenden Sie Built-In als die Standardoption, da diese im Vergleich zu anderen Sprachen die meisten MapForce-Funktionalitäten unterstützt (siehe Anmerkungen zur Unterstützung 525. Außerdem können Mappings bei Auswahl von Built-In als Transformationssprache mit MapForce Server automatisiert werden. Nähere Informationen dazu finden Sie unter Automatisierung mit Altova-Produkten 427.

1.2.4 Integration mit Altova-Produkten

Mit Hilfe der integrierten XSLT/XQuery-Prozessoren können Transformationen innerhalb von MapForce durchgeführt werden. MapForce kann auch in Kombination mit anderen Altova-Produkten verwendet werden (*siehe unten*).

XMLSpy
Wenn XMLSpy auf demselben Rechner installiert ist, können Sie alle unterstützten Dateitypen direkt vom jeweiligen MapForce-Kontext aus in XMLSpy öffnen und bearbeiten. So steht z.B. bei Klick auf eine XML-Komponente der Menübefehl **Komponente | Schema-Definition in XMLSpy bearbeiten** zur Verfügung.
RaptorXML Server
Sie können den generierten XSLT-Code direkt in MapForce ausführen und dort sofort eine Vorschau des Ergebnisses der Datentransformation anzeigen. Um die Verarbeitung schneller auszuführen, können Sie das Mapping auch mit RaptorXML Server, einem ultraschnellen XML-Transformationsprozessor, ausführen.

MapForce Server (Enterprise und Professional Edition)
Mit Hilfe von Altova MapForce Server, der auf Windows-, Linux- und macOS-Betriebssystemen installiert werden kann, lassen sich MapForce-Aufgaben automatisieren. Mit MapForce Server können Sie die in einem Mapping definierten Transformationen nicht nur über die Befehlszeile des jeweiligen Betriebssystems, sondern auch über API-Aufrufe (.NET, COM, Java) ausführen.
FlowForce Server (Enterprise und Professional Edition)
Mit Hilfe von Altova FlowForce Server, der auf Windows-, Linux- und macOS-Betriebssystemen installiert werden kann, lassen sich MapForce-Aufgaben ebenfalls automatisieren. Mit FlowForce Server können MapForce Server-Aufgaben nach einem Zeitplan ausgeführt werden.
StyleVision (Enterprise und Professional Edition)
Mit Hilfe von StyleVision können Sie StyleVision Power Stylesheets erstellen oder vorhandene wiederverwenden und eine Vorschau des Ergebnisses der Mapping-Transformationen als HTML-, RTF-, PDF- oder Word 2007+-Dokumente anzeigen.
MapForce als Plug-in
Die MapForce Professional und Enterprise Edition können als Plug-in der IDEs Visual Studio und Eclipse installiert werden. Auf diese Art können Sie Mappings erstellen und erhalten Zugriff auf die MapForce-Funktionalitäten, ohne die Entwicklungsumgebung Ihrer Wahl verlassen zu müssen.

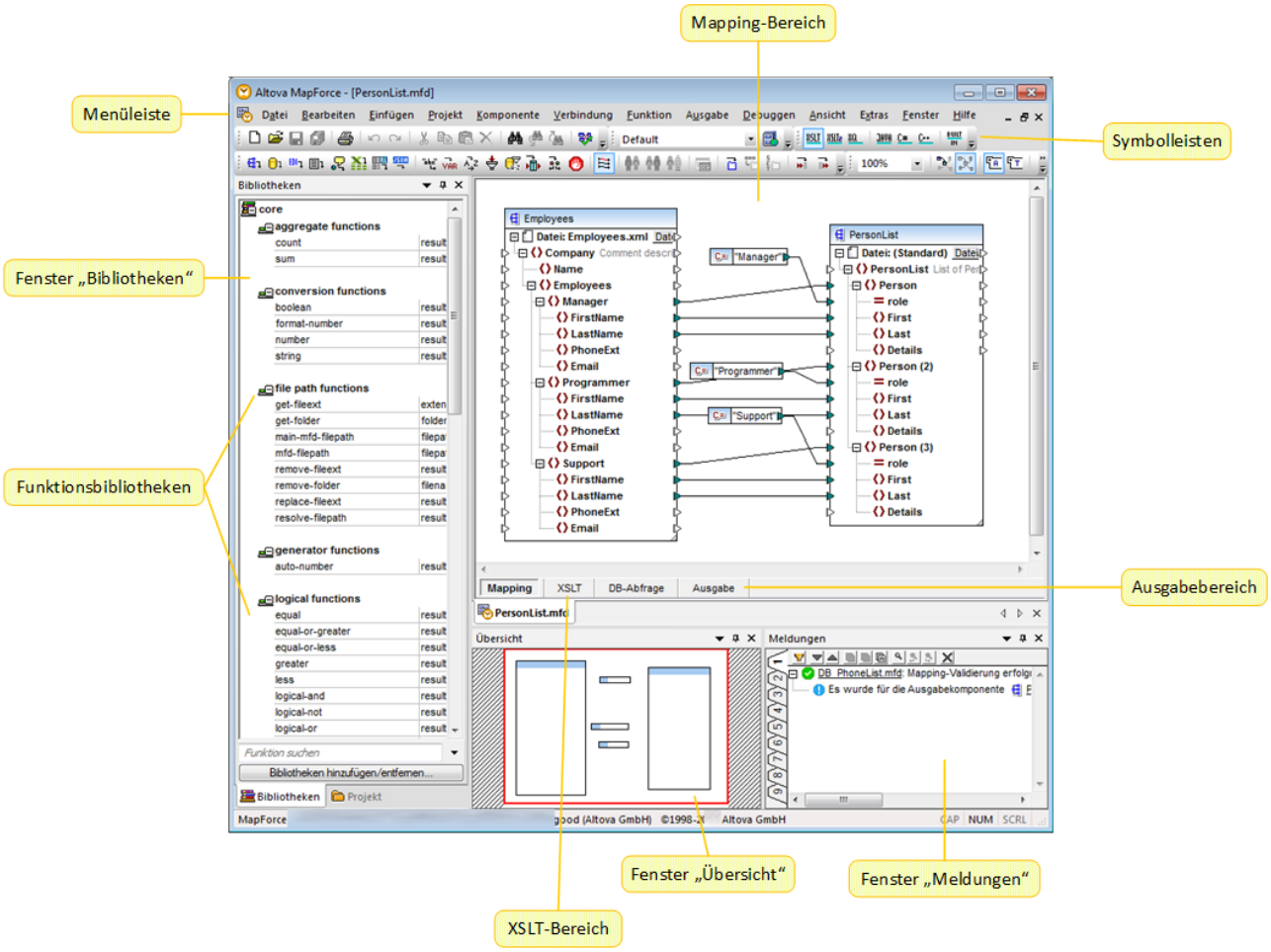
Nähere Informationen zur Automatisierung von Aufgaben finden Sie unter Automatisieren von MapForce-Aufgaben mit Altova-Produkten 427.

 © 2019-2025 Altova GmbH

1.3 Übersicht über die Benutzeroberfläche

Die grafische Benutzeroberfläche von MapForce ist als integrierte Entwicklungsumgebung konzipiert. In der Abbildung unten sehen Sie die wichtigsten Komponenten der Benutzeroberfläche. Mit Hilfe des Menübefehls **Extras | Anpassen** können Sie die Einstellungen der Benutzeroberfläche ändern. Mit Hilfe der Schaltflächen ▼ 📌 ✕ in der rechten oberen Ecke jedes Fensters können Sie das Fenster ein- und ausblenden sowie ab- und andocken. Um die Symbolleisten und Fenster wieder in Ihren ursprünglichen Zustand zurückzusetzen, verwenden Sie den Menübefehl **Extras | Symbolleisten und Fenster wiederherstellen**.

In der nachstehenden Abbildung sind die wichtigsten Bereiche der grafischen Benutzeroberfläche von MapForce dargestellt.



Nähere Informationen über die Features und Funktionen der einzelnen Bereiche finden Sie in den entsprechenden Kapiteln weiter unten.

- Leisten 20
- Fenster 20
- Fenster "Meldungen" 24
- Bereiche 25

1.3.1 Leisten

In diesem Kapitel finden Sie eine Übersicht über die verfügbaren Leisten.

Menüleiste und Symbolleisten

In der **Menüleiste** werden die Menübefehle angezeigt. Jede Symbolleiste enthält eine Reihe von Schaltflächen für verschiedene MapForce-Befehle. Sie können die Symbolleisten an ihren Ziehpunkten mit der Maus an die gewünschte Stelle ziehen. In der Abbildung unten sehen Sie die **Menüleiste** und die Symbolleisten. Was genau auf der Benutzeroberfläche angezeigt wird, hängt von Ihrer MapForce Edition und den gewünschten Einstellungen ab.

Applikationsstatusleiste

Die Applikationsstatusleiste wird am unteren Rand des MapForce-Fensters angezeigt und enthält Informationen zur Applikation. Wenn Sie die Maus über eine Symbolleisten-Schaltflächen platzieren, werden Tooltips dazu angezeigt. Wenn Sie die 64-Bit-Version von MapForce verwenden, wird in der Statusleiste der Applikationsname mit dem Suffix x64 angezeigt. Die 32-Bit-Version hat kein Suffix.

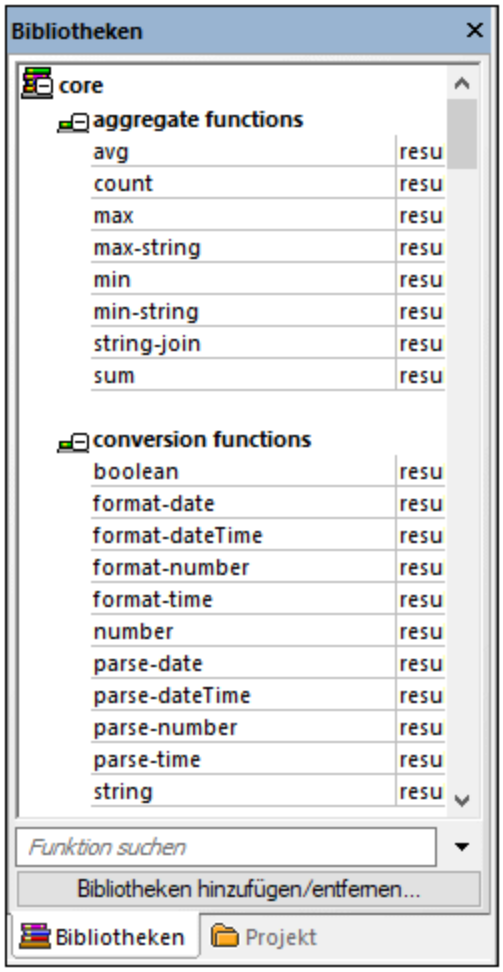
1.3.2 Fenster

In diesem Kapitel finden Sie eine Übersicht über die verfügbaren Fenster.

Fenster "Bibliotheken"

Im Fenster **Bibliotheken** wird eine Liste der vordefinierten MapForce-Funktionen, geordnet nach Bibliotheken, angezeigt. Je nachdem, welche Transformationssprache Sie entweder über das Menü **Ausgabe** oder über die **Sprachauswahl**-Symbolleiste auswählen, stehen unterschiedliche Funktionen zur Verfügung. Nähere Informationen dazu finden Sie unter Transformationssprachen 16. Wenn Sie benutzerdefinierte Funktionen erstellt oder externe Bibliotheken importiert haben, so werden auch diese im Fenster **Bibliotheken** angezeigt.

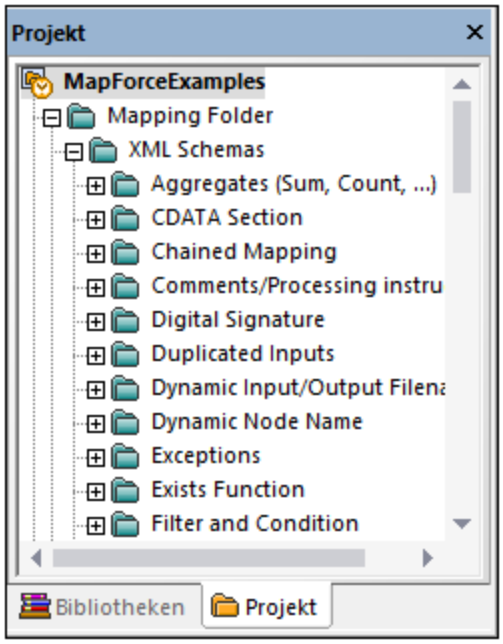
 © 2019-2025 Altova GmbH



Um eine Funktion nach Name oder Beschreibung zu suchen, geben Sie den Suchwert in das Textfeld am unteren Rand des Fensters **Bibliotheken** ein. Um alle Instanzen einer Funktion (im gerade aktiven Mapping) zu suchen, klicken Sie mit der rechten Maustaste auf die Funktion und wählen Sie im Kontextmenü den Befehl **Alle Aufrufe suchen** aus. Sie können den Datentyp der Funktion und ihre Beschreibung auch direkt im Fenster **Bibliotheken** anzeigen. Nähere Informationen dazu finden Sie unter Funktionen 212.

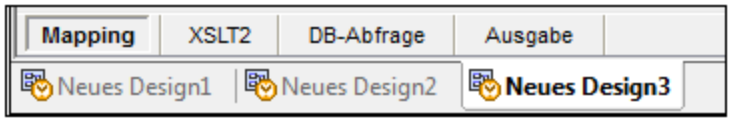
Projektfenster (Enterprise und Professional Edition)

MapForce unterstützt das Multiple Document Interface und gestattet Ihnen, Ihre Mappings in Mapping-Projekten zu gruppieren. Im **Projektfenster** sehen Sie alle Dateien und Ordner, die zum Projekt hinzugefügt wurden. Projektdateien haben die Dateierweiterung **`*.mfp`** (MapForce-Projekt). Um in Projekten nach Mappings zu suchen, klicken Sie an eine beliebige Stelle im **Projektfenster** und drücken Sie **Strg + F**.



Mapping-Fenster

In MapForce wird ein Multiple Document Interface (MDI) verwendet. Jede in MapForce geöffnete Mapping-Datei hat ein eigenes Fenster. Auf diese Art können Sie mit mehreren Mapping-Fenstern arbeiten und diese auf verschiedene Arten im Hauptfenster von MapForce anordnen und in der Größe anpassen. Sie können alle offenen Fenster auch nach den Windows-Standard-Layouts "Horizontal anordnen", "Vertikal anordnen" und "Überlappend" anordnen. Wenn mehrere Mappings in MapForce geöffnet sind, können Sie über die im unteren Bereich des **Mapping**-Fensters angezeigten Register jederzeit zwischen den Mappings wechseln (*siehe Abbildung unten*).



Die Optionen zur Verwaltung der Fenster können über den Menübefehl **Fenster | Fenster** aufgerufen werden. Über das Dialogfeld **Fenster** können Sie an jedem bzw. allen der derzeit offenen Mapping-Fenstern Aktionen wie Speichern, Schließen oder Minimieren vornehmen. Um mehrere Fenster im Dialogfeld **Fenster** auszuwählen, klicken Sie auf die gewünschten Einträge, während Sie die **Strg**-Taste gedrückt halten.

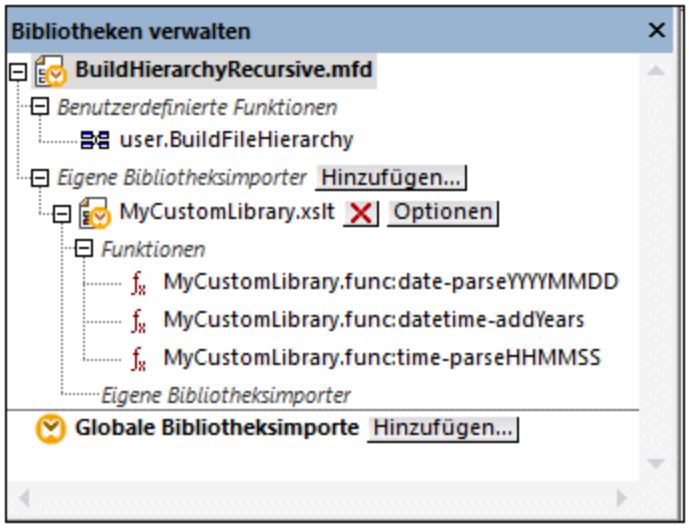
Fenster "Bibliotheken verwalten"

Über dieses Fenster können Sie alle benutzerdefinierten Funktionen (UDFs = user-defined functions) und importierten benutzerdefinierten Bibliotheken, die in derzeit geöffneten Mappings verwendet werden, anzeigen und verwalten.

Standardmäßig wird das Fenster **Bibliotheken verwalten** nicht angezeigt. Um es anzuzeigen, wählen Sie eine der folgenden Methoden:

 © 2019-2025 Altova GmbH

- Klicken Sie im Menü **Ansicht** auf **Bibliotheken verwalten**.
- Klicken Sie im unteren Bereich des Fensters **Bibliotheken** auf die Schaltfläche **Bibliotheken hinzufügen/entfernen**.



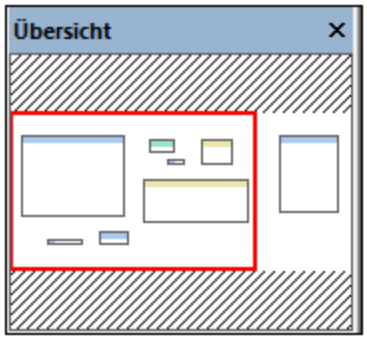
Sie können auswählen, ob benutzerdefinierte Funktionen (UDFs) und Bibliotheken nur für das gerade aktive Mapping-Dokument oder für alle geöffneten Mapping-Dokumente angezeigt werden sollen. Um die importierten Funktionen und Bibliotheken für alle gerade offenen Mapping-Dokumente anzuzeigen, klicken Sie mit der rechten Maustaste in das Fenster und wählen Sie im Kontextmenü den Befehl **Offene Dokumente anzeigen**.

Um anstelle des Namens den Pfad des geöffneten Mapping-Dokuments anzuzeigen, klicken Sie mit der rechten Maustaste in das Fenster und wählen Sie im Kontextmenü den Befehl **Dateipfade anzeigen**.

Nähere Informationen dazu finden Sie unter Verwalten von Funktionsbibliotheken. 215.

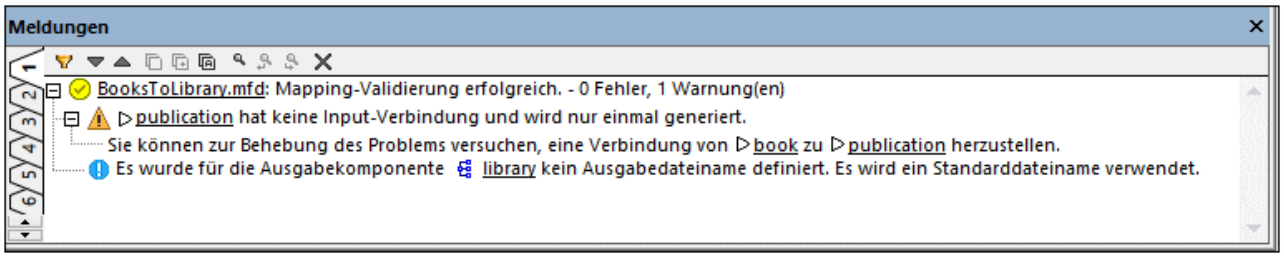
Fenster "Übersicht"

Im Fenster **Übersicht** sehen Sie eine Gesamtübersicht über den Mapping-Bereich 25. Wenn das Mapping sehr groß ist, können Sie hier schnell zu einer bestimmten Stelle im Mapping-Bereich navigieren. Klicken Sie dazu auf das rote Rechteck und ziehen Sie es an die gewünschte Stelle.



1.3.3 Fenster "Meldungen"

Im Fenster **Meldungen** (*siehe Abbildung unten*) werden bei der Mapping-Vorschau oder bei Durchführung einer Mapping-Validierung 63 der Validierungsstatus, Meldungen, Fehler und/oder Warnungen angezeigt. Klicken Sie auf den unterstrichenen Text im Fenster **Meldungen**, um zur Komponente bzw. Struktur zu gelangen, die die Information, Warnung oder Fehlermeldung verursacht hat.



Symbole für den Validierungsstatus

Bei der Validierung eines Mappings überprüft MapForce auf nicht unterstützte Komponentenarten, falsche oder fehlende Verbindungen. Das Validierungsergebnis wird im **Fenster Meldungen** mit einem der folgenden Statussymbole angezeigt:

Symbol	Bedeutung
	Die Validierung war erfolgreich.
	Die Validierung wurde abgeschlossen. Es wurden Warnungen ausgegeben.
	Die Validierung ist fehlgeschlagen.

Zusätzlich dazu werden im Fenster **Meldungen** eventuell die folgenden Arten von Meldungen angezeigt: Informationen, Warnungen und Fehler.

Symbol	Bedeutung
	Kennzeichnet eine Informationsmeldung. Bei Informationsmeldungen wird die Mapping-Ausführung nicht gestoppt.
	Kennzeichnet eine Warnmeldung. Bei Warnungen wird die Mapping-Ausführung nicht gestoppt. Sie werden z.B. angezeigt, wenn keine Verbindungen zu obligatorischen Input-Konnektoren erstellt wurden. In solchen Fällen wird für diejenigen Komponenten, die eine gültige Verbindung haben, dennoch eine Ausgabe generiert.
	Kennzeichnet einen Fehler. Bei einem Fehler schlägt die Mapping-Ausführung fehl und es wird keine Ausgabe generiert. Es kann keine Vorschau auf den XSLT- oder XQuery-Code generiert werden.

Um die Komponente oder Struktur, die die Informations-, Warn- oder Fehlermeldung verursacht hat, zu markieren, klicken Sie im Fenster **Meldungen** auf den unterstrichenen Text.

 © 2019-2025 Altova GmbH

Aktionen im Zusammenhang mit Meldungen

Im Fenster **Meldungen** können die folgenden Aktionen durchgeführt werden:

Symbol	Beschreibung
	Filtert Meldungen nach ihrem Schweregrad (Informationsmeldungen, Fehler, und Warnungen). Wählen Sie **Alle aktivieren,** um alle Schweregrade zu inkludieren (dies ist die Standardeinstellung). Wählen Sie **Alle deaktivieren,** um alle Schweregrade aus dem Filter zu entfernen. In diesem Fall wird nur eine Meldung über den allgemeinen Ausführungs- oder Validierungsstatus angezeigt.
	Geht zur nächsten Zeile.
	Geht zur vorherigen Zeile.
	Kopiert die ausgewählte Zeile in die Zwischenablage.
	Kopiert die ausgewählte Zeile einschließlich aller untergeordneten Zeilen in die Zwischenablage.
	Kopiert den gesamten Inhalt des Fensters **Meldungen** in die Zwischenablage.
	Sucht im Fenster **Meldungen** nach einem bestimmten Text. Um optional nur nach Wörtern zu suchen, wählen Sie die Option **Ganzes Wort**. Um bei der Textsuche die Groß- und Kleinschreibung zu berücksichtigen, wählen Sie die Option **GROSS/klein beachten**.
	Sucht ab der aktuell ausgewählten Zeile bis zum Ende nach einem bestimmten Text.
	Sucht ab der aktuell ausgewählten Zeile bis zum Anfang nach einem bestimmten Text.
	Löscht die Meldungen im Fenster "Meldungen".

Wenn Sie gleichzeitig mit mehreren Mapping-Fenstern arbeiten, ist es sinnvoll, die Informationen, Warnungen und Fehlermeldungen für jedes Mapping auf einem eigenen Register anzuzeigen. Klicken Sie in diesem Fall auf die nummerierten Register auf der linken Seite des Fensters **Meldungen**, bevor Sie das Mapping validieren.

1.3.4 Bereiche

In diesem Kapitel finden Sie eine Übersicht über die verfügbaren Bereiche.

Mapping-Bereich

Der **Mapping**-Bereich ist der Arbeitsbereich, in dem Sie Mappings 63 erstellen. Über das Menü **Einfügen** können Sie Mapping-Komponenten (wie z.B. Dateien, Schemas, Konstanten, Variablen usw.) zum Mapping-Bereich hinzufügen. Nähere Informationen dazu finden Sie unter Hinzufügen von Komponenten zum Mapping 35. Sie können auch Funktionen aus dem Fenster **Bibliotheken** in den **Mapping**-Bereich ziehen. Nähere Informationen dazu finden Sie unter Hinzufügen einer Funktion zum Mapping 212.

XSLT-Bereich

Im **XSLT**-Bereich wird der anhand Ihres Mappings generierte XSLT-Transformationscode angezeigt. Um zu diesem Fenster zu wechseln, wählen Sie als Transformationssprache 16 XSLT, XSLT2 oder XSLT3 aus und klicken Sie auf das Register mit dem entsprechenden Namen.

Dieses Fenster bietet Funktionalitäten zur Anzeige von Zeilennummern und zum Ein- und Ausklappen von Codeabschnitten. Um Codeabschnitte ein- oder auszuklappen, klicken Sie auf das "+" bzw."-"-Symbol am linken Fensterrand. Eingeklappte Codeabschnitte werden mittels Auslassungspunkten markiert. Um eine Vorschau des eingeklappten Abschnitts zu sehen, ohne diesen Abschnitt ausklappen zu müssen, platzieren Sie die Mauszeiger über die Auslassungspunkte. Daraufhin wird ein Tooltip mit der Codevorschau angezeigt, wie in der Abbildung unten gezeigt. Wenn der Abschnitt zu groß für die Vorschau ist, wird am Ende des Tooltips ein weiteres Auslassungssymbol angezeigt.

```
1   <?xml version="1.0" encoding="UTF-8"?>
2   <!--
3   This file was generated by Altova MapForce 2021r3
4
5   YOU SHOULD NOT MODIFY THIS FILE, BECAUSE IT WILL BE
6   OVERWRITTEN WHEN YOU RE-RUN CODE GENERATION.
7
8   Refer to the Altova MapForce Documentation for further details.
9   http://www.altova.com/mapforce
10  -->
11  <xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/
    Transform" xmlns:xs="http://www.w3.org/2001/XMLSchema" exclude-result-
    prefixes="xs">
12      <xsl:output method="xml" encoding="UTF-8" indent="yes"/>
13      <xsl:template match="/">
14          <xsl:variable name="var1_initial" select="."/>
15          <PersonList>
16              <xsl:attribute name="xsi:noNamespaceSchemaLocation"
    namespace="http://www.w3.org/2001/XMLSchema-instance">file:///C:/Users/
    anf/Documents/Altova/MapForce2021/MapForceExamples/PersonList.xsd</
    xsl:attribute>
17              <xsl:for-each select="(./Company/Employees/node())[./
    self::*]">...</xsl:for-each>
31          </PersonList>
32      </xsl
33  </xsl:sty
34
```

```
<xsl:variable name="var2_filter" select="."/>
<Person>
    <xsl:attribute name="role">
        <xsl:value-of select="local-name(.)"/>
    </xsl:attribute>
    <First>
        <xsl:value-of select="FirstName"/>
    </First>
    <Last>
        <xsl:value-of select="LastName"/>
    </Last>
</Person>
```

Mapping | XSLT

PersonList.mfd

Übersicht

Um die Anzeigeeinstellungen einschließlich Einrückung, Zeilenendemarkierungen und anderen zu konfigurieren, klicken Sie mit der rechten Maustaste in den Bereich und wählen Sie im Kontextmenü den

© 2019-2025 Altova GmbH

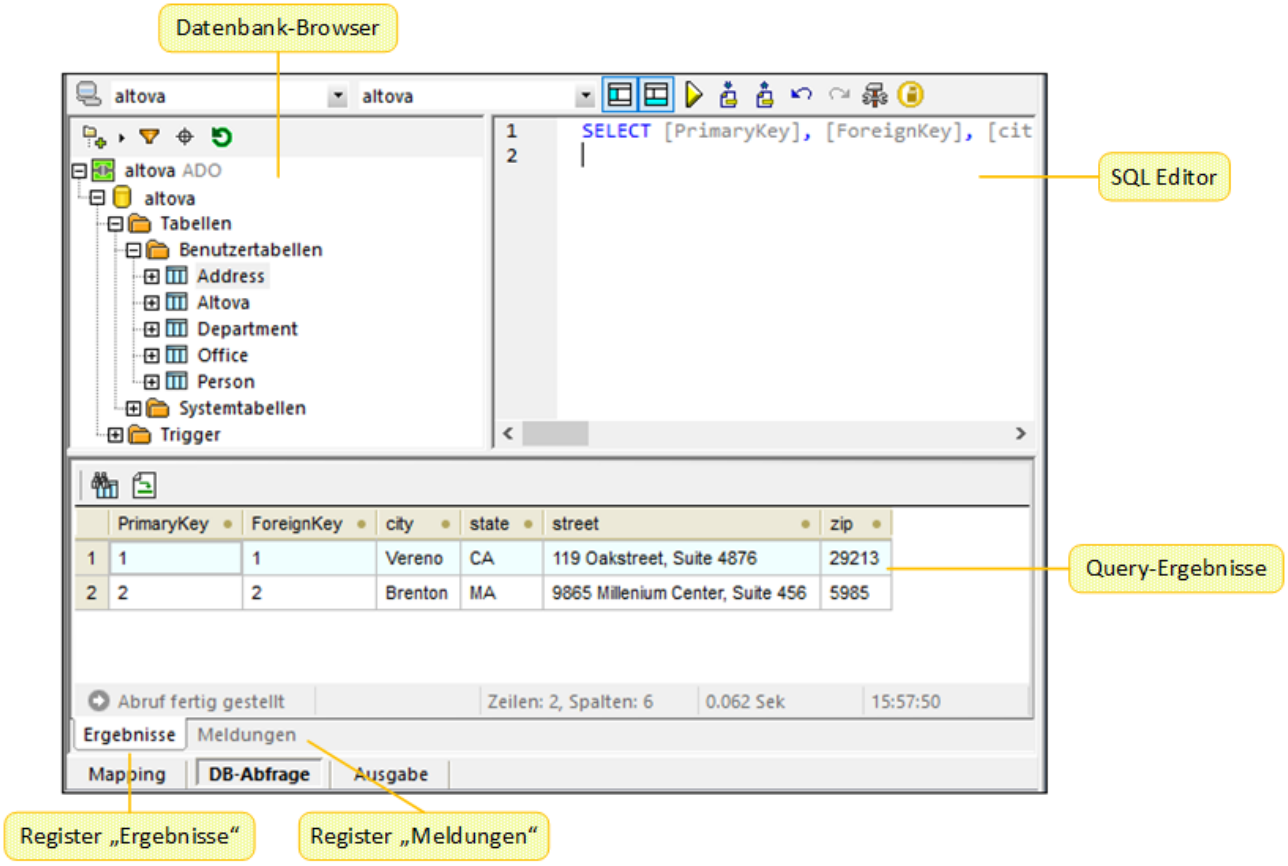
Befehl **Einstellungen für Textansicht** aus. Alternativ dazu können Sie auch auf die Symbolleisten-Schaltfläche (**Einstellungen für Textansicht**) klicken.

XQuery-Bereich (Enterprise und Professional Edition)

Im **XQuery**-Bereich wird der anhand Ihres Mappings generierte XQuery-Transformationscode angezeigt, wenn Sie auf die Schaltfläche **XQuery** klicken. Dieser Bereich steht zur Verfügung, wenn Sie XQuery als Transformationssprache auswählen. Dieser Bereich enthält auch eine Zeilennummerierung und eine Klappleiste, die ähnlich wie im Bereich "XSLT" (siehe oben) funktioniert.

Bereich "DB-Abfrage" (Enterprise und Professional Edition)

Im Bereich **DB-Abfrage** können Sie Abfragen an allen gebräuchlichen Datenbanken durchführen. Sie können mit mehreren aktiven Verbindungen zu verschiedenen Datenbanken arbeiten.



Fenster "Ausgabe"

Im Fenster **Ausgabe** sehen Sie das Ergebnis der Mapping-Transformation. Wenn beim Mapping mehrere Dateien generiert werden, können Sie der Reihe nach durch die einzelnen generierten Dateien navigieren.

```
1   <?xml version="1.0" encoding="UTF-8"?>
2   <PersonList xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="file:///C:/Users/anf/Documents/Altova/
    MapForce2021/MapForceExamples/PersonList.xsd">
3       <Person role="Manager">
4           <First>Vernon</First>
5           <Last>Callaby</Last>
6       </Person>
7       <Person role="Programmer">
8           <First>Frank</First>
9           <Last>Further</Last>
10      </Person>
11      <Person role="Support">
12          <First>Loby</First>
13          <Last>Matise</Last>
14      </Person>
15      <Person role="Support">
16          <First>Susi</First>
17          <Last>Sanna</Last>
18      </Person>
19  </PersonList>
```

Mapping | XSLT | DB-Abfrage | **Ausgabe**

PersonList.mfd

Dieses Fenster enthält auch Zeilennummerierung und eine Klappleiste, die ähnlich wie im Fenster "XSLT" (siehe oben) funktioniert.

StyleVision-Ausgabebereiche (Enterprise und Professional Edition)

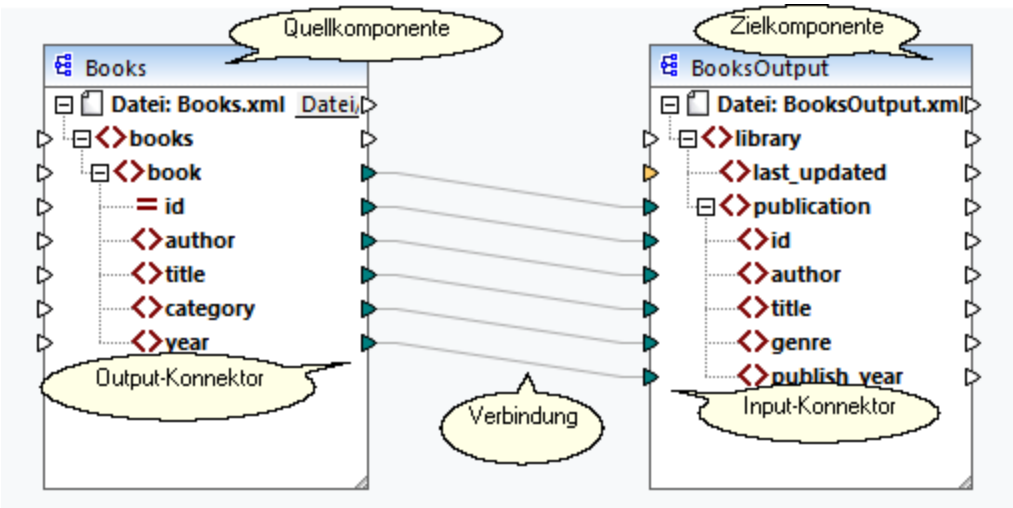
Wenn Sie Altova StyleVision installiert haben, stehen neben dem Fenster **Ausgabe** die StyleVision-Ausgabebereiche zur Verfügung. In den StyleVision-Ausgabebereichen können Sie eine Vorschau der Mapping-Ausgabe in HTML, RTF, PDF und Word 2007+ anzeigen und speichern. Dazu werden in StyleVision StyleVision Power Stylesheet (SPS)-Dateien erstellt und in MapForce einer Mapping-Komponente zugewiesen.

© 2019-2025 Altova GmbH

2 Mapping-Grundlagen

Ein MapForce Mapping-Design (oder "Mapping") ist eine visuelle Darstellung davon, wie Daten von einem Format in ein anderes transformiert werden sollen. Ein Mapping besteht aus *Komponenten*, die zum Mapping-Bereich hinzugefügt werden, um Datentransformationen zu erstellen. Ein gültiges Mapping besteht aus einer oder mehreren *Quellkomponenten*, die mit einer oder mehreren *Zielkomponenten* verbunden werden. Sie können ein Mapping ausführen und direkt in MapForce eine Vorschau auf das Ergebnis anzeigen. Sie können Code generieren und ihn extern ausführen. Sie können ein Mapping auch zu einer MapForce-Ausführungsdatei kompilieren und die Ausführung des Mappings mit MapForce Server oder FlowForce Server automatisieren. MapForce speichert Mappings als `.mfd`-Dateien.

In der Abbildung unten sehen Sie die grundlegende Struktur eines Mappings:



Neues Mapping

Um ein neues Mapping zu erstellen, klicken Sie in der Symbolleiste auf die Schaltfläche (**Neu**). Klicken Sie alternativ dazu im Menü **Datei** auf **Neu**. Wir werden als nächstes Komponenten zum Mapping hinzufügen 35 und Verbindungen 45 zwischen den Komponenten erstellen.

Hauptbereiche eines Mappings

In den folgenden Unterabschnitten werden die Hauptbereiche eines Mapping-Designs beschrieben.

Komponente

Als *Komponente* wird in MapForce die visuelle Darstellung der Struktur Ihrer Daten bzw. die Art, wie die Daten transformiert werden sollen, bezeichnet. Komponenten bilden die zentralen Bestandteile eines jeden Mappings und werden in Form von rechteckigen Kästen dargestellt. Komponenten können in zwei große Gruppen unterteilt werden:

- Quell- und Zielkomponenten
- Struktur- 129 und Transformationskomponenten 163

Beachten Sie, dass diese beiden Gruppen einander nicht gegenseitig ausschließen. In der ersten Gruppe werden die Beziehungen zwischen Komponenten dargestellt. So kann etwa eine Komponente eine Quellkomponente für eine Komponente und für eine andere eine Zielkomponente bilden. MapForce liest die

Daten aus einer Quellkomponente aus und schreibt sie in eine Zielkomponente. Bei der Ausführung des Mappings generiert MapForce anhand der Zielkomponente eine (oder mehrere Dateien) oder übergibt das Ergebnis als String-Wert für die weitere Verarbeitung an ein externes Programm. Unten sind die Komponentenarten aus der ersten Gruppe beschrieben:

- Eine *Quellkomponente* befindet sich auf der linken Seite des Mapping-Bereichs. Aus der Quellkomponente werden Daten ausgelesen.
- Eine *Zielkomponente* befindet sich rechts von der Quellkomponente. In die Zielkomponente werden Daten geschrieben.
- Eine *Weiterleitungskomponente* ist ein Subtyp von Quell- und Zielkomponenten. Eine Weiterleitungskomponente fungiert sowohl als Quell- als auch Zielkomponente. Nähere Informationen dazu finden Sie unter Verkettete Mappings 111. Beachten Sie, dass nur Strukturkomponenten als Weiterleitungskomponenten fungieren können.

In der zweiten Gruppe (Struktur-/Transformationskomponenten) wird dargestellt, ob eine Komponente eine Datenstruktur hat oder zum Transformieren von aus einer anderen Komponente gemappten Daten verwendet wird.

Nähere Informationen über Komponenten und Aktionen im Zusammenhang mit Komponenten finden Sie unter Komponenten 31.

Konnektor
Als Konnektor wird das kleine Dreieck auf der linken oder rechten Seite einer Komponente bezeichnet. Über die Input-Konnektoren auf der linken Seite einer Komponente werden Daten *in diese Komponente eingegeben*. Über die Output-Konnektoren auf der rechten Seite einer Komponente werden Daten *aus dieser Komponente* ausgegeben.

Verbindung
Eine Verbindung ist eine Linie, die Sie zwischen zwei MapForce-Konnektoren ziehen. Durch Erstellen von Verbindungen weisen Sie MapForce an, Daten auf eine bestimmte Art zu transformieren: z.B. Daten aus einem XML-Dokument zu lesen und in ein anderes XML-Dokument zu schreiben.

In diesem Abschnitt

In diesem Kapitel werden die häufigsten MapForce-Aufgaben und Konzepte beschrieben. Dieser Abschnitt ist in die folgenden Unterabschnitte gegliedert:

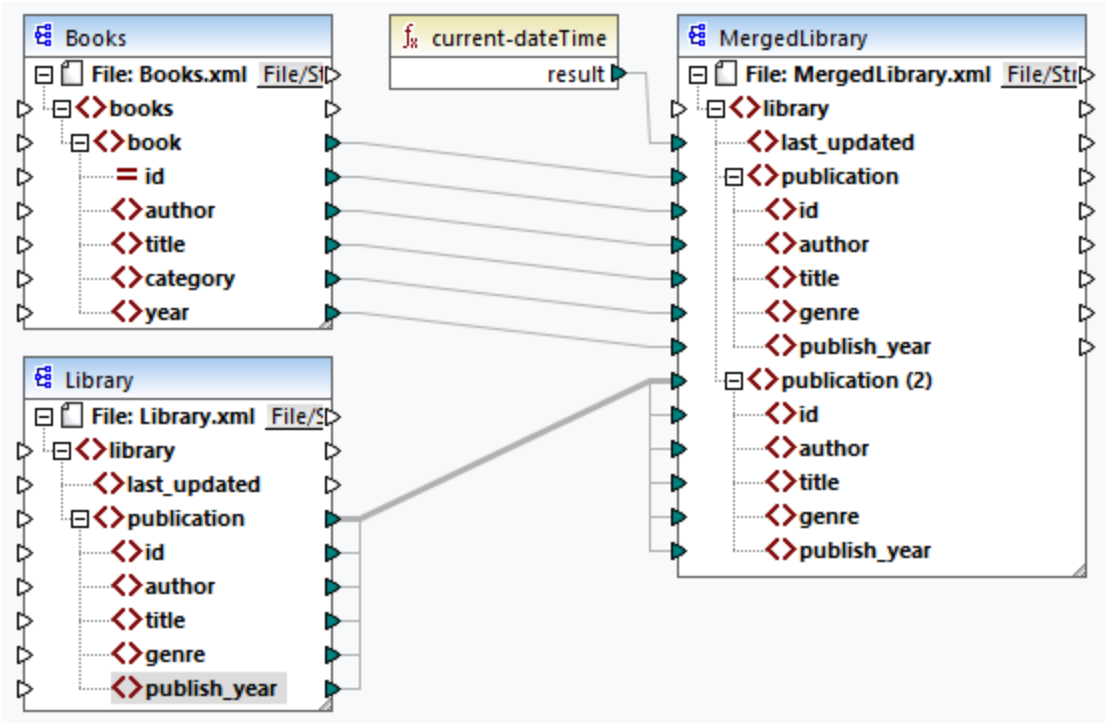
- Komponenten 31
- Verbindungen 45
- Allgemeine Verfahren und Funktionalitäten 63
- Grundlegende Regeln und Strategien 77

 © 2019-2025 Altova GmbH

2.1 Komponenten

Komponenten bilden das zentrale Element eines jeden Mapping-Designs in MapForce. Komponenten werden im Mapping-Bereich als rechteckige Kästen dargestellt. Dieses Kapitel enthält eine Übersicht über Struktur- und Transformationskomponenten (*siehe Beispiel unten*). Der Unterschied besteht darin, ob eine Komponente eine Datenstruktur hat oder zum Transformieren von Daten verwendet wird. Eine Beschreibung dieser beiden Arten finden Sie in den Unterabschnitten weiter unten. Siehe auch Mapping-Grundlagen 29. Neben Struktur- und Transformationskomponenten können Sie auch Kommentare zu Ihrem Mapping hinzufügen (*siehe Kommentare weiter unten*).

Beispiel
Im unten gezeigten Beispiel-Mapping sehen Sie zwei Quellkomponenten (`Books` und `Library`), eine Zielkomponente (`MergedLibrary`) sowie eine Transformationskomponente (die Funktion **`current-dateTime`**).



Strukturkomponenten

Mit einer Strukturkomponente wird eine abstrakte Struktur Ihrer Daten dargestellt (z.B. eine XML-Datei). Unter Strukturkomponenten 129 finden Sie eine Liste von Strukturkomponenten, die als Datenquellen und -ziele verwendet werden können. Mit Hilfe von Strukturkomponenten können Sie Daten aus einer oder mehreren Quellen auslesen, Daten in eine oder mehrere Zielkomponenten schreiben und Daten in einer Zwischenphase des Mappings speichern (z.B. um eine Vorschau der Daten anzuzeigen). Die nachstehende Tabelle enthält eine Übersicht über Strukturkomponenten und die dazugehörigen Symbolleisten-Schaltflächen.

Symbol	Beschreibung
	XML-Komponente

Symbol	Beschreibung
	Textkomponente (*Professional und Enterprise Edition*)
	Datenbankkomponente (*Professional und Enterprise Edition* für SQL-Datenbanken; *Enterprise Edition* für NoSQL-Datenbanken)
	JSON-Komponente (*Enterprise Edition*)
	Microsoft Excel-Komponente (*Enterprise Edition*)
	EDI-Komponente (*Enterprise Edition*)
	XBRL-Komponente (*Enterprise Edition*)
	Protocol Buffer (*Enterprise Edition*)
	PDF (*Enterprise Edition*)

Transformationskomponenten

Mit Hilfe von Transformationskomponenten können Sie Daten transformieren 212, ein Mapping-Zwischenergebnis für die weitere Verarbeitung speichern 175, einen Wert durch einen anderen ersetzen 200 und Ihre Daten sortieren 188, gruppieren 297 und filtern 194. Die nachstehende Tabelle enthält eine Übersicht über Transformationskomponenten und die dazugehörigen Symbolleisten-Schaltflächen.

Symbol	Beschreibung
	Einfache Input-Komponente
	Einfache Output-Komponente
	Filter-Komponente
	Sortierkomponente
	Vordefinierte Funktion
	Benutzerdefinierte Funktion
	Wertezuordnungskomponente
	Variable
	SQL/NoSQL-WHERE/ORDER-Komponente (*Professional und Enterprise Edition*)
	Konstante
	If-Else-Bedingung
	Ausnahme (*Professional und Enterprise Edition*)
	Join-Komponente (*Professional und Enterprise Edition*)

© 2019-2025 Altova GmbH

Symbol	Beschreibung
	Webservice-Aufruf einer HTTP API oder eines WSDL-basierten Service (*Enterprise Edition*)
	Webservice-Aufruf der Shopify API oder einer generischen GraphQL API (*Enterprise Edition*)

Kommentare

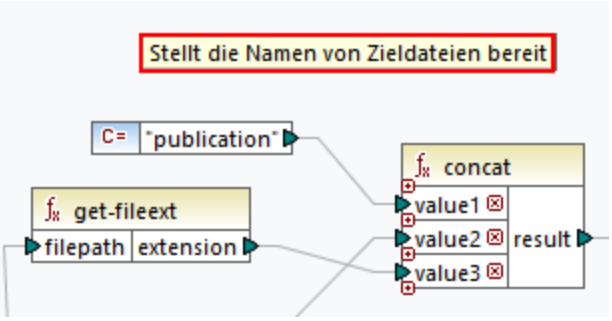
Kommentare können in MapForce als eigenständige Komponenten und als Notizen unterhalb von vorhandenen Komponenten hinzugefügt werden.

Kommentarkomponenten

Kommentarkomponenten sind freistehende Kästchen, in denen mehrzeiliger Text angezeigt werden kann. Diese Komponenten können nicht mit einer anderen Komponente verbunden werden. Wählen Sie eine der folgenden Optionen, um eine Kommentarkomponente hinzuzufügen:

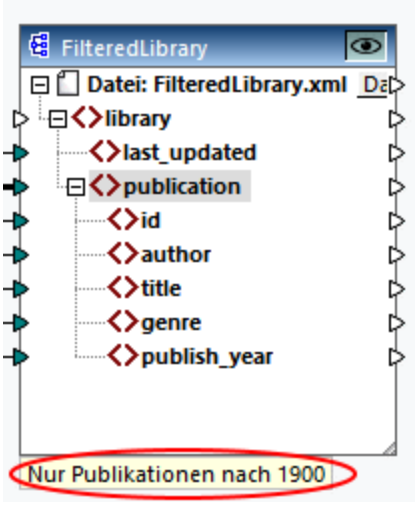
- Wählen Sie den Menübefehl , woraufhin ein Dialogfeld angezeigt wird, in das Sie Ihren Kommentar eingeben können.
- Wählen Sie den Menübefehl **Einfügen | Kommentar**, woraufhin ein Dialogfeld angezeigt wird, in das Sie Ihren Kommentar eingeben können.
- Doppelklicken Sie auf den leeren Bereich Ihres Mappings, geben Sie das Zeichen # und einen Kommentar ein und drücken Sie die **Eingabetaste**. Das Zeichen # wird im Kommentarkästchen nicht angezeigt.

Um einen Kommentar zu verschieben, ziehen Sie sie ihn an die gewünschte Position. Um einen Kommentar zu löschen, klicken Sie darauf und drücken Sie die Taste **Löschen**. Unten sehen Sie ein Beispiel für eine Kommentarkomponente (*rot umrandetes Kästchen*).



Komponentenkommentare

Neben freistehenden Kommentarkästchen können Sie Kommentare auch zu jeder beliebigen Komponente hinzufügen. Solche Kommentare werden unterhalb der Komponente angezeigt (*unten rot umrandet*).



Wählen Sie eine der folgenden Optionen, um einen Kommentar unterhalb einer Komponente hinzuzufügen:

- Klicken Sie mit der rechten Maustaste in die Komponente und wählen Sie im Kontextmenü den Befehl **Kommentar bearbeiten**. Daraufhin wird ein Dialogfeld geöffnet, in das Sie Ihren Kommentar eingeben können.
- Wählen Sie eine Komponente aus, zu der Sie einen Kommentar hinzufügen möchten. Klicken Sie anschließend im Menü **Komponente** auf **Kommentar bearbeiten**. Daraufhin wird ein Dialogfeld geöffnet, in das Sie Ihren Kommentar eingeben können.

Sie können die Anzeige von Komponentenkommentaren über das Menü **Extras | Optionen | Allgemein | Mapping-Ansicht** auf eine bestimmte Anzahl von Zeilen beschränken. Nähere Informationen dazu finden Sie unter Optionen 508.

Um einen Komponentenkommentar zu entfernen, wählen Sie eine der folgenden Methoden:

- Doppelklicken Sie auf den Kommentar, löschen Sie den gesamten Text und drücken Sie die **Eingabetaste**.
- Klicken Sie mit der rechten Maustaste auf den Kommentar oder in die Komponente, wählen Sie im Kontextmenü den Befehl **Kommentar bearbeiten**, löschen Sie den Text und klicken Sie auf **OK**.

Kommentare bearbeiten
Sie können beide Arten von Kommentaren auf die folgenden Arten zum Mapping hinzufügen:

- Doppelklicken Sie auf den Text des Kommentars und beginnen Sie ihn direkt im Kästchen zu bearbeiten. Drücken Sie anschließend die **Eingabetaste**.
- Klicken Sie mit der rechten Maustaste auf das Kommentarkästchen, bearbeiten Sie den Text im Dialogfeld **Kommentar bearbeiten** und klicken Sie auf **OK**. Bei Komponentenkommentaren kann das Dialogfeld **Kommentar bearbeiten** auch durch Rechtsklick in den Komponente und Auswahl der Option **Kommentar bearbeiten** im Kontextmenü aufgerufen werden.

In diesem Abschnitt

Dieser Abschnitt enthält eine Übersicht über Komponenten und ist in die folgenden Kapitel gegliedert:

- Hinzufügen von Komponenten 35

 © 2019-2025 Altova GmbH

- Komponentengrundlagen 38
- Dateipfade 40

2.1.1 Hinzufügen von Komponenten

In diesem Kapitel wird erklärt, wie Sie Komponenten zu einem Mapping hinzufügen. Um eine Komponente hinzuzufügen, müssen Sie zuerst ein neues Mapping-Design erstellen 29. Wählen Sie anschließend eine der folgenden Methoden:

- Wählen Sie im Menü **Einfügen** einen Komponententyp aus (z.B. **XML-Schema/Datei**).
- Ziehen Sie eine Datei aus dem Windows Datei-Explorer in den Mapping-Bereich.
- Klicken Sie in der **Komponente einfügen**-Symbolleiste auf die entsprechende Schaltfläche (*siehe Abbildung unten*).

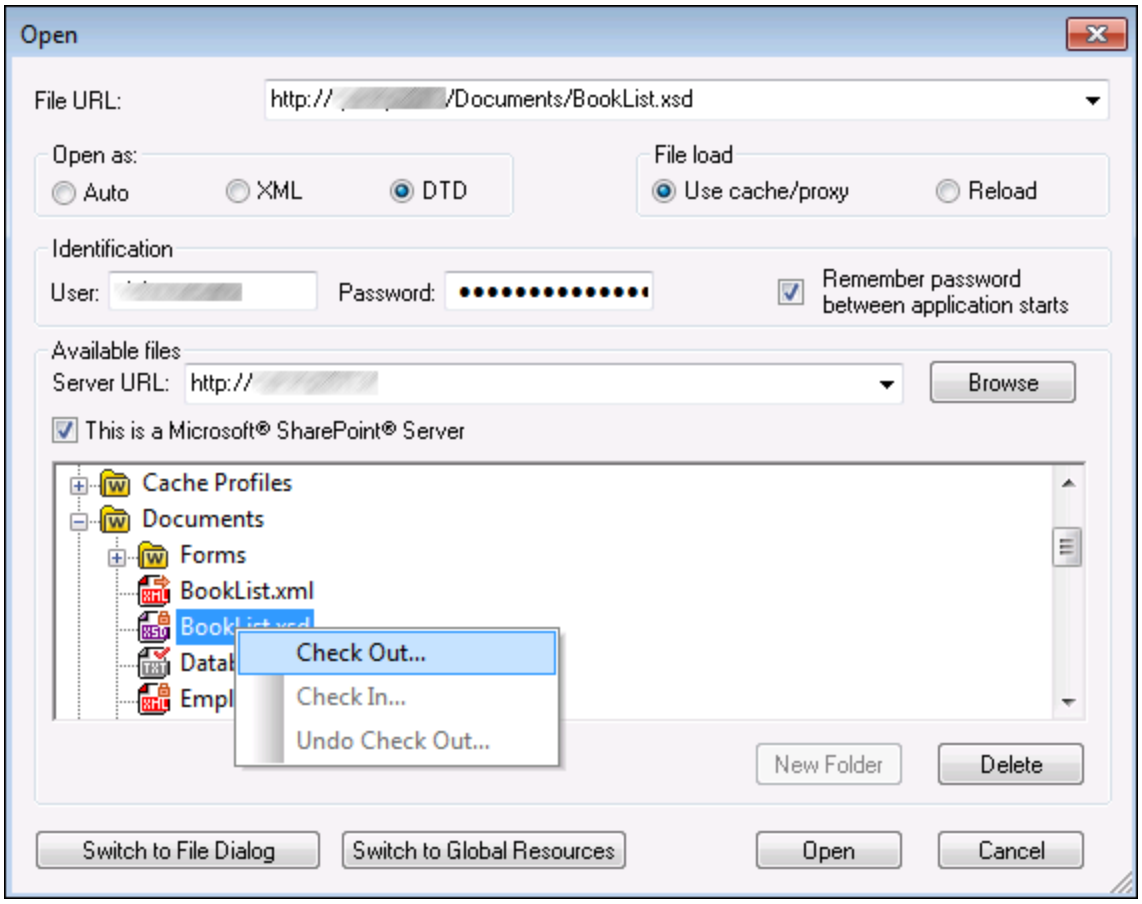
Jede Komponente hat einen bestimmten Zweck und weist ein bestimmtes Verhalten auf. Eine Übersicht über Komponenten finden Sie unter Komponenten 31. Nähere Informationen über Datenstrukturen, die als Quell- und Zielkomponenten verwendet werden können, finden Sie unter Strukturkomponenten 129. Informationen zu MapForce-Komponenten, mit denen Daten temporär gespeichert oder transformiert werden, finden Sie unter Transformationskomponenten 163.

Wenn Sie eine Strukturkomponente zum Mapping hinzufügen, haben Sie die Wahl zwischen einer lokalen Datei, einer Komponente über eine URL oder einer Komponente aus der Liste der globalen Ressourcen (*siehe Unterabschnitte weiter unten*).

Hinzufügen von Komponenten über eine URL

Das Hinzufügen von Komponenten über eine URL wird nur für Quellkomponenten 29 unterstützt. Es werden die Protokolle HTTP, HTTPS und FTP unterstützt. Je nach Art der Datenstruktur unterscheidet sich die Art und Weise, wie eine Komponente über eine URL hinzugefügt wird, eventuell. Die meisten Datenstrukturen werden folgendermaßen hinzugefügt:

1. Wählen Sie den gewünschten Komponententyp aus (z.B. **XML-Schema/Datei**).
2. Klicken Sie im Dialogfeld **Öffnen** auf **Zu URL wechseln**.
3. Geben Sie die URL der Datei in das Textfeld *Datei-URL* ein und klicken Sie auf **Öffnen** (*siehe unten*).



In der folgenden Liste sind die verschiedenen Optionen des Dialogfelds **Öffnen** beschrieben.

- *Öffnen als:* Mit dieser Option wird die Grammatik für den Parser definiert. Die Standardeinstellung, die auch die empfohlene Option ist, ist *Auto*.

- *Datei laden:* Wenn die Datei, die geladen wird, höchstwahrscheinlich nicht geändert wird, wählen Sie die Option *Cache/proxy verw.,* um Daten im Cache zu speichern und den Ladevorgang zu beschleunigen. Wenn Sie möchten, dass die Datei jedes Mal, wenn Sie das Mapping öffnen, neu geladen wird, wählen Sie die Option *Neu laden*.

- *Identifizierung:* Falls für den Server eine Passwort-Authentifizierung erforderlich ist, müssen Sie Ihren Benutzernamen und das Passwort eingeben. Wenn sich MapForce Ihren Benutzernamen und Ihr Passwort für das nächste Mal merken soll, aktivieren Sie das Kontrollkästchen *Passwort speichern zwischen Applikationsstarts*.

- *Server URL:* Bei Servern mit WebDAV- (Web Distributed Authoring and Versioning)-Support können Sie Dateien nach Eingabe der Server-URL in das Textfeld *Server URL* und Klicken auf **Durchsuchen**, durchsuchen. In der Vorschau werden alle Dateitypen angezeigt, stellen Sie daher sicher, dass Sie denselben Dateityp wie in Schritt 1 auswählen. Andernfalls kommt es zu Fehlern.

- *Ein/Auschecken:* Wenn Sie einen Microsoft SharePoint Server verwenden, aktivieren Sie das Kontrollkästchen *Microsoft SharePoint Server*. Daraufhin werden im Vorschaubereich die ein-/ausgescheckten Dateien angezeigt. Wenn Sie sicher stellen möchten, dass niemand anderer die

 © 2019-2025 Altova GmbH

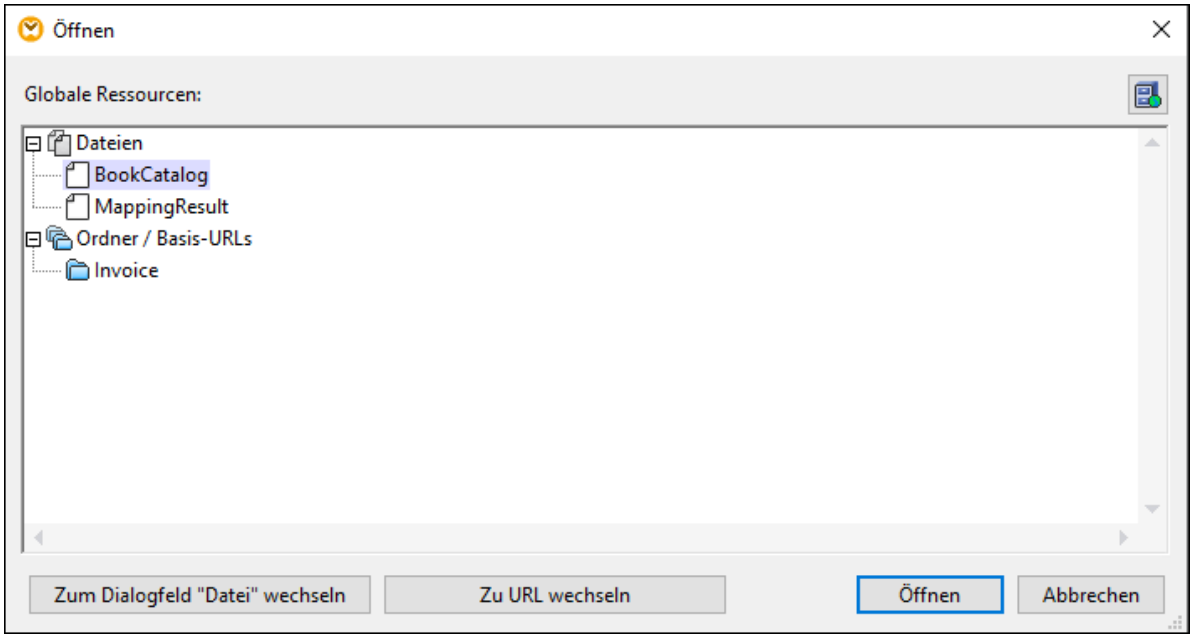
Datei auf dem Server bearbeiten kann, während Sie diese verwenden, klicken Sie mit der rechten Maustaste auf die Datei und wählen Sie den Befehl **Auschecken**(*siehe Abbildung oben*). Um eine von Ihnen vorher ausgecheckte Datei einzuchecken, klicken Sie mit der rechten Maustaste auf die Datei und wählen Sie **Einchecken**.

- *Zum Dialogfeld "Datei" wechseln:* Bei Klick auf diese Schaltfläche gelangen Sie zu dem Dialogfeld, in dem Sie eine lokale Datei auswählen können.
- *Zu globalen Ressourcen:* Bei Klick auf diese Schaltfläche gelangen Sie zu dem Dialogfeld, in dem Sie eine globale Ressource auswählen können.

Hinzufügen von globalen Ressourcen

Wenn Sie eine Datenbank (*Professional und Enterprise Edition*), eine Datei oder einen Ordner als globale Ressource definiert haben, können Sie diese zum Ihrem Mapping hinzufügen. Nähere Informationen zu globalen Ressourcen finden Sie unter Globale Altova-Ressourcen 432. Je nach Art der Datenstruktur werden globale Ressourcen eventuell unterschiedlich hinzugefügt. Die meisten Datenstrukturen werden folgendermaßen hinzugefügt:

1. Wählen Sie im den gewünschten Komponententyp aus (z.B. **XML-Schema/Datei**).
2. Klicken Sie im Dialogfeld **Öffnen** auf **Globale Ressourcen**.
3. Wählen Sie eine der Ressourcen aus der Liste aus und klicken Sie auf **Öffnen** (*siehe unten*).



Um globale Ressourcen hinzuzufügen, zu bearbeiten oder zu löschen, klicken Sie auf die Schaltfläche **Globale Ressourcen verwalten** (*oben rot umrandet*). Im Dialogfeld **Öffnen** für globale Ressourcen können Sie zurück zum Dialogfeld zum Öffnen von lokalen Dateien wechseln (**Zum Dialogfeld "Datei" wechseln**) oder eine Datei über eine URL öffnen (**Zu URL wechseln**).

2.1.2 Komponentengrundlagen

In diesem Kapitel wird beschrieben, wie Sie Strukturkomponenten [31] konfigurieren, durchsuchen und bearbeiten. Nähere Informationen dazu finden Sie in den Unterabschnitten weiter unten.

Ändern der Komponenteneinstellungen

Nachdem Sie eine Komponente zum Mapping-Bereich hinzugefügt haben, können Sie die Einstellungen dafür über das Dialogfeld **Komponenteneinstellungen** konfigurieren. Sie können das Dialogfeld **Komponenteneinstellungen** auf eine der folgenden Arten öffnen:

- Doppelklicken Sie auf den Komponententitel.
- Wählen Sie die Komponente aus und klicken Sie im Menü **Komponente** auf **Eigenschaften**.
- Klicken Sie mit der rechten Maustaste auf den Komponententitel und wählen Sie **Eigenschaften**.

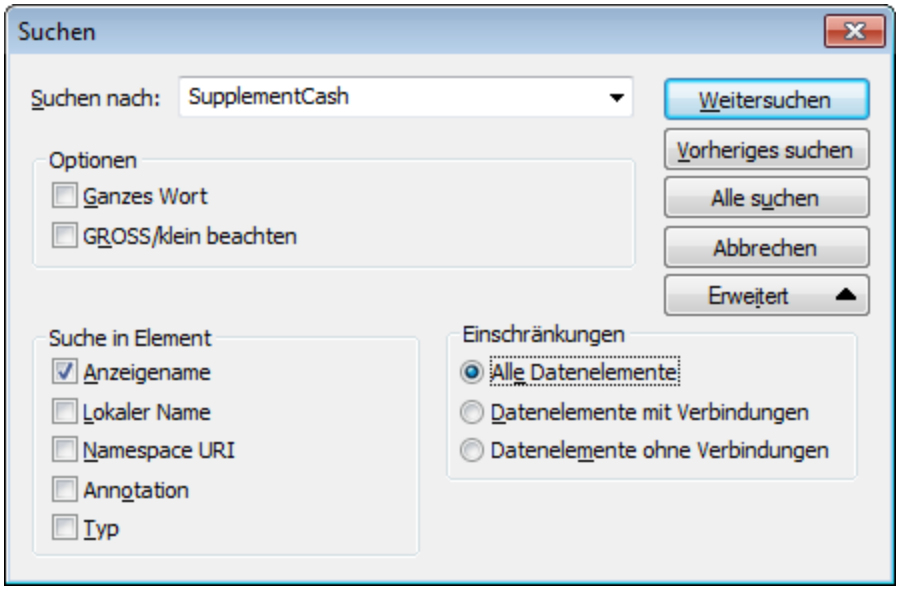
Eine Liste der verfügbaren Einstellungen finden Sie unter XML-Komponenteneinstellungen [131].

Bei allen dateibasierten Komponenten, wie z.B. einer XML-Datei wird neben dem Root-Node die Schaltfläche Datei Datei (*Basic Edition*) bzw. Datei/String (*Professional und Enterprise Edition*) angezeigt. Über diese Schaltfläche werden erweiterte Optionen zum Verarbeiten oder Generieren mehrerer Dateien in einem einzigen Mapping angezeigt. Nähere Informationen dazu finden Sie unter Verarbeitung mehrerer Input- oder Output-Dateien [422].

Durchsuchen einer Komponente

Um in einer Komponente nach einem bestimmten Node zu suchen, gehen Sie folgendermaßen vor:

1. Klicken Sie auf die gewünschte Komponente und drücken Sie die Tasten **Strg+F**.
2. Geben Sie einen Suchbegriff ein und klicken Sie auf **Weitersuchen/Vorheriges suchen/Alle suchen** (*siehe Abbildung unten*).

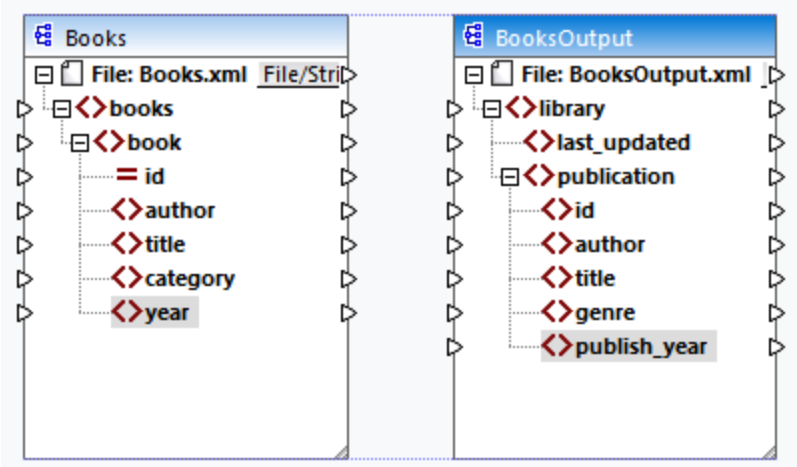


Über Option **Erweitert** können Sie festlegen, welche Datenelemente (Nodes) durchsucht werden sollen. Außerdem können Sie die Suchoptionen auf Basis bestimmter Verbindungen einschränken.

 © 2019-2025 Altova GmbH

Ausrichten von Komponenten

Wenn Sie Komponenten im Mapping-Bereich verschieben, werden Hilfslinien (gepunktete Linien) zur Ausrichtung der Komponenten angezeigt (*siehe Abbildung unten*).



Um diese Option zu aktivieren, gehen Sie folgendermaßen vor:

1. Klicken Sie im Menü **Extras** auf **Optionen.**
2. Aktivieren Sie in der Gruppe **Bearbeiten** das Kontrollkästchen **Komponenten beim Ziehen mit der Maus aneinander ausrichten**.

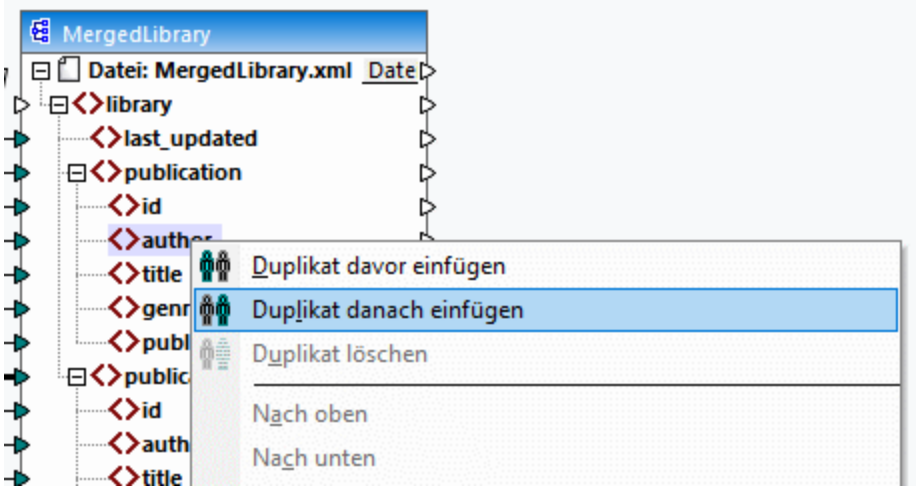
Input duplizieren

Manchmal muss eine Komponente so konfiguriert werden, dass sie Daten aus mehr als einer Datenquelle erhalten kann. Damit das Zielschema die Daten aus mehr als einem Quellschemas akzeptiert, können Sie jeden beliebigen Input-Node in Ihrer Zielkomponente duplizieren. Das Duplizieren von Input-Nodes ist nur bei Zielkomponenten sinnvoll. Duplizierte Nodes können Daten nur aufnehmen, Daten aus duplizierten Nodes können aber nicht gemappt werden. Sie können beliebig viele Nodes duplizieren.

Es gibt zwei Methoden, um einen Input zu duplizieren: (i) durch Auswahl des Befehls **Duplikat davor einfügen/Duplikat danach einfügen** aus dem Kontextmenü und (ii) durch Verbinden eines Quell-Node mit einem Ziel-Node, der bereits mit einem anderen Node verbunden ist. Die erste Option ist unten beschrieben. Informationen über die zweite Option finden Sie im zweiten Tutorial 106.

Duplikat davor/danach einfügen

Um einen bestimmten Input-Node zu duplizieren, klicken Sie mit der rechten Maustaste darauf und wählen Sie im Kontextmenü den Befehl **Duplikat davor einfügen/Duplikat danach einfügen** (*siehe Abbildung unten*). In der Abbildung oben wird der Node `author` dupliziert, damit Sie Daten von einem anderen Quell-Node darauf mappen können.



Anmerkung: Das Duplizieren von XML-Attributen ist unzulässig, da die XML-Instanz dadurch ungültig würde.

2.1.3 Dateipfade

Ein Mapping-Design (`*.mfd`) kann Referenzen zu mehreren Schema- und Instanzdateien enthalten. Anhand der Schemadateien ermittelt MapForce die Struktur der zu mappenden Daten und validiert diese. In der MapForce Professional und Enterprise Edition können Mappings auch Referenzen zu StyleVision Power Stylesheet-Dateien (`*.sps`) enthalten. Anhand dieser Dateien werden Daten für Ausgabeformate wie PDF, HTML und Word generiert. Mappings können auch Referenzen zu dateibasierten Datenbanken wie Microsoft Access oder SQLite haben.

Referenzen zu Dateien werden von MapForce erstellt, wenn Sie eine Komponente zum Mapping hinzufügen. Sie können solche Pfadreferenzen jedoch gegebenenfalls immer manuell einstellen oder ändern.

Dieser Abschnitt enthält eine Anleitung, wie Sie Pfade zu anderen von einem Mapping referenzierten Dateitypen konfigurieren oder ändern. Dieser Abschnitt ist in die folgenden Kapitel gegliedert:

- Relative und absolute Pfade 40
- Pfade in Ausführungsumgebungen 43

2.1.3.1 Relative und absolute Pfade

In diesem Kapitel wird erläutert, wie Sie in den von Ihrer Komponente referenzierten Dateien absolute und relative Pfade verwenden. Ein absoluter Pfad enthält den vollständigen Pfad zu einer Datei, beginnend mit dem Root-Verzeichnis (*siehe Beispiel XML-Komponenten unten*). Ein relativer Pfad gibt den Pfad der Datei relativ zu aktuellen Arbeitsverzeichnis an, z.B. `Books.xml`.

Sie können im Dialogfeld **Komponenteneinstellungen** (*siehe Beispiel unten*) absolute oder relative Pfade für verschiedene Dateien, die von der Komponente referenziert werden können, definieren. Im Folgenden finden Sie eine Liste dieser Dateien:

 © 2019-2025 Altova GmbH

- Datendateien (z.B. XML, JSON, CSV, usw.)
- Schema-Dateien (relevant für Komponenten mit Schemas)
- Von komplexen Input- oder Output-Parametern benutzerdefinierter Funktionen verwendete Dateien und Variablen vom Typ "complex"
- StyleVision Power Stylesheet (**`*.sps`**)-Dateien, anhand derer Daten für Ausgabeformate wie PDF, HTML und Word formatiert werden (*Professional und Enterprise Edition*)
- Datenbankdateien (*Professional und Enterprise Edition*)
- Von Datenbankkomponenten, die XML-Felder unterstützen, referenzierte Schema-Dateien (*Professional und Enterprise Edition*)

Kopieren-Einfügen und relative Pfade
Wenn Sie eine Komponente aus einem Mapping in ein anderes kopieren, wird überprüft, ob sich die relativen Pfade von Schema-Dateien anhand des Ordners des Ziel-Mappings auflösen lassen. Wenn die Pfade nicht aufgelöst werden können, werden Sie aufgefordert, die relativen Pfade zu absoluten zu machen.

Falsche Pfade
Wenn Sie eine Dateireferenz zu einem Mapping hinzufügen oder diese ändern und der Pfad nicht aufgelöst werden kann, zeigt MapForce eine Warnmeldung an. In den folgenden Fällen kann es zu falschen Pfadreferenzen kommen:

- Wenn Sie relative Pfade verwenden und die Mapping-Datei anschließend in ein neues Verzeichnis verschieben, ohne die Schema- und Instanzdatei ebenfalls zu verschieben.
- Wenn Sie absolute Pfade zu Dateien, die sich im selben Verzeichnis wie die Mapping-Datei befinden, verwenden und das Verzeichnis anschließend in einen anderen Ordner verschieben.

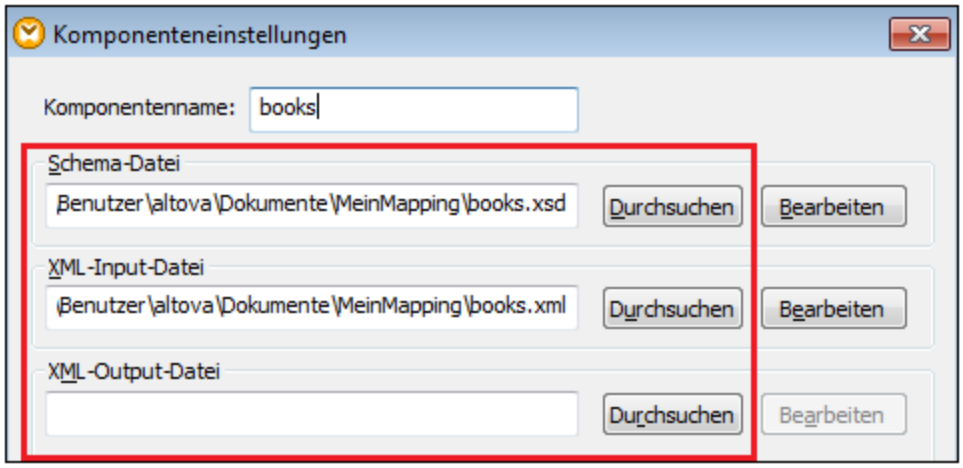
Bei falschen Pfadreferenzen markiert MapForce die betroffene Komponente rot. Um das Problem zu beheben, doppelklicken Sie auf die Überschrift der Komponente und aktualisieren Sie die fehlerhaften Pfadreferenzen im Dialogfeld **Komponenteneinstellungen**. Siehe auch Ändern der Komponenteneinstellungen 38.

Beispiel 1: XML-Komponente

Im Beispiel unten wird gezeigt, wie Dateipfade in einer XML-Komponente verwendet werden. Wenn Sie alle mit dem Mapping in Zusammenhang stehenden Dateien relativ zur Mapping-Datei (**`.mfd`**) speichern möchten, aktivieren Sie das Kontrollkästchen *Alle Dateipfade relativ zur MFD-Datei speichern* am unteren Rand des Dialogfelds **Komponenteneinstellungen**. Dies ist eine Standardoption, die empfohlen wird. Sie wirkt sich auf alle von der Komponente referenzierten Dateien (*in der Abbildung unten rot umrandet*) aus. Wenn Sie Ihr Mapping noch nicht gespeichert haben, werden im Dialogfeld **Komponenteneinstellungen** absolute Pfade zum Schema und/oder den Instanzdateien angezeigt. Damit im Dialogfeld **Komponenteneinstellungen** relative Pfade angezeigt werden, gehen Sie folgendermaßen vor:

1. Erstellen Sie ein neues Mapping 29 und fügen Sie eine Strukturkomponente 35, z.B. eine XML-Datei, der ein XML-Schema zugewiesen wurden, hinzu.
2. Doppelklicken Sie auf die Titelleiste der Komponente, um das Dialogfeld **Komponenteneinstellungen** zu öffnen.
3. Aktivieren Sie am unteren Rand des Dialogfelds **Komponenteneinstellungen** das Kontrollkästchen *Alle Dateipfade relativ zur MFD-Datei speichern*.
4. Speichern Sie Ihr Mapping.
5. Sie können die **Komponenteneinstellungen** nun erneut öffnen, um die relativen Pfade in den jeweiligen Textfeldern zu überprüfen.

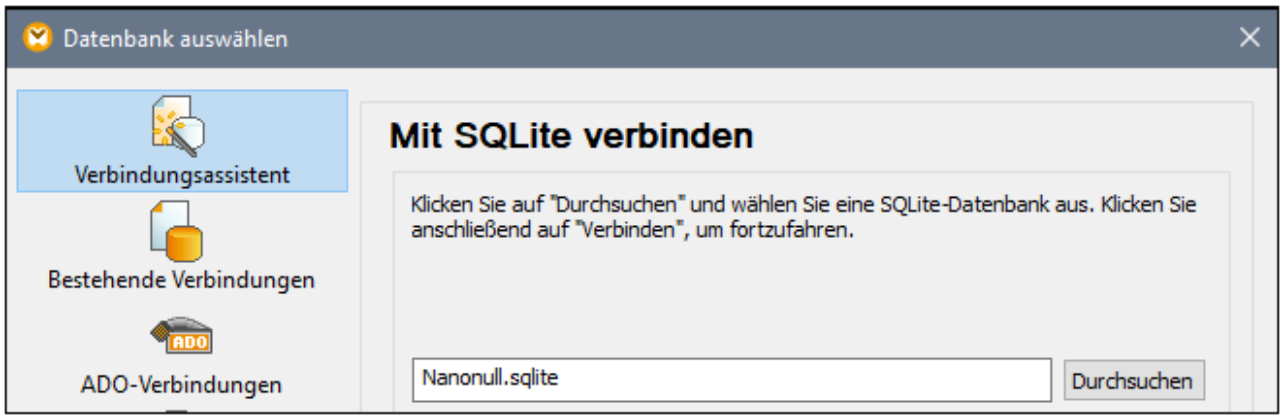
Anmerkung: Pfade, die kein lokales Laufwerk referenzieren und Pfade, in denen eine URL verwendet wird, werden nicht relativ gemacht.



Wenn das Kontrollkästchen *Alle Dateipfade relativ zur MFD-Datei speichern* aktiviert ist, aktualisiert MapForce die von der Komponente referenzierten Dateien auch dann, wenn Sie das Mapping in einem anderen Ordner speichern. Wenn sich alle Dateien im selben Verzeichnis wie das Mapping befinden, funktionieren die Pfadreferenzen weiterhin, auch wenn Sie das gesamte Verzeichnis in einen anderen Ordner auf dem Rechner verschieben.

Beispiel 2: Datenbankkomponente (Professional und Enterprise Edition)

Wenn Sie eine Datenbankdatei wie Microsoft Access oder SQLite zum Mapping hinzufügen, können Sie im Dialogfeld **Datenbank auswählen** anstelle eines absoluten Pfads einen relativen Pfad eingeben (*siehe Abbildung unten*). Bevor Sie relative Dateipfade eingeben, stellen Sie sicher, dass Sie die **`.mfd`**-Datei zuerst speichern. Wenn Sie den Pfad einer Datenbankkomponente, die bereits im Mapping verwendet wird, ändern möchten, klicken Sie im Dialogfeld **Komponenteneinstellungen** auf **Ändern**.



Anmerkung: Wenn Sie Programmcode generieren oder MapForce Server-Ausführungsdateien (**`.mfx`**) kompilieren oder wenn Sie das Mapping auf FlowForce Server bereitstellen, wird ein relativer Pfad in einen absoluten Pfad konvertiert, wenn in den Mapping-Einstellungen 74 das Kontrollkästchen *Pfade im*

 © 2019-2025 Altova GmbH

generierten Code absolut machen aktiviert ist. Nähere Informationen dazu finden Sie unter Pfade im generierten Code 43.

2.1.3.2 Pfade in Ausführungsumgebungen

Wenn Sie Code anhand von Mappings generieren, werden die generierten Dateien von der gewählten Zielumgebung, z.B. RaptorXML Server ausgeführt. Damit das Mapping erfolgreich ausgeführt werden kann, müssen alle relativen Pfade auch in der Umgebung, in der das Mapping ausgeführt wird, funktionieren. Im Folgenden sehen Sie die Basispfade für die einzelnen Zielsprachen.

Zielsprache	Basispfad
XSLT, XSLT2, XSLT3	Pfad der XSLT-Datei
XQuery*	Pfad der XQuery-Datei
C++, C#, Java*	Arbeitsverzeichnis der generierten Applikation
Built-In* (bei Anzeige einer Mapping-Vorschau in MapForce)	Pfad der Mapping-Datei (`.mfd`).
Build-In* (bei Ausführung des Mappings mit MapForce Server)	Das aktuelle Arbeitsverzeichnis
Built-In* (bei Ausführung des Mappings mit MapForce Server durch FlowForce Server)	Das Arbeitsverzeichnis für den Auftrag oder das Arbeitsverzeichnis von FlowForce Server.

** In der MapForce Professional und Enterprise Edition verfügbare Sprachen*

Konvertierung relativer Pfade in absolute
Wenn Sie Programmcode generieren oder MapForce Server-Ausführungsdateien (`.mfx`) kompilieren oder wenn Sie das Mapping auf FlowForce Server bereitstellen, wird ein relativer Pfad in einen absoluten Pfad konvertiert, wenn in den Mapping-Einstellungen 74 das Kontrollkästchen **Pfade im generierten Code absolut machen** aktiviert ist.

Wenn Sie Code generieren und das Kontrollkästchen aktiviert ist, löst MapForce alle relativen Pfade anhand des Verzeichnisses der `.mfd`-Datei auf und macht sie im generierten Code zu absoluten Pfaden. Diese Einstellung wirkt sich auf die Pfade der folgenden Dateien aus:

- Input- und Output-Instanzdateien für alle dateibasierten Komponenten
- Access- und SQLite-Datenbankdateien, die als Mapping-Komponenten verwendet werden (*Professional und Enterprise Edition*).

Bibliothekspfade im generierten Code

Mapping-Dateien können optional Pfadreferenzen zu verschiedenen Bibliotheken enthalten. So können etwa benutzerdefinierte Funktionen aus anderen Mapping-Dateien oder Funktionen aus benutzerdefinierten XSLT-, XQuery-*, C#-* oder Java-Bibliotheken* aus `.mff*`-Dateien (MapForce Funktionsdateien) importiert werden. Nähere Informationen dazu finden Sie unter Verwalten von Funktionsbibliotheken. 215.

** In der MapForce Professional und Enterprise Edition verfügbare Funktionalitäten*

Die Option **Pfade im generierten Code absolut machen** gilt nur für Mapping-Komponenten und wirkt sich nicht auf die Pfade zu externen Bibliotheken aus. Pfade zu allen anderen Bibliotheken als XSLT und XQuery werden im generierten Code in absolute Pfade konvertiert. Wenn Ihre Mapping-Datei z.B. Bibliotheksreferenzen wie .NET-, **`.dll`**- oder Java-Klassendateien enthält und der generierte Code in einer anderen Umgebung ausgeführt werden soll, so müssen die referenzierten Bibliotheken in der Zielumgebung unter demselben Pfad vorhanden sein.

Wenn Sie beabsichtigen, eine XSLT- oder XQuery-Datei anhand eines Mappings zu generieren, können Sie den Bibliothekspfad folgendermaßen relativ zur generierten XSLT- oder XQuery-Datei machen:

1. Öffnen Sie die Mapping-Einstellungen 74.
2. Aktivieren Sie das Kontrollkästchen **Bibliotheken relativ zu den generierten XSLT / XQuery-Dateien referenzieren**. Stellen Sie sicher, dass die XSLT- oder XQuery-Bibliotheksdatei unter diesem Pfad auch tatsächlich vorhanden ist.

© 2019-2025 Altova GmbH

2.2 Verbindungen

Eine Verbindung ist eine Linie, die eine Quellkomponente mit einer Zielkomponente verbindet. Verbindungen zeigen visuell an, wie Daten von einem Node auf einen anderen gemappt sind. In den Unterabschnitten weiter unten werden verschiedene Aktionen beschrieben, die Sie im Zusammenhang mit Verbindungen durchführen können.

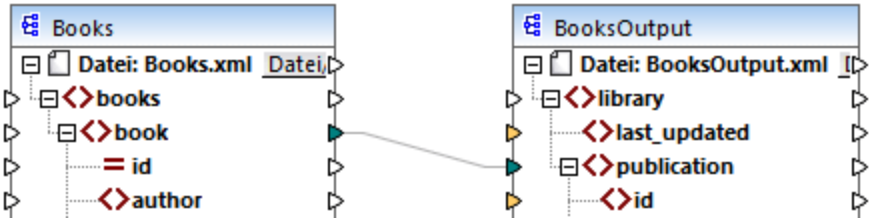
Erstellen, Kopieren, Löschen einer Verbindung

Um eine Verbindung zwischen zwei Datenelementen zu erstellen, klicken Sie auf den Output-Konnektor 30 eines Quell-Node und ziehen Sie ihn auf einen Ziel-Node. Ein Input-Konnektor kann *nur eine* eingehende Verbindung haben. Wenn Sie versuchen, eine zweite Verbindung zum selben Input hinzuzufügen, werden Sie gefragt, ob Sie die Verbindung durch eine neue ersetzen oder das Input-Datenelement duplizieren 39 möchten. Ein Output-Konnektor kann mehrere Verbindungen zu unterschiedlichen Inputs haben.

Um eine Verbindung auf ein anderes Datenelement zu kopieren, klicken Sie auf den dick angezeigten Abschnitt des Verbindungsendes (*siehe Abbildung in Verschieben einer Verbindung*) und ziehen Sie ihn bei gedrückter **Strg**-Taste auf das ausgewählte Zieldatenelement.

Um eine Verbindung zu löschen, klicken Sie darauf und drücken Sie die Taste **Löschen**. Klicken Sie alternativ dazu mit der rechten Maustaste auf die Verbindung und wählen Sie im Kontextmenü den Befehl **Löschen**.

Obligatorische Inputs
Als Hilfe beim Mapping werden obligatorische Inputs in Zielkomponenten orange markiert. Im Beispiel unten sehen Sie, dass bei Verbindung des Elements `book` der Komponente `Books` mit dem Element `publication` der Komponente `BookOutput` die Konnektoren der obligatorischen Nodes der Komponente `BooksOutput` markiert werden. Wenn obligatorische Inputs nicht verbunden werden, werden die entsprechenden Nodes nicht auf die Zielkomponente gemappt und das Mapping wird ungültig.



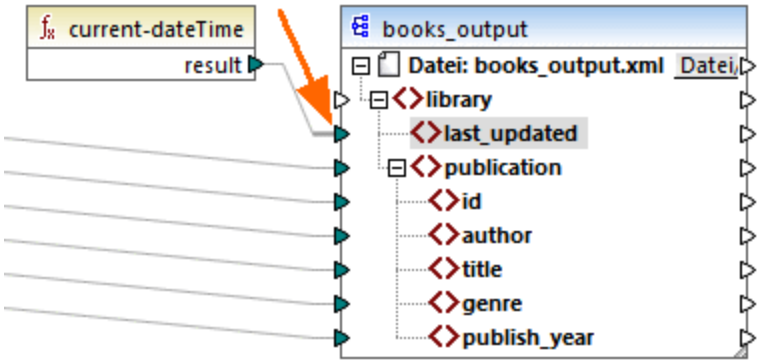
Fehlende Parent-Verbindungen
Wenn Sie manuell eine Verbindung zwischen einem Quell- und einem Ziel-Node erstellen, analysiert MapForce die möglichen Mapping-Ergebnisse. Wenn Sie zwei Child-Nodes verbinden, ohne ihre Parent-Nodes zu verbinden, wird eine Meldung angezeigt, in der vorgeschlagen wird, auch den Parent des Quell-Node mit dem Parent des Ziel-Node zu verbinden. Dadurch wird verhindert, dass eventuell nur ein einziger Child-Node im **Ausgabefenster** angezeigt wird.

Um die Anzeige solcher Meldungen zu deaktivieren, gehen Sie folgendermaßen vor:

1. Klicken Sie im Menü **Extras** auf **Optionen.**
2. Öffnen Sie die Gruppe **Meldungen**.
3. Deaktivieren Sie das Kontrollkästchen **Verbindung von übergeordneten Datenelementen beim Verbinden von Datenelementen vorschlagen**.

Verschieben einer Verbindung

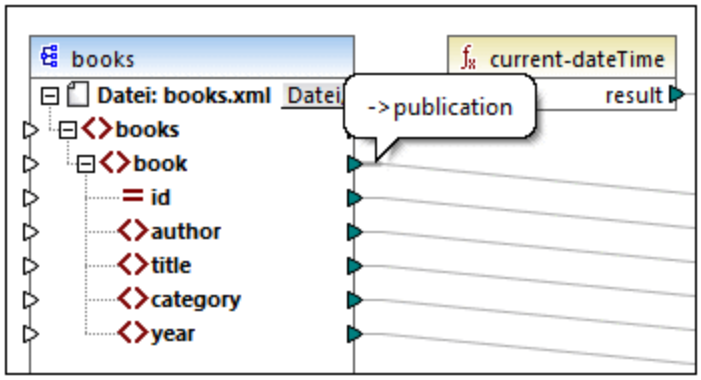
Um eine Verbindung auf einen anderen Node zu verschieben, klicken Sie auf den dick angezeigten Abschnitt des Verbindungsendes (*siehe Abbildung unten*) und ziehen Sie ihn auf den ausgewählten Ziel-Node.



Tooltips zu Verbindungen anzeigen

Mit Hilfe von Tooltips zu Verbindungen könne Sie die Namen von (i) Nodes, auf die Daten gemappt werden oder (ii) Nodes, von denen aus Daten gemappt werden, anzeigen. Damit Tipps angezeigt werden, aktivieren Sie die Symbolleisten-Schaltfläche (**Tipps anzeigen**). Um die Namen von Nodes, auf die Daten gemappt werden, zu sehen, platzieren Sie den Cursor über den dicken Abschnitt einer Verbindung neben dem Output-Konnektor (*siehe Abbildung unten*). Um den Namen eines Node, von dem aus Daten gemappt werden, zu sehen, platzieren Sie den Cursor über den dicken Abschnitt einer Verbindung in der Nähe des Input-Konnektors. Wenn vom selben Output aus mehrere Verbindungen definiert wurden, werden im Tooltip maximal zehn Datenelementnamen angezeigt.

Im Beispiel unten hat der Ziel-Node, auf den die Daten aus dem Element `book` gemappt werden, den Namen `publication`.



Ändern der Verbindungseinstellungen

Um die Verbindungseinstellungen zu ändern, wählen Sie eine der folgenden Methoden:

© 2019-2025 Altova GmbH

- Wählen Sie eine Verbindung aus. Klicken Sie anschließend im Menü **Verbindung** auf **Eigenschaften**
- Doppelklicken Sie auf die Verbindung.
- Klicken Sie mit der rechten Maustaste auf die Verbindung und wählen Sie den Befehl **Eigenschaften**.

Nähere Informationen dazu finden Sie unter Verbindungseinstellungen 56.

Selektives Markieren von Verbindungen

Sie können in MapForce Verbindungen in einem Mapping selektiv hervorheben. Diese Funktionalität ist nützlich, wenn Ihr Mapping mehrere Komponenten mit mehreren Verbindungen aufweist. Durch selektives Hervorheben der Verbindungen können Sie besser überprüfen, ob die Nodes der ausgewählten Komponente korrekt gemappt wurden. Beachten Sie, dass mit dem für die Symbolleisten-Schaltflächen verwendeten Begriff *Konnektor* eine Verbindung gemeint ist, d.h. eine Verbindungslinie zwischen den Komponenten-Nodes. Siehe verfügbare Optionen unten.

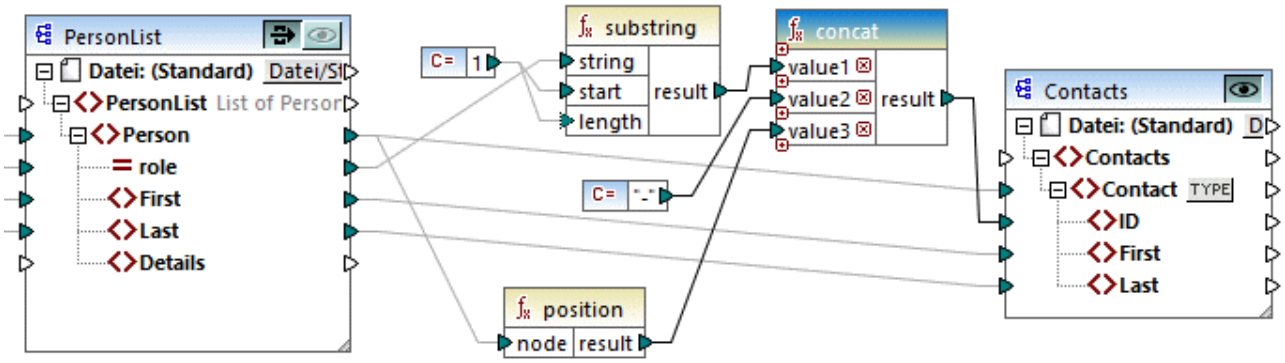
	Ausgewählte Komponentenkonnektoren anzeigen (nur direkte Verbindungen)
	Konnektoren zwischen Quelle und Ziel anzeigen (direkte und indirekte Verbindungen)

Nur direkte Verbindungen
Wenn die Schaltfläche **Direkte Verbindungen** *nicht* aktiv ist, werden alle Verbindungen schwarz angezeigt. Wenn die Schaltfläche **Direkte Verbindungen** aktiv ist, werden nur Verbindungen, die mit der aktuell ausgewählten Komponente in Zusammenhang stehen, schwarz angezeigt. Andere Verbindungen erscheinen hellgrau.

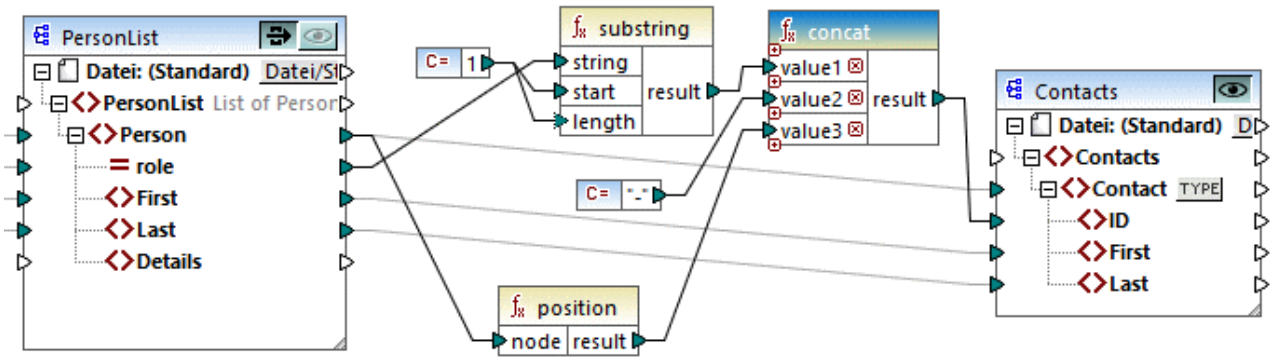
Direkte und indirekte Verbindungen
Die Schaltfläche **Quelle-Ziel-Verbindungen** steht nur zur Verfügung, wenn die Schaltfläche **Direkte Verbindungen** aktiv ist. Wenn die Schaltfläche **Quelle-Ziel-Verbindungen** gedrückt wird, können Sie Verbindungen der aktuell ausgewählten Komponente, einschließlich ihrer direkten Verbindungen und der Verbindungen der damit verbundenen Komponenten bis zu den Quell- und Zieldateien zurückverfolgen.

Ein Beispiel für die Funktionsweise dieser beiden Optionen finden Sie unten.

Beispiel
In der Abbildung unten sehen Sie einen Ausschnitt aus dem Mapping **`ChainedPersonList.mfd`** aus dem Ordner **`MapForceExamples`**. Im unten gezeigten Mapping haben wir die Schaltfläche **Direkte Verbindungen** gedrückt und auf die Titelleiste der Komponente **`concat`** geklickt, die Schaltfläche **Quelle-Ziel-Verbindungen** aber noch nicht gedrückt. Wir sehen daher nun, dass nur die direkten Verbindungen der **`concat`**-Funktion zur Konstante **`"-"`** und den Komponenten `substring`, `position` und `Contacts` schwarz angezeigt werden. Die anderen Verbindungen im Mapping sind hellgrau.



Als nächstes werden wir nun auf die Schaltfläche **Quelle-Ziel-Verbindungen** drücken. In der Abbildung unten sehen Sie, was sich geändert hat:



Wenn die Schaltfläche **Quelle-Ziel-Verbindungen** aktiv ist, werden einige weitere Verbindungen schwarz angezeigt: (i) die Verbindungen der **`substring`**-Funktion zur Konstante **`1`** und der Komponente `PersonList` und (ii) die Verbindung der **`position`**-Funktion mit der Komponente `PersonList`. Die Verbindungen zwischen der Komponente `PersonList` und der davor liegenden Komponente bleiben hingegen hellgrau. Wenn Sie also die Schaltfläche **Quelle-Ziel-Verbindungen** aktivieren und auf eine Komponente klicken, können Sie die direkten Verbindungen der Komponente zurückverfolgen. Wenn die ausgewählte Komponente mit Transformationskomponenten [32] (z.B. Funktionen, Konstanten, Filtern, Sortierkomponenten, SQL-NoSQL-WHERE/ORDER-Komponenten, if-else-Bedingungen, Wertezuordnungen) verbunden ist, sehen Sie auch deren Verbindungen bis zu den Strukturkomponenten [31] (wie z.B. `PersonList` oben), Variablen, Join-Komponenten oder Webservice-Funktion, mit denen diese Transformationskomponenten verbunden sind.

In diesem Abschnitt

Dieser Abschnitt enthält eine Übersicht über Verbindungen und ist in die folgenden Kapitel gegliedert:

- Verbindungsarten [49]
- Verbindungseinstellungen [56]
- Kontextmenü für Verbindungen [57]
- Fehlerhafte Verbindungen [59]
- Beibehalten von Verbindungen nach Löschen von Komponenten [61]

 © 2019-2025 Altova GmbH

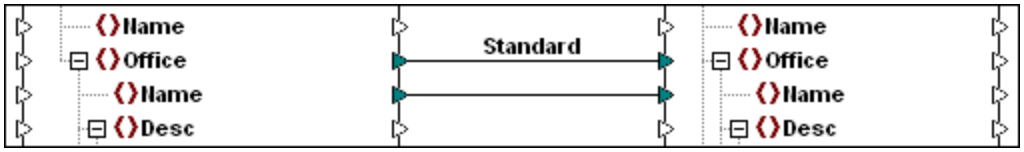
2.2.1 Verbindungsarten

In MapForce stehen die folgenden Verbindungsarten zur Verfügung:

- Zielorientierte Verbindungen 49 (Standard)
- Quellorientierte Verbindungen 49 (Mixed Content)
- Verbindungen mit identen Sub-Einträgen 52
- Alles kopieren-Verbindungen 54 (Sub-Einträge kopieren)

Zielorientierte im Vergleich zu quellorientierten Verbindungen

Zielorientierte und quellorientierte Verbindungen schließen einander gegenseitig aus. Welche dieser beiden Optionen Sie auswählen sollten, hängt von der Reihenfolge, in der Nodes gemappt werden müssen, ab. In zielorientierten Verbindungen wird die Reihenfolge der Nodes in der Ausgabe durch das *Ziel*-Schema vorgegeben. Diese Verbindungsart eignet sich für die meisten Mapping-Szenarien und ist der Standardverbindungstyp in MapForce. Zielorientierte Verbindungen werden in einem Mapping als durchgezogene Linien angezeigt (*siehe Abbildung unten*).



Wenn Sie XML-Nodes mit gemischtem Inhalt (mixed content, d.h. Child Nodes und Text) mappen möchten, sind zielorientierte Verbindungen manchmal nicht geeignet. In diesem Fall wird eine quellorientierte Verbindung 49 empfohlen. Dabei wird die Reihenfolge der Nodes in der Ausgabe durch das *Quell*-Schema vorgegeben.

Idente Sub-Einträge und "Alles kopieren"-Verbindungen

Idente Sub-Einträge und "Alles kopieren"-Verbindungen gehören zu einer Untergruppe von ziel- und quellorientierte Verbindungen. Damit werden Daten zwischen Nodes mit Child-Nodes, die in der Quell- und Zielkomponente ähnlich oder identisch sind, gemappt. "Alles kopieren"-Verbindungen sind Verbindungen mit identen Sub-Einträgen ähnlich, weisen aber anstelle mehrerer Verbindungen nur eine dicke Verbindung auf, wodurch das Mapping übersichtlicher wird.

In diesem Abschnitt finden Sie Informationen über die einzelnen Verbindungsarten und die Szenarien, in denen diese Verbindungsarten sich als nützlich erweisen.

2.2.1.1 Quellorientierte Verbindungen

Bei einer quellorientierten Verbindung, kann gemischter Inhalt (Mixed Content) (Text- und Child-Nodes) automatisch in derselben Reihenfolge, wie er in der XML-*Quelldatei* vorkommt, gemappt werden. Mixed Content-Verbindungen werden auf Ebene des Parent-Node als gepunktete Linien angezeigt (*siehe Mappen des <para>-Elements*). In diesem Kapitel wird erläutert, wie Sie Mixed Content mappen. Außerdem wird gezeigt, wie sich die Verwendung von Standardverbindungen (zielorientierten Verbindungen) mit Mixed Content auswirkt.

Anmerkung: Auch in Datenbankfeldern mit gemischtem Inhalt (Mixed Content) können quellorientierte Verbindungen verwendet werden (*Professional und Enterprise Edition*).

Anmerkung: Damit Mixed Content akzeptiert wird, müssen die Zielkomponenten Mixed Content Nodes haben.

Mappen von gemischtem Inhalt

In diesem Kapitel wird erläutert, wie Sie Mixed Content (gemischten Inhalt) mittels einer quellorientierten Verbindung mappen. Sie benötigen die folgenden Dateien: **`Tut-OrgChart.xml`**, **`Tut-Orgchart.mfd, Tut-Person.xsd`** und **`Tut-OrgChart.xsd`**, die sich im Ordner Tutorial 15 befinden.

XML-Quellinstanz

Unten sehen Sie einen Ausschnitt aus der Datei **`Tut-OrgChart.xml`**. In diesem Beispiel geht es um das Mixed Content-Element `<para>` mit seinen Child-Nodes `<bold>` und `<italic>`. Das Element `para` enthält auch eine Processing Instruction (`<?sort alpha-ascending?>`) sowie einen Kommentar (`<!--Company details... -->`), die, wie unten gezeigt, ebenfalls gemappt werden können. Beachten Sie die Reihenfolge der `text`-Nodes und der `bold/italic` Nodes in der XML-Instanzdatei.

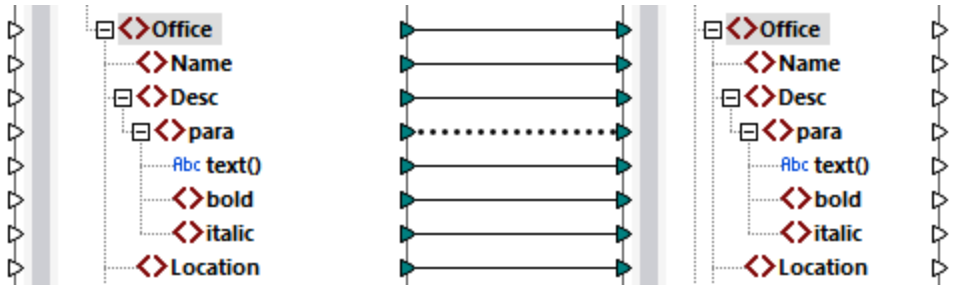
```
8     <Desc>
9        <para>The company was established in <bold>Vereno</bold> in 1995. Nanonull develops
nanoelectronic technologies for <italic>multi-core processors.</italic> February 1999 saw the
unveiling of the first prototype <bold>Nano-grid.</bold> The company hopes to expand its
operations <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>
```

Mappen des <para>-Elements

In der Abbildung unten sehen Sie einen Ausschnitt aus **`Tut-Orgchart.mfd`**. Die gepunktete Linie im Beispiel unten zeigt, dass das Element `<para>` Mixed Content hat. Um manuell Mixed Content-Verbindungen zu erstellen, gehen Sie folgendermaßen vor:

1. Wählen Sie die Menüoption **Verbindung | Idente Sub-Einträge automatisch verbinden**, um idente Sub-Einträge 52 automatisch zu verbinden. Alternativ dazu können Sie den Node `<para>` mit seinen Sub-Einträgen manuell mappen.
2. Verbinden Sie das Datenelement `<para>` in der Quellkomponente mit dem Datenelement `<para>` in der Zielkomponente. Daraufhin wird eine Meldung angezeigt, in der Sie gefragt werden, ob die Verbindung als quellorientierte Verbindung definiert werden soll.
3. Klicken Sie auf **Ja**, um eine Mixed Content-Verbindung zu erstellen.
4. Klicken Sie auf **Ausgabe**, um das Ergebnis des Mappings zu sehen. Klicken Sie in der Symbolleiste im **Ausgabebereich** auf die Schaltfläche (**Zeilenumbruch**), um das Codefragment zur Gänze (d.h. so dass es nicht über die Bildlaufleiste hinausragt) im **Ausgabebereich** zu sehen. Der gemischte Inhalt des `<para>`-Node wurde in derselben Reihenfolge, wie er in der XML-Quelldatei aufscheint, gemappt.

 © 2019-2025 Altova GmbH



Processing Instructions und Kommentare
Um Processing Instructions und/oder Kommentare zu mappen, gehen Sie folgendermaßen vor:

1. Klicken Sie mit der rechten Maustaste auf die Mixed Content-Verbindung (gepunktete Linie) und wählen Sie **Eigenschaften**.
2. Aktivieren Sie unter **Quellorientiert (Mixed content)**, die Optionen **Processing Instructions mappen** und **Comments mappen**.

Zielorientierte Verbindungen mit Mixed Content
Wenn Sie für Mixed Content zielorientierte Verbindungen verwenden, erhalten Sie eventuell unerwünschte Ergebnisse. Um zu sehen, wie sich zielorientierte Verbindungen auf die Reihenfolge von Mixed Content Nodes auswirken, befolgen Sie die Anweisungen unten:

1. Öffnen Sie **`Tut-OrgChart.mfd`** aus dem Ordner **`Tutorial`**.
2. Aktivieren Sie die Symbolleisten-Schaltfläche (Idente Sub-Einträge automatisch verbinden 52). Deaktivieren Sie in den Einstellungen für "Idente Sub-Einträge" 52 das Kontrollkästchen **Alles kopieren"-Verbindungen erstellen**. Dadurch wird verhindert, dass automatisch "Alles kopieren"-Verbindungen 54 erstellt werden.
3. Erstellen Sie eine Verbindung zwischen dem Node `para` in der Quellkomponente und dem Node `para` in der Zielkomponente. Daraufhin wird eine Meldung angezeigt, in der Sie gefragt werden, ob die Verbindung als quellorientierte Verbindung definiert werden soll. Klicken Sie auf **Nein**. Dadurch wird eine zielorientierte Verbindung erstellt.
4. Klicken Sie auf den **Ausgabe**-Bereich, um das Ergebnis des Mappings zu sehen (*siehe Abbildung unten*).

```
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>
```

In der Abbildung oben sehen Sie, dass der Inhalt des Datenelements `text()` aus der Quellkomponente auf die Zielkomponente gemappt wurde. Die Reihenfolge der Child-Nodes (`bold` und `italic`) entspricht in der Ausgabe allerdings der im XML-Zielschema vorgegebenen Reihenfolge dieser Nodes, d.h. die Elemente `bold` und `italic` wurden nicht in den Text integriert, sondern separat gemappt.

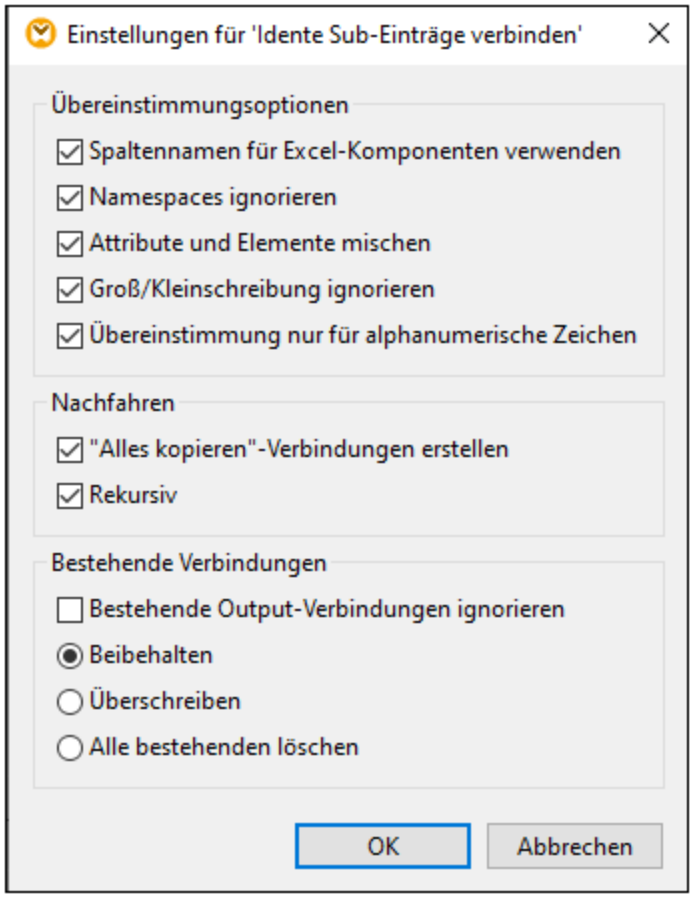
2.2.1.2 Verbindungen mit identen Sub-Einträgen

Bei Verbindungen mit identen Sub-Einträgen werden alle Sub-Einträge, die in der Quell- und Zieldatei denselben Namen haben, automatisch miteinander verbunden. Um diese Option zu aktivieren, wählen Sie eine der folgenden Methoden:

- Aktivieren Sie die Symbolleisten-Schaltfläche (**Idente Sub-Einträge automatisch verbinden**).
- Klicken Sie im Menü **Verbindung** auf den Befehl **Idente Sub-Einträge automatisch verbinden**.

Einstellungen für Verbindungen mit identen Sub-Einträgen

Um die Einstellungen für die Verbindung identischer Sub-Einträge zu konfigurieren, klicken Sie mit der rechten Maustaste auf eine beliebige Verbindung und wählen Sie im Kontextmenü die Option **Idente Sub-Einträge verbinden** oder gehen Sie zum Menü **Verbindung** und klicken Sie auf **Einstellungen für 'Idente Sub-Einträge verbinden'**. Daraufhin wird das Dialogfeld **Einstellungen für 'Idente Sub-Einträge verbinden'** aufgerufen (*Abbildung unten*).



In der nachstehenden Liste finden Sie eine Beschreibung der Optionen im Dialogfeld **Einstellungen für 'Idente Sub-Einträge verbinden'**. Die Einstellungen in diesem Dialogfeld werden nur wirksam, wenn die Symbolleisten-Schaltfläche (**Aktiviert/Deaktiviert die automatische Verbindung von Sub-Einträgen**) aktiviert ist.

© 2019-2025 Altova GmbH

Übereinstimmungsoptionen
Im Abschnitt *Übereinstimmungsoptionen* können Sie die Kriterien für die Übereinstimmung weniger streng definieren und festlegen, wie Node-Namen verglichen werden sollen. Es stehen die folgenden Optionen zur Verfügung:

- *Spaltennamen für Excel-Komponenten verwenden* Diese Option ist nur auf Excel-Komponenten anwendbar (*Enterprise Edition*). Sie bedeutet, dass für den Vergleich anstelle von Spaltenreferenznamen (z.B. A, B, C) benutzerdefinierte Spaltennamen (z.B. `Company`) verwendet werden. Benutzerdefinierte Spaltennamen werden im Dialogfeld **Zellenbereich auswählen** definiert und scheinen in Excel-Komponenten als Annotationen auf.
- *Namespaces ignorieren:* Idente Sub-Einträge werden unabhängig von den Namespaces von Child-Nodes verbunden.
- *Attribute und Elemente mischen:* Damit können Verbindungen zwischen Attributen und Elementen desselben Namens erzeugt werden. So wird z.B. zwischen zwei Nodes mit dem Namen `ID` eine Verbindung erstellt, selbst wenn es sich bei einem der Nodes um ein Element und beim anderen um ein Attribut handelt.
- *Groß/Kleinschreibung ignorieren:* Idente Sub-Einträge werden unabhängig von der Groß- und Kleinschreibung der Child-Node-Namen verbunden.
- *Übereinstimmung nur für alphanumerische Zeichen:* Wenn diese Option aktiv ist, werden nur Ziffern und Buchstaben verglichen. Andere Zeichen wie Leerzeichen, Kommas, Punkte, usw. werden vor dem Vergleich verworfen.

Nachfahren
Im Abschnitt *Nachfahren* wird definiert, wie mit Child-Nodes verfahren werden soll. Es stehen die folgenden Optionen zur Verfügung:

- *"Alles kopieren"-Verbindungen erstellen:* Diese Einstellung ist standardmäßigt aktiv. Damit wird (wenn möglich) eine "Alles kopieren"-Verbindung 54 erstellt, die Daten zwischen Nodes mit Child-Nodes, die einander ähnlich oder miteinander identisch sind, mappt. Eine "Alles kopieren"-Verbindung wird durch eine einzige dicke Linie dargestellt, wodurch das Mapping übersichtlicher wird.
- *Rekursiv:* Erstellt zwischen identen Nodes neue Verbindungen, wenn Sie denselben Namen haben. Es spielt keine Rolle, wie tief die Nodes in der Struktur verschachtelt sind.

Bestehende Verbindungen
Im Abschnitt *Bestehende Verbindungen* wird definiert, was mit bestehenden Verbindungen geschehen soll. Es stehen die folgenden Optionen zur Verfügung:

- *Bestehende Output-Verbindungen ignorieren:* Mit dieser Option werden zusätzliche Verbindungen für alle identischen Nodes erstellt, selbst wenn diese bereits ausgehende Verbindungen aufweisen.
- *Beibehalten:* Mit dieser Option werden bestehende Verbindungen beibehalten.
- *Überschreiben:* Mit dieser Option werden bestehende Verbindungen überschrieben.
- *Alle bestehenden löschen:* Mit dieser Option werden alle bestehenden Verbindungen gelöscht, bevor neue erzeugt werden.

Löschen von Verbindungen als Gruppe

Um Verbindungen als Gruppe zu löschen, gehen Sie vor, wie unten beschrieben:

1. Klicken Sie mit der rechten Maustaste auf einen Node-Namen in der Komponente.
2. Wählen Sie im Kontextmenü den Befehl **Verbindungen löschen | Alle <...> Verbindungen löschen** (*siehe Abbildung unten*).

- *Alle direkten Verbindungen löschen:* Mit dieser Option löschen Sie alle Verbindungen, die direkt auf den oder vom ausgewählten Node aus gemappt sind.
- *Alle eingehenden Child-Verbindungen löschen:* Diese Option ist nur aktiv, wenn Sie mit der rechten Maustaste auf einen Parent Node in einer Zielkomponente geklickt haben. Mit dieser Option löschen Sie alle eingehenden Child-Verbindungen des ausgewählten Parent Node.
- *Alle hinausgehenden Child-Verbindungen löschen:* Diese Option ist nur aktiv, wenn Sie mit der rechten Maustaste auf einen Parent Node in einer Quellkomponente geklickt haben. Mit dieser Option löschen Sie alle ausgehenden Child-Verbindungen des ausgewählten Parent Node.

2.2.1.3 "Alles kopieren"-Verbindungen

Mit "Alles kopieren"-Verbindungen werden Daten zwischen Nodes mit Child-Nodes, die einander ähnlich oder miteinander identisch sind, gemappt. Alles kopieren"-Verbindungen sind nur für dasselbe Format (z.B. JSON auf JSON oder XML auf XML) möglich. Dies gilt auch für alle Textkomponenten: Flat Flat Files, FlexText- und EDI-Dateien. Da es sich bei allen diesen Formaten um Textdateien handelt, können Sie jedes davon miteinander kombinieren und z.B. zwischen EDI- und FlexText-Dateien eine "Alles kopieren"-Verbindung erstellen.

Der Hauptvorteil von "Alles kopieren"-Verbindungen ist, dass der Mapping-Arbeitsbereich dadurch übersichtlicher wird: Anstelle mehrerer Verbindungen wird eine einzige "dicke" Verbindungslinie gezogen (*siehe Beispiel unter Manuelles Erstellen von "Alles kopieren"-Verbindungen*). In den folgenden Unterabschnitten wird erläutert, wie Sie "Alles kopieren"-Verbindungen automatisch und manuell erstellen.

Erstellen einer "Alles kopieren"-Verbindung

Um eine "Alles kopieren"-Verbindung automatisch zu erstellen, gehen Sie folgendermaßen vor:

1. Gehen Sie zum Menü **Verbindung**.
2. Klicken Sie auf **Einstellungen für 'Idente Sub-Einträge verbinden'**.
3. Aktivieren Sie das Kontrollkästchen **"Alles kopieren"-Verbindungen erstellen** und klicken Sie auf **OK**.
4. Klicken Sie auf die Symbolleisten-Schaltfläche **Aktiviert/Deaktiviert die automatische Verbindung von Sub-Einträgen**. Klicken Sie alternativ dazu im Menü **Verbindung** auf den Befehl **Idente Sub-Einträge automatisch verbinden**.

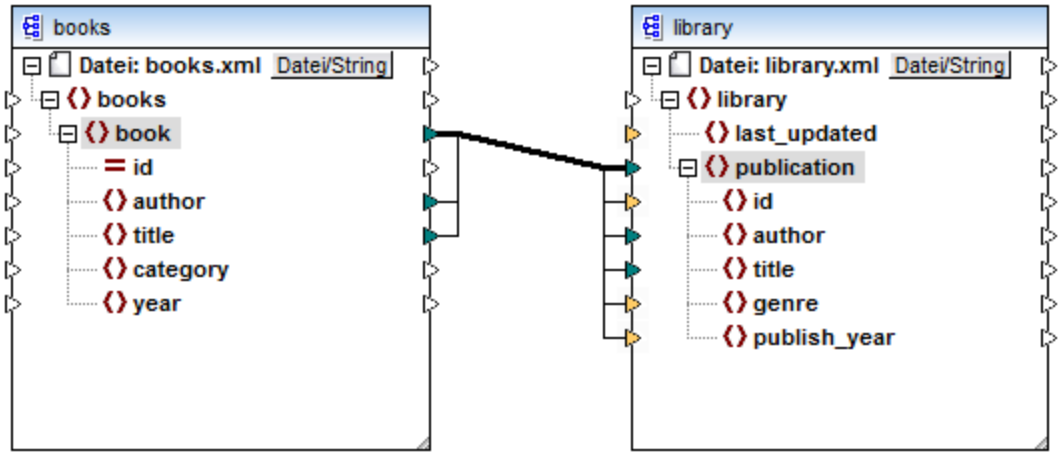
Wenn der Typ und/oder Name von Child-Nodes in der Quell- und der Zielkomponente nicht identisch ist, wird nicht automatisch eine "Alles kopieren"-Verbindung erstellt und Sie müssen diese manuell herstellen.

Manuelles Erstellen von "Alles kopieren"-Verbindungen

Um manuell eine "Alles kopieren"-Verbindung zu erstellen, gehen Sie folgendermaßen vor:

 © 2019-2025 Altova GmbH

1. Fügen Sie eine Quelldatei hinzu: Klicken Sie im Menü **Einfügen** auf **XML-Schema/Datei** und navigieren Sie zur Datei **`Books.xml`** im Ordner BasicTutorials [15].
2. Fügen Sie eine Zieldatei hinzu: Klicken Sie im Menü **Einfügen** auf **XML-Schema/Datei** und navigieren Sie zur Datei **`Library.xsd`** im selben Ordner wie **`Books.xml`**. Klicken Sie auf **Überspringen**, wenn Sie von MapForce aufgefordert werden, eine XML-Beispieldatei hinzuzufügen.
3. Mappen Sie den Node `<book>` der Komponente **`Books`** auf den Node `<publication>` der Komponente **`Library`**. Da die Struktur der Elemente `<book>` und `<publication>` nicht vollständig identisch ist, wird keine "Alles kopieren"-Verbindung erstellt. Statt dessen werden mit Hilfe der Funktion **Idente Sub-Einträge automatisch verbinden** automatisch alle Child-Nodes desselben Namens miteinander verbunden, wie im Tutorial 1 [102] beschrieben.
4. Um die automatische Verbindung in eine "Alles kopieren-Verbindung zu ändern, klicken Sie mit der rechten Maustaste auf die Verbindung zwischen `<book>` und `<publication>` und wählen Sie im Kontextmenü den Befehl **Alles kopieren (Sub-Einträge kopieren)**.
5. Daraufhin erscheint ein Fenster, in dem vorgeschlagen wird, die vorhandenen Verbindungen durch eine "Alles kopieren"-Verbindung zu ersetzen. Klicken Sie auf **OK.** Die Quell- und Zielkomponente haben nun eine "Alles kopieren"-Verbindung (*siehe Abbildung unten*).



Im obigen Mapping sind nur zwei Child-Nodes in den beiden Strukturen miteinander identisch: `<author>` und `<title>`. Daher bestehen keine "Alles kopieren"-Verbindungen zwischen diesen Nodes. Child-Nodes, die nicht identisch sind, können nicht miteinander verbunden werden. In der Abbildung sehen Sie, dass `id` nicht in die "Alles kopieren"-Verbindung miteinbezogen wird, da der Typ in der Quell- und Zielkomponente nicht derselbe ist: `id` ist in der Quellkomponente ein Attribut und in der Zielkomponente ein Element. Wenn Sie versuchen, zwischen nicht identischen Nodes wie z.B. zwischen `<category>` und `<genre>` eine Verbindung zu erstellen, fragt MapForce Sie, ob diese Verbindung ersetzt oder der Input dupliziert werden soll.

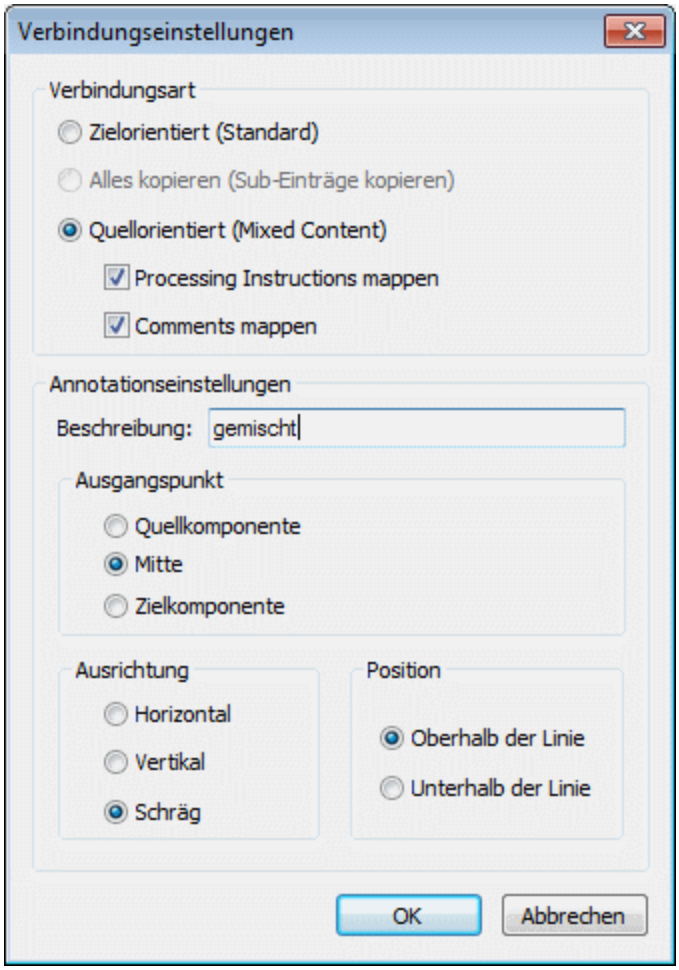
Eine Duplizierung des Input [39] ist nur dann sinnvoll, wenn die Zielkomponente Daten aus mehr als einem Input erhalten soll, was hier nicht erforderlich ist. Wenn Sie die "Alles kopieren"-Verbindung ersetzen, werden Sie in einem Meldungsfeld noch einmal aufgefordert, die "Alles kopieren"-Verbindung aufzulösen oder zu löschen. Klicken Sie auf **"Alles kopieren"-Verbindung auflösen**, wenn Sie die "Alles kopieren"-Verbindung durch einzelne zielorientierte Verbindungen [49] ersetzen möchten. Wenn Sie die "Alles kopieren"-Verbindung lieber vollständig entfernen möchten, klicken Sie auf **Child-Verbindungen löschen**.

Achtung

Bei der Erstellung von "Alles kopieren"-Verbindungen zwischen einem Schema und einem Parameter einer benutzerdefinierten Funktion 219 muss den beiden Komponenten dasselbe Schema zugrunde liegen. Die beiden Komponenten müssen dabei aber nicht dasselbe Root-Element haben.

2.2.2 Verbindungseinstellungen

Im Dialogfeld **Verbindungseinstellungen** können Sie die Einstellungen einer Verbindung definieren. Um dieses Dialogfeld zu öffnen, doppelklicken Sie auf die Verbindung. Klicken Sie alternativ dazu mit der rechten Maustaste auf die Verbindung und wählen Sie im Kontextmenü den Befehl **Eigenschaften**. Die Einstellungen sind in zwei Bereiche unterteilt: "Verbindungsart" und "Annotationseinstellungen". Nähere Informationen dazu finden Sie in den Unterabschnitten weiter unten.



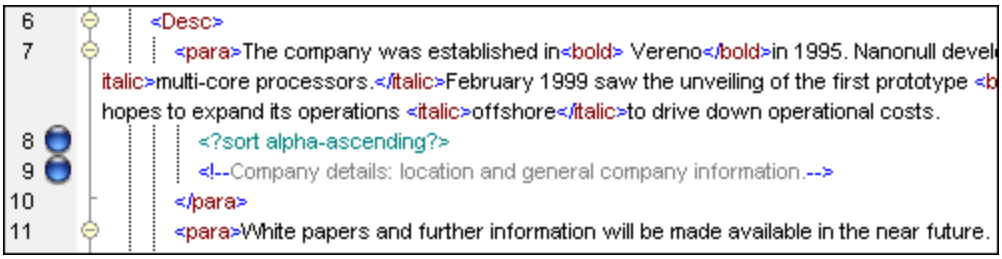
- Verbindungsarten

  Sie können eine der unten beschriebenen Verbindungsarten auswählen:

  - Zielorientierte (Standard) 49-Verbindungen eignen sich für die meisten Mapping-Szenarien.

© 2019-2025 Altova GmbH

- Alles kopieren (Sub-Einträge kopieren) [54]-Verbindungen: Wenn eine Quell- und Zielkomponente identische oder ähnliche Nodes mit identischen Child-Nodes hat, wird zwischen diesen identischen Nodes automatisch eine "Alles kopieren"-Verbindung erstellt.
- Bei einer quellorientierten (Mixed Content) [49]-Verbindung, kann gemischter Inhalt (Mixed Content) (Text- und Child-Nodes) in derselben Reihenfolge, wie er in der XML-Quelldatei vorkommt, gemappt werden. Wenn Sie die Kontrollkästchen **Processing Instructions mappen** und/oder **Comments mappen** aktivieren, werden diese Datengruppen in die Ausgabedatei inkludiert (*siehe Abbildung unten*).



Annotationseinstellungen

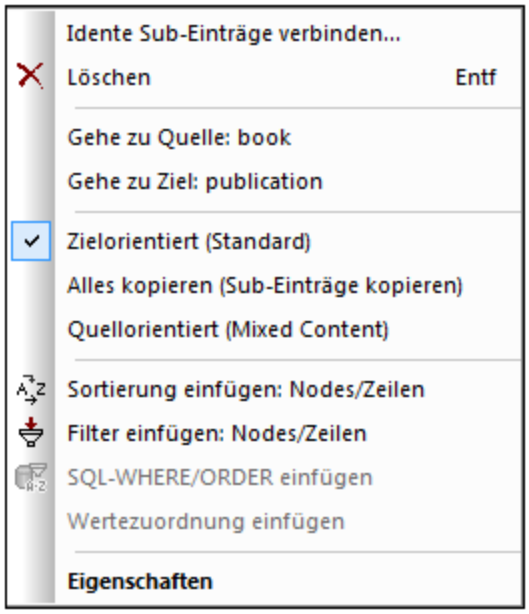
Über den Abschnitt **Annotationseinstellungen** können Sie eine Verbindung beschriften. Diese Option steht für alle Verbindungsarten zur Verfügung. Um eine Verbindung zu beschriften, gehen Sie folgendermaßen vor:

1. Klicken Sie mit der rechten Maustaste auf die Verbindung und wählen Sie im Kontextmenü den Befehl **Eigenschaften**. Doppelklicken Sie alternativ dazu auf die Verbindung.
2. Geben Sie den Namen der ausgewählten Verbindung in das Feld **Beschreibung** ein. Dadurch werden alle Optionen im Bereich **Annotationseinstellungen** aktiv.
2. In den restlichen Bereichen können Sie den **Ausgangspunkt**, die **Ausrichtung** und **Position** der Beschriftung definieren.
3. Aktivieren Sie die Symbolleisten-Schaltflächen (**Annotationen anzeigen**). Wenn die Schaltfläche in der Symbolleiste noch nicht zu sehen ist, aktivieren Sie in der Symbolleiste die Schaltfläche **Optionen anzeigen**.

Anmerkung: Wenn die Schaltfläche **Annotationen anzeigen** deaktiviert ist, können Sie den Annotationstext dennoch sehen, wenn Sie den Mauszeiger über die Verbindung platzieren. Die Annotation wird als Tooltip angezeigt, wenn die Schaltfläche (**Tipps anzeigen**) in der Symbolleiste **Anzeigeoptionen** aktiv ist.

2.2.3 Kontextmenü für Verbindungen

In diesem Kapitel werden Befehle aus dem Kontextmenü für Verbindungen beschrieben. Wenn Sie mit der rechen Maustaste auf eine Verbindung klicken, stehen die folgenden Kontextmenübefehle zur Verfügung:



Nähere Informationen dazu finden Sie in den Unterabschnitten weiter unten.

Allgemeine Einstellungen

- *Idente Sub-Einträge verbinden:* Öffnet das Dialogfeld Idente Sub-Einträge verbinden 52. Dieser Befehl ist aktiv, wenn die Verbindung idente Sub-Einträge haben kann.
- *Löschen:* Löscht die markierte Verbindung.
- *Gehe zu Quelle: <Datenelementname>* Markiert den Output 30-Konnektor der ausgewählten Verbindung.
- *Gehe zu Ziel: <Datenelementname>* Markiert den Input 30-Konnektor der ausgewählten Verbindung.

Verbindungsarten

Nähere Informationen zu den Verbindungsarten finden Sie unter Verbindungsarten 49 und Verbindungseinstellungen 56.

Einfügebefehle

- *Sortierung einfügen: Nodes/Zeilen:* Fügt zwischen einem Quell- und Ziel-Node eine Sortierkomponente 188 ein.
- *Filter einfügen: Nodes/Zeilen:* Fügt zwischen einem Quell- und Ziel-Node eine Filter 194-Komponente ein.
- *SQL/NoSQL-WHERE/ORDER einfügen:* Fügt zwischen einem Quell- und Ziel-Node eine SQL/NoSQL-WHERE/ORDER-Komponente ein (*Professional und Enterprise Edition*).
- *Wertezuordnung einfügen:* Fügt zwischen einem Quell- und Ziel-Node eine Wertezuordnung 200 ein.

Eigenschaften

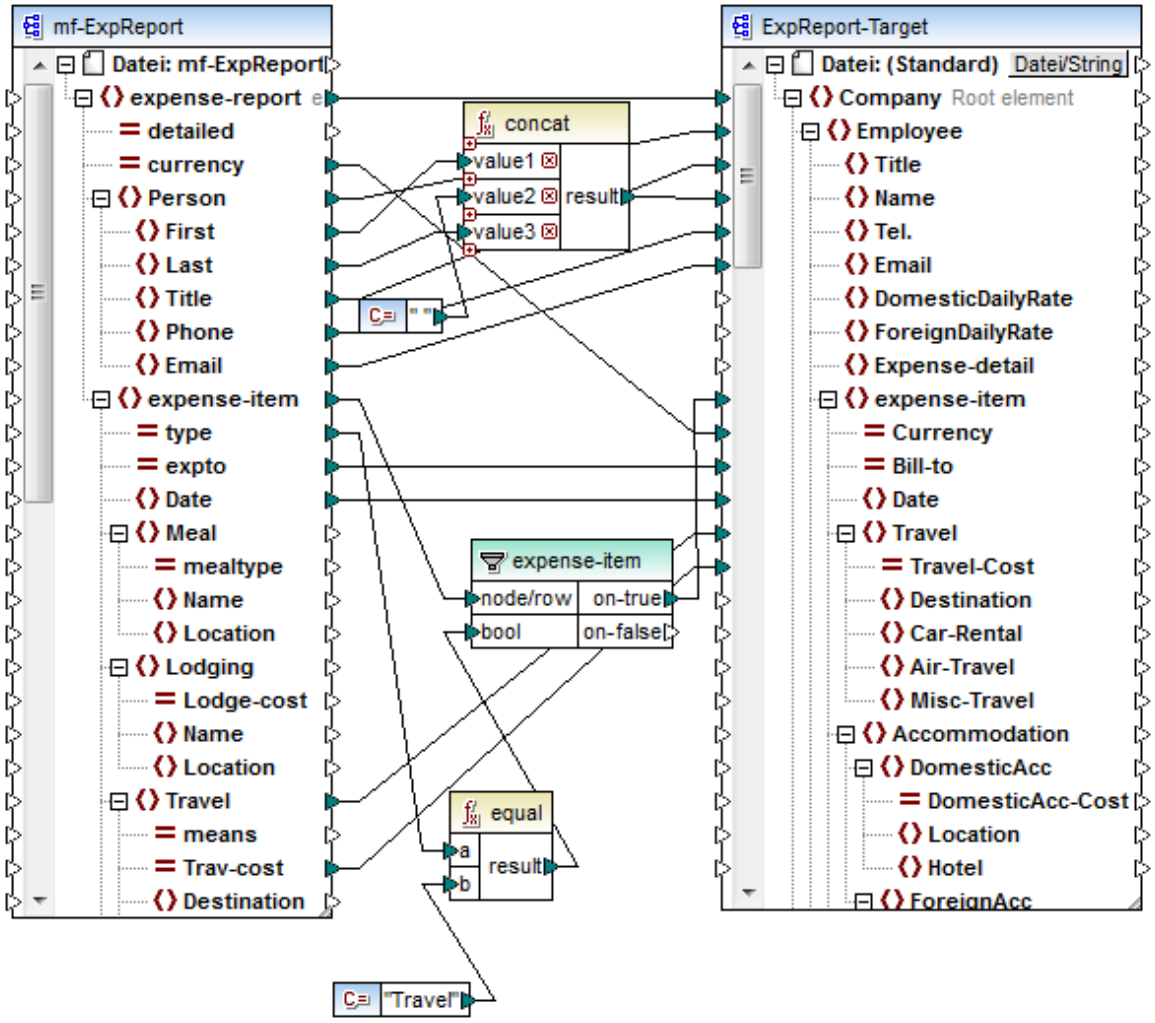
Öffnet das Dialogfeld Verbindungseinstellungen 56.

 © 2019-2025 Altova GmbH

2.2.4 Fehlerhafte Verbindungen

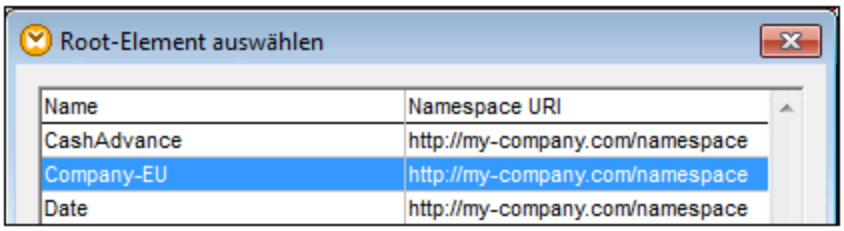
Manchmal muss ein Schema in einer Quell- oder Zielkomponente geändert werden. Änderungen an einem Schema können sich auf die Gültigkeit Ihres Mappings auswirken und zu einer Reihe fehlerhafter Verbindungen führen. In diesem Kapitel wird erläutert, wie Sie solche Verbindungen reparieren, nachdem Sie die Schema-Datei geändert haben. Befolgen Sie die Anleitung im Beispiel unten, um zu sehen, wie Sie fehlerhafte Verbindungen reparieren können.

1. Öffnen Sie **`Tut-ExpReport.mfd`** aus dem Tutorial-Ordner[15]. Unterhalb sehen Sie einen Ausschnitt aus diesem Mapping.

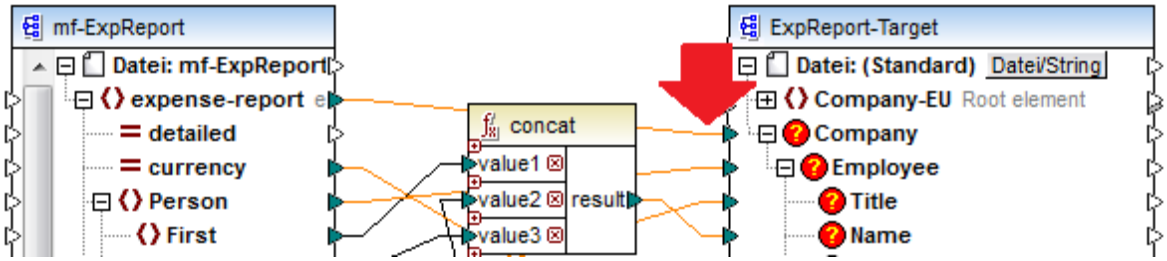


2. Öffnen Sie **`ExpReport-Target.xsd`** in einem Editor (z.B., Altova XMLSpy) und ändern Sie das Root-Element `Company` im Ziel-Schema in `Company-EU`. Sie müssen MapForce dazu nicht schließen.
3. Nachdem Sie das Root-Element des Ziel-Schemas bearbeitet haben**,** wird in MapForce eine **Meldung über geänderte Dateien** angezeigt. Klicken Sie auf die Schaltfläche **Neu laden**. Da das Root-Element geändert wurde, werden in der Komponente mehrere fehlende Nodes angezeigt.

4. Klicken Sie oben in der Komponente auf **Neues Root-Element auswählen** (*siehe Abbildung unten*). Sie können das Root-Element auch ändern, indem Sie mit der rechten Maustause auf die Titelleiste der Komponente klicken und im Kontextmenü den Befehl **Root-Element ändern** auswählen.



5. Wählen Sie als neues Root-Element `Company-EU` aus und klicken Sie auf **OK**. Das Root-Element `Company-EU` wird nun auf der obersten Ebene der Komponente angezeigt.
6. Sie müssen die Verbindung nun vom fehlenden Node `Company` auf das neue Root-Element verschieben. Klicken Sie auf den dicken Abschnitt der Verbindung von `Company` (*siehe roter Pfeil unten*). Ziehen Sie anschließend die Verbindung auf das Root-Element `Company-EU`.



Daraufhin wird ein Dialogfeld mit der Frage angezeigt, ob Sie alle übereinstimmenden verbundenen Child-Nodes verschieben möchten. Sie können wählen, ob Sie nur die ausgewählte Verbindung oder die ausgewählte Verbindung mit ihren Child-Nodes, die mit den Child-Nodes im neuen Root-Element übereinstimmen, verschoben werden soll. In unserem Beispiel haben wir die Option **Child-Verbindungen inkludieren** gewählt. Sobald Sie auf diese Schaltfläche klicken, verschwinden alle fehlerhaften Nodes aus der Komponente.

Anmerkung: Wenn der Node, auf den gemappt werden soll, zwar denselben Namen wie der Quell-Node, aber einen anderen Namespace hat, wird im Benachrichtigungsdialogfeld eine zusätzliche Schaltfläche **Child-Verbindungen inkludieren und Namespace mappen** angezeigt. Wenn Sie auf diese Schaltfläche klicken, werden Child-Verbindungen desselben Namespace wie die des Parent-Node der Quellkomponente auf dieselben Child-Nodes unter dem Node des anderen Namespace verschoben.

Alternative Lösung

Eine andere Lösung für das oben beschriebene Problem wäre, die in Ihrem Mapping nicht mehr benötigten fehlerhaften Nodes zu löschen. Wenn Sie z.B. die Verbindung zwischen der **`concat`**-Funktion und `Name` löschen, verschwindet der Node `Name` aus der `ExpReport-Target`-Komponente.

Fehlerhafte Verbindungen in Datenbanken (Professional und Enterprise Edition)

Wenn Ihre Datenbankkomponente fehlerhafte Verbindungen aufweist, müssen Sie die Komponenteneinstellungen ändern 38. Wenn Sie im Dialogfeld **Komponenteneinstellungen** auf die Schaltfläche **Ändern** klicken, können Sie eine andere Datenbank auswählen oder Tabellen in Ihrer Datenbankkomponente ändern. Alle gültigen/korrekten Verbindungen und relevanten Datenbankdaten bleiben erhalten, wenn Sie eine Datenbank mit derselben Struktur auswählen.

 © 2019-2025 Altova GmbH

2.2.5 Beibehalten von Verbindungen nach Löschen von Komponenten

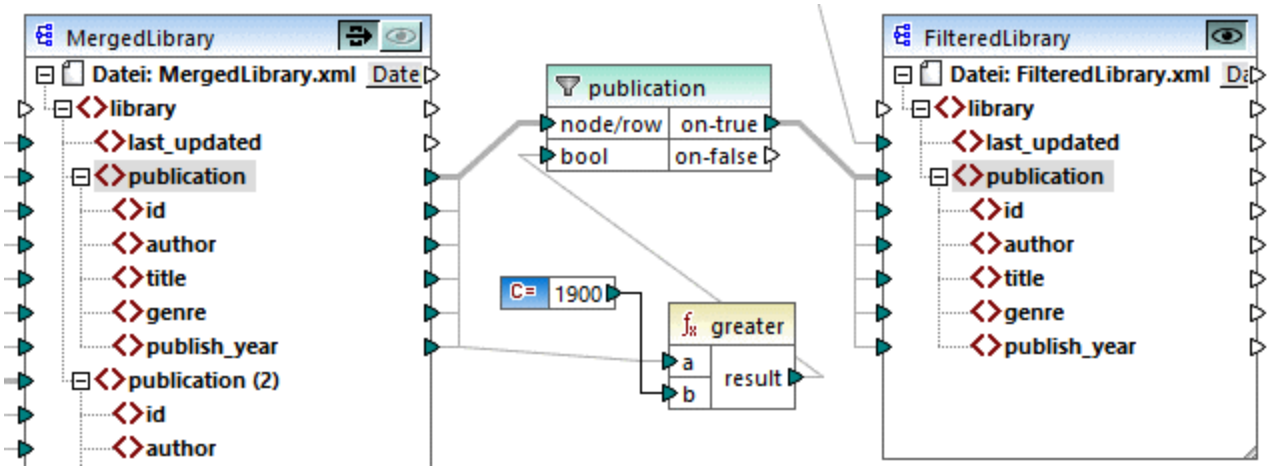
Sie können in MapForce Verbindungen auch nach Löschen einiger Transformationskomponenten 31 (z.B. Variablen, Sortier- und Filter-Komponenten, Wertezuordnungen, einfache Input-Komponenten, SQL/NoSQL-WHERE/ORDER-Komponenten) beibehalten. Bei Verbindungen kann es sich um Einfach- und Mehrfachverbindungen handeln. Vor allem die Beibehaltung von Verbindungen mit mehreren Child-Verbindungen kann sich als nützlich erweisen, da Sie dadurch nach Löschung einer Transformationskomponente nicht jede einzelne Child-Verbindung manuell wiederherstellen müssen. Um diese Option zu aktivieren, wählen Sie **Extras | Optionen | Bearbeiten** und aktivieren Sie **Intelligente Komponentenlöschung (nützliche Verbindungen beibehalten)**. Diese Option ist standardmäßig deaktiviert, sodass bei Löschung einer Transformationskomponente auch die direkten Verbindungen dazu gelöscht werden.

Beispiel

Zur Veranschaulichung einer intelligenten Komponentenlöschung werfen Sie einen Blick in die Beispieldatei **`Tut3-ChainedMapping`**. Die Beispieldatei steht im Ordner BasicTutorials 15 zur Verfügung.

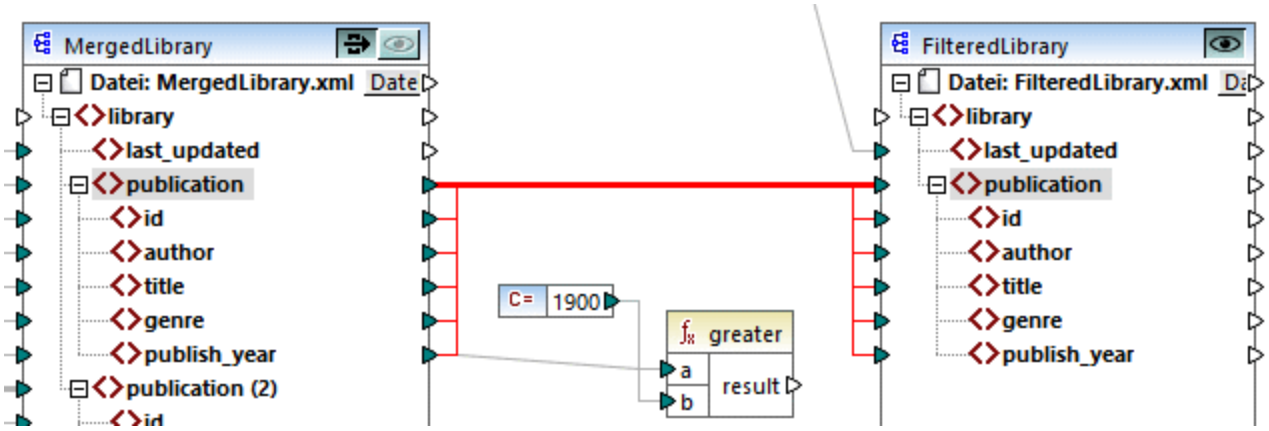
Vor der Löschung

In der Abbildung unten sehen Sie, dass zwischen der Komponente `MergedLibrary` und dem Filter `publication` und zwischen dem Filter `publication` und der Komponente `FilteredLibrary` Alles kopieren-Verbindungen 54 bestehen. Der Filter `publication` soll nun gelöscht werden, aber die "Alles kopieren"-Verbindungen sollen bestehen bleiben. Aktivieren Sie nun im Dialogfeld **Optionen** das Kontrollkästchen **Intelligente Komponentenlöschung** (*siehe oben*).



Nach der Löschung

Nachdem die `publication`-Funktion gelöscht wurde, wird die "Alles kopieren"-Verbindung direkt zwischen dem Node `publication` in `MergedLibrary` und dem Node `publication` in `FilteredLibrary` erstellt (*siehe Abbildung unten*).



Anmerkung: Wenn bei einer Filter-Komponente sowohl der `on-true` als auch der `on-false`-Output verbunden ist, bleiben die Verbindungen beider Outputs erhalten.

© 2019-2025 Altova GmbH

2.3 Allgemeine Verfahren und Funktionalitäten

Neben der Erstellung von Mappings stehen auch Funktionen zum Validieren Ihres Mappings und Ihrer Ausgabe, zum Generieren von Code, zur Verwendung der Textansicht und zum Definieren von Mapping-Einstellungen zur Verfügung. Dieser Abschnitt ist in die folgenden Kapitel gegliedert:

- Validierung 63
- Codegenerierung 65
- Funktionalitäten der Textansicht 67
- Suchen in der Textansicht 71
- Mapping-Einstellungen 74

2.3.1 Validierung

In diesem Kapitel wird erläutert, wie Sie Mappings validieren. Außerdem erfahren Sie hier, wie Sie eine Vorschau auf die Ausgabe anzeigen und diese speichern und validieren.

Validieren von Mappings

MapForce validiert Mappings automatisch, wenn Sie auf das Register **Ausgabe** klicken. Sie können Ihr Mapping auch manuell validieren. Auf diese Art können Sie potenzielle Fehler und Warnungen ausfindig machen und korrigieren, bevor das Mapping ausgeführt wird. Um ein Mapping manuell zu validieren, klicken Sie in den **Mapping**-Bereich und wählen Sie eine der folgenden Methoden:

- Klicken Sie im Menü **Datei** auf **Mapping validieren**.
- Klicken Sie in der Symbolleiste auf (**Validieren**).

Bei der Validierung eines Mappings überprüft MapForce das Mapping auf nicht unterstützte Komponentenarten, falsche oder fehlende Verbindungen. Nähere Informationen zum Validierungsstatus finden Sie unter Fenster "Meldungen" 24. Außerdem können Sie im Fenster **Meldungen** Aktionen im Zusammenhang mit Meldungen 25 durchführen. Um das Ergebnis jeder einzelnen Validierung auf einem eigenen Register anzuzeigen, klicken Sie auf der linken Seite des **Meldungsfensters** auf die nummerierten Register. Dies ist z.B. hilfreich, wenn Sie gleichzeitig mit mehreren Mappings arbeiten.

Validierung von Transformationskomponenten
Transformationskomponenten 32 werden folgendermaßen validiert:

- Wenn ein zwingend erforderlicher **Input-Konnektor** nicht verbunden ist, wird eine Fehlermeldung generiert und die Transformation wird gestoppt.
- Wenn ein **Output-Konnektor** nicht verbunden ist, wird eine Warnmeldung generiert und die Transformation wird fortgesetzt. Die Komponente, die die Warnung verursacht hat, und ihre Daten werden ignoriert und nicht auf die Zielkomponente gemappt.

Anzeige einer Vorschau und Validieren der Ausgabe

Sie können in MapForce eine Vorschau der Ausgabe anzeigen, ohne den generierten Code mit einem externen Prozessor oder Compiler ausführen und kompilieren zu müssen. Bevor Sie den generierten Code extern

verarbeiten, sollten Sie eine Vorschau der Transformationsausgabe in MapForce anzeigen. Bei einer Vorschau auf das Mapping-Ergebnis führt MapForce das Mapping aus und zeigt die Ausgabe im Ausgabefenster 27 an.

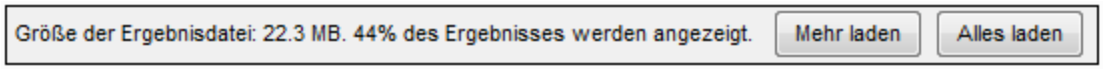
Sobald die Daten im **Ausgabefenster** zur Verfügung stehen, können Sie diese validieren und gegebenenfalls speichern. Außerdem können Sie mit dem Befehl **Suchen** (**Strg + F)** schnell nach einem bestimmten Textmuster in der Ausgabedatei suchen. Nähere Informationen dazu finden Sie unter Suchen in der Textansicht 71. Alle Fehler, Warnungen oder Informationsmeldungen im Zusammenhang mit der Mapping-Ausführung werden im Fenster "Meldungen" 24 angezeigt.

Bei Auswahl von C++, C# oder Java (*Professional und Enterprise Edition*) als Transformationssprache 16, führt MapForce das Mapping mit dem Built-in-Transformationsprozessor aus und zeigt das Ergebnis im **Ausgabefenster** an.

Um die Transformationsausgabe zu speichern, klicken Sie in den **Mapping**-Bereich und wählen Sie eine der folgenden Methoden:

- Klicken Sie im Menü **Ausgabe** auf **Ausgabedatei speichern**.
- Klicken Sie in der Symbolleiste auf **(Generierte Ausgabe speichern)**.

Optionen zum Laden
Bei großen Ausgabedateien beschränkt MapForce die Menge der im **Ausgabefenster** angezeigten Daten, d.h. es wird nur ein Teil der Daten im Ausgabefenster angezeigt. In diesem Fall erscheint im unteren Bereich eine Schaltfläche **Mehr laden** (*siehe Abbildung unten*). Wenn Sie darauf **klicken, wird der nächste Teil angezeigt. Die Vorschaueinstellungen können im Dialogfeld Optionen** auf dem Register **Allgemein** konfiguriert werden. Nähere Informationen dazu finden Sie unter MapForce Optionen 508.

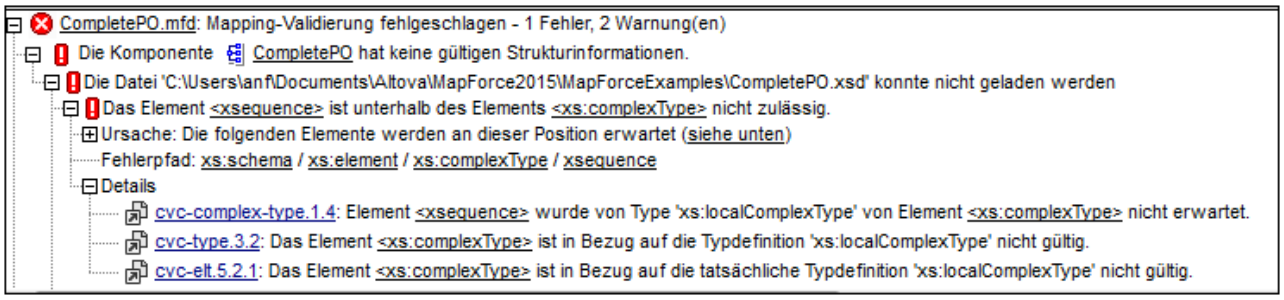


Validieren der Ausgabe
Sobald die Ausgabe im Fenster **Ausgabe** zur Verfügung steht, können Sie diese anhand des damit verknüpften Schemas validieren. Anmerkung: Die Schaltfläche **Ausgabe validieren** und der entsprechende Menübefehl (**Ausgabe | Ausgabedatei validieren**) sind nur aktiv, wenn die Ausgabedatei die Validierung anhand eines Schemas unterstützt. Das Ergebnis der Validierung wird im Fenster **Meldungen** angezeigt. Wählen Sie eine der folgenden Methoden, um die Ausgabe zu validieren:

- Öffnen Sie den **Ausgabebereich** und klicken Sie in der Symbolleiste auf (**Ausgabe validieren**).
- Öffnen Sie den **Ausgabebereich** und klicken Sie im Menü **Ausgabe** auf **Ausgabedatei validieren**.

In der Abbildung unten sehen Sie eine nicht erfolgreiche Validierung. Das Fenster **Meldungen** enthält ausführliche Informationen über die Fehler. Wenn Sie z.B. auf den `<Name>`-Link klicken, wird dieses Element im **Ausgabebereich** markiert.

 © 2019-2025 Altova GmbH



2.3.2 Codegenerierung

Code Generator ist eine integrierte MapForce-Funktionalität, mit Hilfe derer Sie Code anhand von Mapping-Dateien generieren können. Mit Hilfe des generierten Codes könne Sie Ihre Mappings außerhalb vom MapForce ausführen und somit Ihre Mappings automatisieren. Code kann in den folgenden Datentransformationssprachen [16] generiert werden:

- XSLT 1.0/XSLT 2.0/XSLT 3.0 (*Alle Editionen*)
- XQuery (*Professional und Enterprise Edition*)
- Java (*Professional und Enterprise Edition*)
- C# (*Professional und Enterprise Edition*)
- C++ (*Professional und Enterprise Edition*)

Sie können Code entweder anhand eines einzigen Mapping-Designs (**`.mfd`**) oder anhand eines Mapping-Projekts (**`.mfp`**) generieren. Die Generierung von Code anhand eines Projekts wird nur in der Professional und der Enterprise Edition unterstützt. Nähere Informationen dazu finden Sie in den Unterabschnitten weiter unten.

Wichtige Punkte
Beachten Sie bei der Codegenerierung die folgenden Aspekte:

- Bestimmte MapForce-Funktionalitäten werden in generiertem Programmcode nicht unterstützt. Nähere Informationen dazu finden Sie unter Unterstützte Funktionalitäten im generierten Code [525].
- Informationen zur Behandlung von Pfaden im generierten Code finden Sie unter Pfade in Ausführungsumgebungen [43].
- *Professional und Enterprise Edition:* Sie können die allgemeinen Codegenerierungsoptionen im Abschnitt *Code-Generierung* des Dialogfelds **Optionen** ändern.
- *Professional und Enterprise Edition:* Je nach Plattform variiert die Unterstützung für Datenbankverbindungen und es gibt Verbindungsarten, die nicht auf allen Plattformen unterstützt werden. Wenn Sie in Ihrem Mapping eine Verbindung zu einer Datenbank herstellen, wählen Sie eine Datenbankverbindung, die mit der Zielumgebung, für die Sie Code generieren, kompatibel ist.

Informationen zur Unterstützung
Die nachstehende Tabelle enthält einen Überblick über die Unterstützung für C++, C# und Java.

Zielsprache	C++	C#	Java
Entwicklungsumgebungen	Microsoft Visual Studio 2013, 2015, 2017, 2019, 2022	Microsoft Visual Studio 2013, 2015, 2017, 2019, 2022 Ziel-Frameworks: • .NET Framework • .NET Core 3.1 • .NET 5.0 • .NET 6.0 • .NET 8.0	Java SE JDK 8, 11, 17, 21 (einschließlich OpenJDK) Eclipse 4.4 oder höher Apache Ant
XML DOM Implementierungen	MSXML 6.0 Apache Xerces 3	System.Xml	JAXP
Datenbank API	ADO	ADO.NET	JDBC

Generieren von Code anhand eines Mappings

Um Code anhand einer Mapping-Designs (`.mfd`) zu generieren, gehen Sie folgendermaßen vor:

1. Wählen Sie im Abschnitt *Code-Generierung* des Dialogfelds **Optionen** und in den Mapping-Einstellungen 74 die entsprechenden Codegenerierungsoptionen aus (gilt für C# und C++).
2. Klicken Sie auf **Datei | Code generieren in** und wählen Sie die gewünschte Transformationssprache aus. Klicken Sie alternativ dazu auf **Datei | Code in ausgewählter Sprache generieren**. In diesem Fall wird der Code in der in der Symbolleiste ausgewählten Sprache generiert.
3. Wählen Sie ein Zielverzeichnis für die generierten Dateien aus und klicken Sie anschließend zur Bestätigung auf **OK**. MapForce generiert den Code und zeigt das Ergebnis der Operation im Fenster Meldungen 24 an.

Generieren von Code anhand eines Projekts (Professional und Enterprise Edition)

Sie können anhand eines Mapping-Projekts (`.mfp`), das aus mehreren Mapping-Design-Dateien (`.mfd`) besteht, Code generieren. Beachten Sie, dass alle Mapping-Design-Dateien im Projekt für die Codegenerierung geeignet sein müssen, d.h. alle ihre Komponenten müssen in der ausgewählten Transformationssprache unterstützt werden, wie unter Unterstützte Funktionalitäten im generierten Code 525 beschrieben.

Um Code anhand eines Mapping-Projekts zu generieren, gehen Sie folgendermaßen vor:

1. Öffnen Sie das entsprechende Mapping-Projekt, anhand dessen Sie Code generieren möchten.
2. Klicken Sie mit der rechten Maustaste im Fenster Projekt auf den Projektnamen und wählen Sie im Kontextmenü den Befehl **Eigenschaften**. Klicken Sie alternativ dazu auf den Projektnamen und wählen Sie den Menübefehl **Projekt | Eigenschaften**.
3. Überprüfen Sie die Projekteinstellungen und ändern Sie sie gegebenenfalls. Stellen Sie v.a. sicher, dass die Zielsprache und das Ausgabeverzeichnis korrekt eingestellt wurden. Klicken Sie anschließend auf **OK**.
4. Klicken Sie im Menü **Projekt** auf den Befehl **Code für das gesamte Projekt generieren**.

 © 2019-2025 Altova GmbH

Unabhängig von der im Dialogfeld **Projekteigenschaften** ausgewählten Sprache können Sie jederzeit Projektcode in einer anderen Sprache generieren, indem Sie den Menübefehl **Projekt | Code generieren in | <Sprache>** auswählen.

Der Fortschritt und das Ergebnis der Codegenerierung werden im Fenster "Meldungen" angezeigt. Standardmäßig ist der Name der generierten Applikation mit dem Projektnamen identisch. Wenn der Projektname Leerzeichen enthält, werden diese im generierten Code in Unterstriche konvertiert. Standardmäßig wird Code im selben Verzeichnis wie das MapForce-Projekt generiert, nämlich im untergeordneten **`Ausgabeverzeichnis`**.

Sie können das Ausgabeverzeichnis und/oder den Namen des Projekts im Dialogfeld **Projekteigenschaften** ändern. Wenn Ihr MapForce-Projekt Ordner enthält, können Sie die Codegenerierungseinstellungen für die einzelnen Ordner konfigurieren. Klicken Sie mit der rechten Maustaste auf den gewünschten Ordner und wählen Sie im Kontextmenü den Befehl **Eigenschaften**. Andernfalls erben alle Projektordner die auf der obersten Ebene definierten Einstellungen.

Nächste Schritte

Je nachdem, welche Transformationssprache Sie für die Codegenerierung gewählt haben, unterscheiden sich die nächsten Schritte. Wenn Sie Code in XSLT 1-3 oder XQuery generiert haben, wird im nächsten Schritt die Transformation über die Befehlszeile ausgeführt (*nähere Informationen siehe unten*).

Wenn Sie Java-, C#- oder C++-Code generiert haben, wird in den nächsten Schritten der generierte Code gebaut und ausgeführt. Sie können den generierten Java-, C#- und C++-Code auch ändern und in Ihren benutzerdefinierten Code integrieren.

XSLT- und XQuery-Code
Nachdem Sie XSLT 1-3-Code generiert haben, enthält der Zielordner die folgenden Dateien:

1. Eine XSLT-Transformationsdatei mit dem folgenden Format: **`<Mapping>MapTo<Zieldateiname>.xslt`**. `<Mapping>` ist der Wert des Felds *Applikationsname* in den Mapping-Einstellungen 74. `<Zieldateiname>` ist der Name der Zielkomponente. Öffnen Sie die Einstellungen der Zielkomponente und bearbeiten Sie den Wert des Felds *Komponentenname*, um den Wert zu ändern. Nähere Informationen dazu finden Sie unter Ändern der Komponenteneinstellungen 38 und Bibliothekspfade im generierten Code 43.
2. Die Datei **`DoTransform.bat`**, mit der Sie die XSLT-Transformation mit Altova RaptorXML Server über die Befehlszeile ausführen können. Um den Befehl ausführen zu können, müssen Sie RaptorXML installieren.

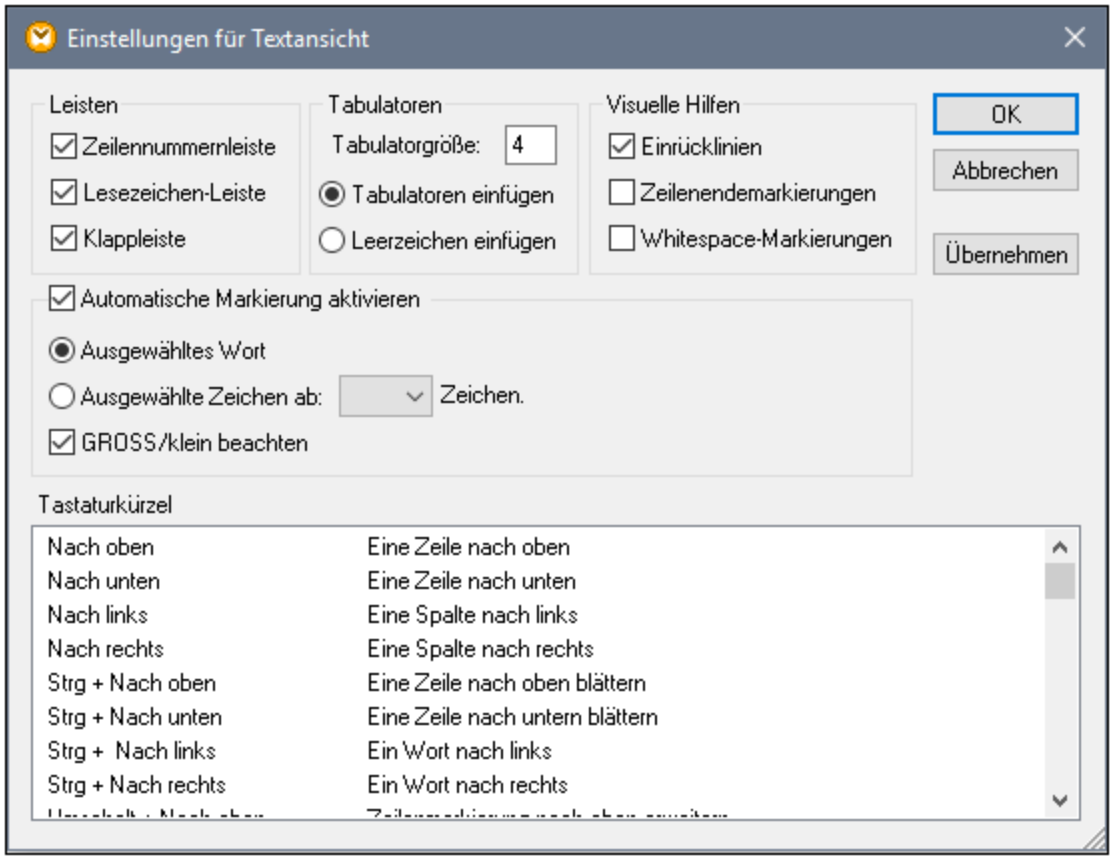
Wenn es sich um ein verkettetes 111 Mapping handelt, wird für jede Zielkomponente eine separate Transformationsdatei generiert.

Die XQuery-Codegenerierung ähnelt der XSLT-Codegenerierung, mit dem Unterschied, dass die Transformationsdatei(e) die Erweiterung **`.xq`** und das folgende Format haben: **`<Mapping>MapTo<TargetFileName>.xq`**.

2.3.3 Funktionalitäten der Textansicht

Die Fenster Ausgabe 27 und XSLT 26 verfügen über eine Reihe von visuellen Hilfsmitteln für die Textanzeige, wie Leisten, Textmarkierung, Einrücklinien, Zeilenendezeichen und Whitespace-Markierungen. Sie können

diese Funktionalitäten im Dialogfeld **Einstellungen für die Textansicht** anpassen (*siehe Abbildung unten*). Die Einstellungen in diesem Dialogfeld gelten für die gesamte Applikation.



Um das Dialogfeld **Einstellungen für Textansicht** zu öffnen, wählen Sie eine der folgenden Methoden:

- Wählen Sie **Ausgabe** | **Einstellungen für Textansicht**.
- Klicken Sie in der Symbolleiste auf (**Einstellungen für Textansicht**).
- Klicken Sie mit der rechten Maustaste in einen leeren Bereich des **Ausgabefensters** und wählen Sie im Kontextmenü den Befehl **Einstellungen für Textansicht**.

Einige der Navigationshilfen können auch über die Symbolleiste **Textansicht**, das Applikationsmenü oder Tastaturkürzel ein- und ausgeschaltet werden. Nähere Informationen zu Tastaturkürzeln finden Sie im Abschnitt **Tastaturkürzel** des oben gezeigten Dialogfelds **Einstellungen für Textansicht**.

Siehe die Liste der verfügbaren Einstellungen unten.

⊟ Leisten

Zeilennummernleiste
Die Zeilennummern werden in der Zeilennummernleiste angezeigt, die über das Dialogfeld **Einstellungen für Textansicht** ein- und ausgeblendet werden können. Wenn ein Textabschnitt eingeklappt ist, werden auch die Zeilennummern der entsprechenden Textzeilen ausgeblendet.

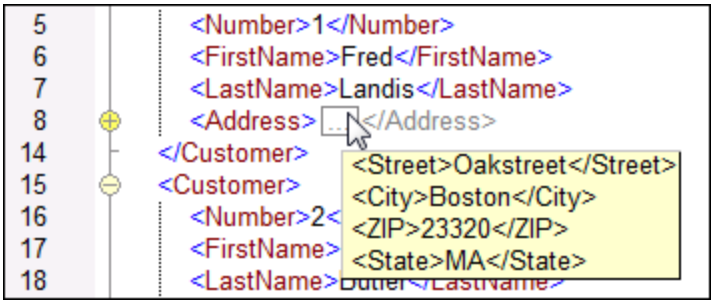
 © 2019-2025 Altova GmbH

Lesezeichenleiste
Für den raschen Zugriff darauf können Zeilen in einem Dokument mit Lesezeichen versehen werden. Wenn im Dialogfeld **Einstellungen für Textansicht** das Kontrollkästchen **Lesezeichenleiste** aktiviert ist, werden die Lesezeichen in der Lesezeichenleiste angezeigt (*siehe Abbildung unten*). Wenn das Kontrollkästchen **Lesezeichenleiste** nicht aktiviert ist, werden mit Lesezeichen versehene Zeilen in Zyan markiert.

Die Bearbeitung von und Navigation zwischen Lesezeichen erfolgt über die Befehle in der Tabelle unten. Diese Befehle stehen im Menü **Ausgabe** zur Verfügung. Außerdem finden Sie die Lesezeichenbefehle im Kontextmenü, wenn Sie mit der rechten Maustaste in das Fenster **Ausgabe**, **XSLT** oder **XQuery** klicken.

	Lesezeichen einfügen/löschen (Strg + F2)
	Nächstes Lesezeichen (F2)
	Vorhergehendes Lesezeichen (Umschalt + F2)
	Alle Lesezeichen löschen (Strg + Umschalt + F2)

Klappleiste
Mit Hilfe der Klappleiste können Sie Nodes erweitern und reduzieren. Diese Funktionalität wird in der der Klappleiste angezeigt. Die Klappleiste kann über das Dialogfeld **Einstellungen für Textansicht** ein- und ausgeblendet werden. Um Textabschnitte ein- oder auszuklappen, klicken Sie auf das "+" bzw."-"-Symbol am linken Fensterrand. Eingeklappte Codeabschnitte werden mittels Auslassungspunkten markiert (*siehe Abbildung unten*). Um eine Vorschau des eingeklappten Abschnitts zu sehen, ohne diesen Abschnitt ausklappen zu müssen, platzieren Sie die Mauszeiger über die Auslassungspunkte. Daraufhin wird ein Tooltipp mit der Codevorschau angezeigt, wie in der Abbildung unten gezeigt. Wenn der Abschnitt zu groß für die Vorschau ist, wird am Ende des Tooltipps ein weiteres Auslassungssymbol angezeigt.



Automatische Markierung aktivieren

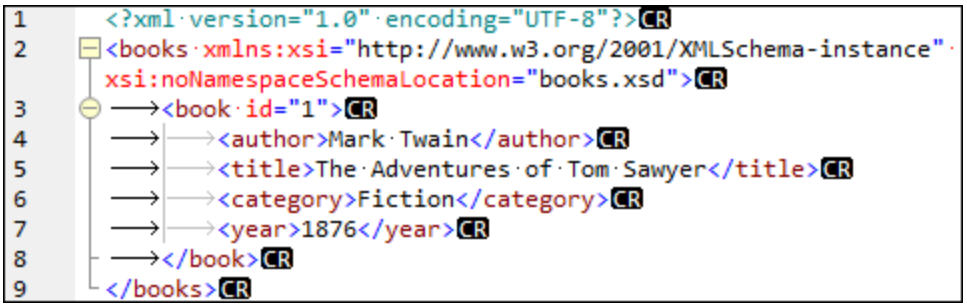
Mit der Einstellung **Automatische Markierung aktivieren** können Sie alle Übereinstimmungen mit dem ausgewählten Textabschnitt anzeigen. Der ausgewählte Text wird hellblau markiert und die Übereinstimmungen erscheinen hellbraun markiert. Die Auswahl und die Übereinstimmungen werden in der Bildlaufleiste durch graue Markierungsquadrate gekennzeichnet. Die aktuelle Cursorposition wird in der Bildlaufleiste durch die blaue Cursormarkierung markiert. Als Auswahl kann ein ganzes Wort oder eine festgelegte Anzahl von Zeichen definiert werden. Sie können außerdem definieren, ob die Groß- und Kleinschreibung berücksichtigt werden soll.

Bei Auswahl einer Anzahl von Zeichen können Sie die Mindestanzahl der Zeichen ab dem ersten Zeichen in der Auswahl, für die eine Übereinstimmung gefunden werden soll, definieren. So können Sie z.B. nach zwei oder mehr Zeichen suchen. Bei Wortsuchen werden folgende Einträge als separate Wörter behandelt: Elementnamen (ohne spitze Klammern), die spitzen Klammern von Element-Tags, Attributnamen und Attributwerte ohne Anführungszeichen.

Visuelle Hilfen

Einrücklinien
Einrücklinien sind vertikal Linien, anhand derer Sie den Einrückungsgrad einer Zeile sehen können. Sie können im Dialogfeld **Einstellungen für Textansicht** ein- und ausgeblendet werden. Die Optionen **Tabulatoren einfügen** und **Leerzeichen einfügen** werden wirksam, wenn Sie die Option **Ausgabe | Pretty-Print** verwenden.

Zeilenendezeichen und Whitespace-Markierungen
Zeilenendezeichen und Whitespace-Markierungen (*siehe Abbildung unten*) können im Dialogfeld **Einstellungen für Textansicht** ein- und ausgeblendet werden. Die Pfeile stehen für Tabulatorzeichen. Die Abkürzung *CR* steht für Carriage Return (Wagenrücklauf). Die Punkte stehen für Leerzeichen.



Weitere Einstellungen der Textansicht

Syntaxfärbung
Eine weitere visuelle Hilfe ist die Syntaxfärbung. Sie macht Codefragmente übersichtlicher. Die Syntaxfärbung richtet sich nach dem semantischen Wert des Texts. So hat der Node-Name (und in einigen Fällen der Node-Inhalt) z.B. in XML-Dokumenten, je nachdem, ob es sich beim XML-Node um ein Element, Attribut, Inhalt, einen CDATA-Abschnitt, Kommentar oder eine Processing Instruction handelt, eine andere Farbe.

Vergrößern und Verkleinern
Durch Scrollen mit der Maus und gleichzeitiges Gedrückthalten der **Strg**-Taste können Sie in die Textansicht hinein- und daraus herauszoomen. Drücken Sie alternativ dazu die "-" bzw. "+"-Taste, während Sie die **Strg**-Taste gedrückt halten.

Pretty-Print-Anzeige
Mit dem Befehl **Pretty-Print** wird das aktive XML-Dokument in der **Textansicht** neu formatiert, um das Dokument strukturiert anzuzeigen. Standardmäßig wird jeder Child-Node wird um vier Leerzeichen vom Parent eingerückt. Diese Einstellung kann im Dialogfeld **Einstellungen für Textansicht** angepasst werden. Um ein XML-Dokument mit der Pretty-Print-Option anzuzeigen, wählen Sie den Menübefehl **Ausgabe | Pretty-Print** oder klicken Sie in der Symbolleiste auf (**Pretty-Print**).

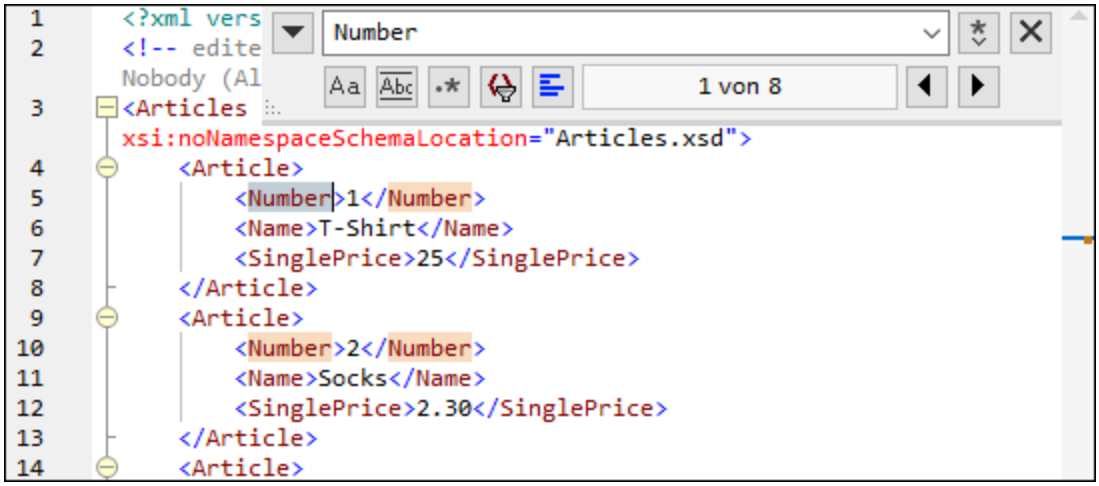
Zeilenumbruch
Mit Hilfe von Zeilenumbrüchen kann ein Codefragment innerhalb des Arbeitsbereichs angezeigt werden. Wenn Zeilenumbrüche nicht aktiviert sind, sind einige Textabschnitte im Arbeitsbereich eventuell nicht zu sehen. Um Zeilenumbrüche im gerade aktiven Dokument zu aktivieren bzw. zu deaktivieren, wählen Sie den Menübefehl **Ausgabe | Zeilenumbruch** oder klicken Sie in der Symbolleiste auf (**Zeilenumbruch**).

2.3.4 Suchen in der Textansicht

Der Text in den Fenstern **Ausgabe** und **XSLT** kann unter Verwendung einer Reihen von Optionen und visuellen Hilfsmitteln durchsucht werden.

Sie können im gesamten Dokument oder in einem ausgewählten Textbereich nach einem Begriff suchen. Drücken Sie **Strg+F** oder wählen Sie den Menübefehl **Bearbeiten | Suchen**, um eine Suche zu starten. Geben Sie den gewünschten String ein oder verwenden Sie die Auswahlliste, um einen String aus den letzten 10 Such-Strings auszuwählen. Nach Eingabe bzw. Auswahl des gewünschten String werden alle Übereinstimmungen markiert und die Position der Treffer wird durch orange Markierungen in der Bildlaufleiste gekennzeichnet (*siehe Abbildung unten*). Die Position der aktuell ausgewählten Übereinstimmung (grau markiert) hängt davon ab, wo sich der Cursor zuletzt befunden hat.

Sie sehen die Gesamtzahl der Treffer sowie die Indexposition des aktuell ausgewählten Treffers. Über die Schaltflächen (**Gehe zu vorherigem Ergebnis**) und (**Gehe zu nächstem Ergebnis**) gelangen Sie von einem Treffer zum nächsten.



Suchoptionen

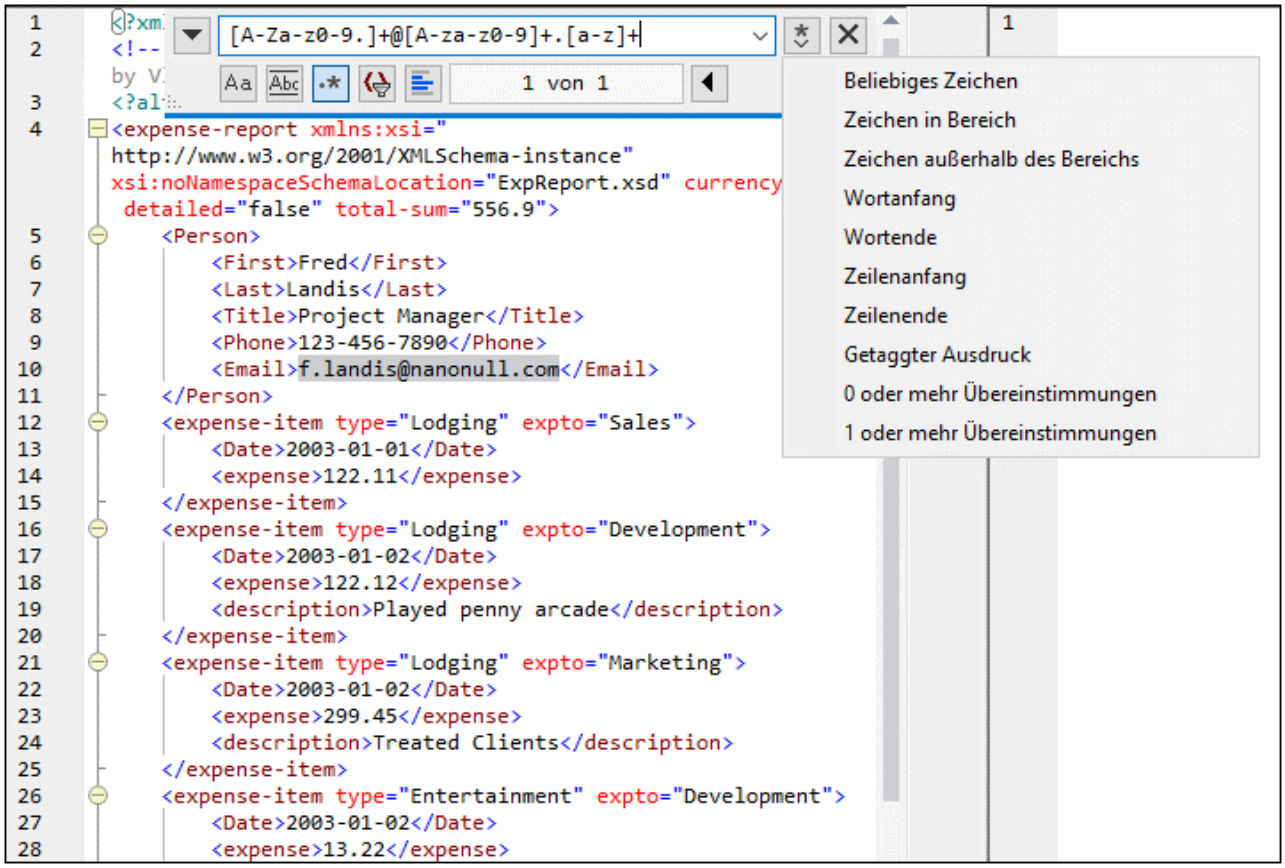
Über Schaltflächen unterhalb des Suchfelds können Sie Suchkriterien festlegen. In der folgenden Tabelle, finden Sie eine Liste der verfügbaren Optionen:

Option	Symbol	Beschreibung
GROSS/klein beachten	Aa	*Wenn die Schaltfläche aktiv ist, wird die Groß- und Kleinschreibung bei der Suche berücksichtigt (Address* ist nicht gleich *address*).
Ganzes Wort	Abc	Nur die exakte Wortentsprechung wird gefunden.
Regular Expression verwenden	.*	Wenn diese Option aktiv ist, wird der Suchbegriff als Regular Expression gelesen. Siehe *Regular Expressions* unten.
Anker suchen		Die Position des Ankers hängt von der Stelle ab, an der sich der Cursor zuletzt befunden hat. Die Position des Ankers ändert sich nicht durch Klicken auf **Gehe zu vorherigem Ergebnis**) und **Gehe zu nächstem Ergebnis**.
In Auswahl suchen		Bei einer Auswahl handelt es sich um einen markierten Textbereich. Um einen Begriff innerhalb einer Auswahl zu suchen, markieren Sie einen Textbereich, drücken Sie **Strg + F**, stellen Sie sicher, dass die Schaltfläche **In Auswahl suchen** aktiv ist und geben Sie den Suchbegriff in das Suchfeld ein.

Regular Expressions

Sie können zum Suchen eines Text-String Regular Expressions verwenden. Aktivieren Sie dazu zuerst die Option **Regular Expression** (*siehe Tabelle oben*). Geben Sie anschließend die Regular Expression in das Suchfeld ein. Bei Klick auf (**Regular Expression Builder**) erhalten Sie eine Liste von Beispielausdrücken für Regular Expressions (*siehe unten*). In der Abbildung unten sehen Sie eine Regular Expression zum Suchen von E-Mail-Adressen.

 © 2019-2025 Altova GmbH



Regular Expression-Metazeichen

Die unten stehende Tabelle enthält Metazeichen, die Sie zum Suchen und Ersetzen von Text verwenden können. Alle Metazeichen mit Ausnahme der beiden letzten entsprechen Menüeinträgen im **Regular Expression Builder** (*siehe oben*).

Menübefehl	Metazeichen	Beschreibung
Beliebiges Zeichen	.	Steht für jedes beliebige Zeichen. Dies ist ein Platzhalter für ein einzelnes Zeichen.
Zeichen im Bereich	[...]	Steht für jedes beliebige Zeichen in dieser Gruppe. **[abc]** z.B. steht für jedes der Zeichen a, b oder c. Sie können auch Bereiche angeben, z.B. **[a-z]** für alle klein geschriebenen Zeichen.
Zeichen nicht im Bereich	[^...]	Steht für jedes beliebige Zeichen in dieser Gruppe. **[^A-Za-z]** z.B. steht für jedes Zeichen mit Ausnahme alphabetischer Zeichen.
Wortanfang	\<	Steht für den Anfang eines Worts.
Wortende	\>	Steht für das Ende eines Worts.
Zeilenanfang	^	Steht für den Zeilenanfang, es sei denn dieses Zeichen wird innerhalb einer Menge verwendet (*siehe oben*).

Menübefehl	Metazeichen	Beschreibung
Zeilenende	$	Steht für das Zeilenende. Beispiel: **A+$** findet ein oder mehrere As am Ende der Zeile.
Getaggter Ausdruck	(abc)	Die Klammern markieren Beginn und Ende eines getaggten Ausdrucks. Getaggte (markierte) Ausdrücke eignen sich dazu, eine gesuchte Region zu markieren ("sich diese zu merken"), um diese später referenzieren zu können. Es können bis zu neun Unterausdrücke getaggt und später rückreferenziert werden. So wird etwa mit **(the) \1** der String the the gefunden. Diese Funktion bedeutet Folgendes: Suche den String the und merke ihn Dir als getaggte Region; der Ausdruck muss von einem Leerzeichen und einer Rückreferenz auf die zuvor gesuchte getaggte Region gefolgt sein.
0 oder mehr Übereinstimmungen	*	Steht für 0 oder mehr Übereinstimmungen mit dem vorhergehenden Ausdruck. Mit **Sa*m** werden z.B. Sm, Sam, Saam, Saaam usw. gefunden.
1 oder mehr Übereinstimmungen	+	Steht für 1 oder mehrere Instanzen des vorhergehenden Ausdrucks. Mit **Sa+m** werden z.B. Sam, Saam, Saaam usw. gefunden.
	\n	n steht für 1 bis 9 und bezieht sich auf die erste bis neunte getaggte Region (*siehe oben*).
	\x	Damit können Sie ein Zeichen x verwenden, das sonst eine spezielle Bedeutung hätte. So würde z.B. \[als [und nicht als der Beginn einer Zeichengruppe interpretiert werden.

Suchen von Sonderzeichen

Wenn die Option **Regular Expression verwenden** aktiv ist, können Sie im Text nach einem beliebigen der folgenden Sonderzeichen suchen.

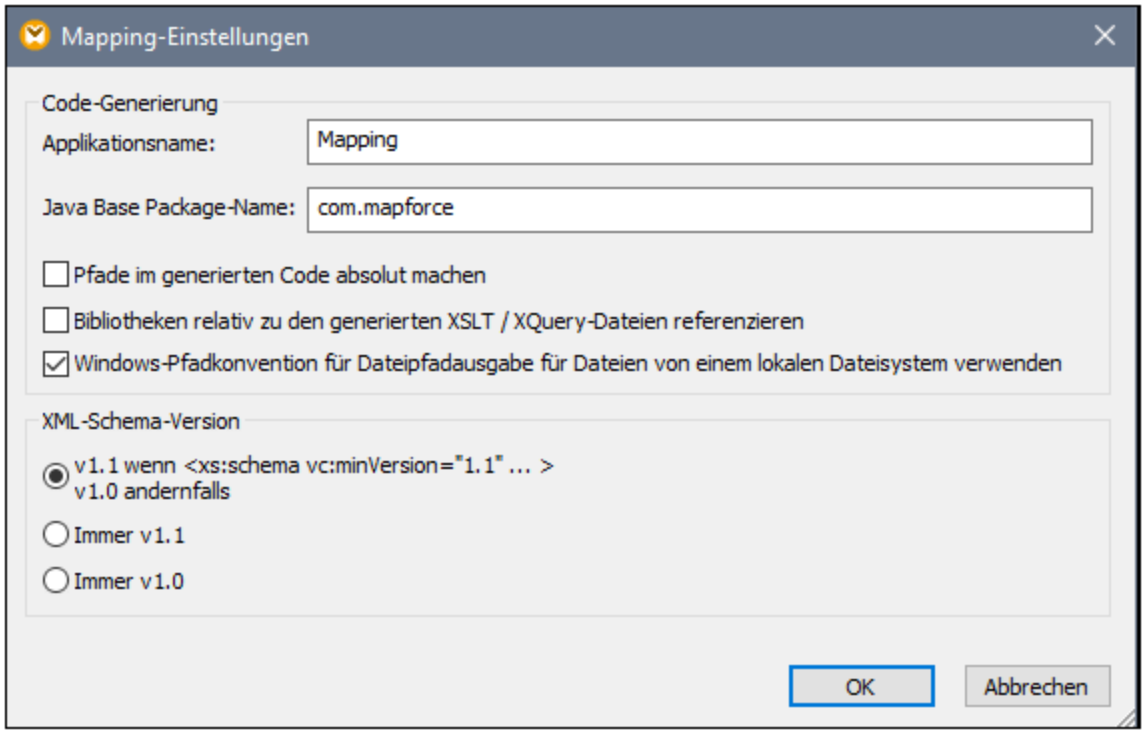
- \t (Tab)
- \r (Wagenrücklauf)
- \n (Neue Zeile)
- \\ (Umgekehrter Schrägstrich)

Um z.B. ein Tabulatorzeichen zu suchen, drücken Sie **Strg + F**, aktivieren Sie die Option **Regular Expression verwenden** und geben Sie in das **Suchfeld** **\t** ein.

2.3.5 Mapping-Einstellungen

Im Dialogfeld **Mapping-Einstellungen** (*siehe Abbildung unten*) können Sie dokumentspezifische Einstellungen definieren. Um dieses Dialogfeld zu öffnen, gehen Sie zum Menü **Datei** und klicken Sie auf **Mapping-Einstellungen**. Klicken Sie alternativ dazu mit der rechten Maustaste in einen leeren Bereich des Mappings und wählen Sie im Kontextmenü den Befehl **Mapping-Einstellungen**.

 © 2019-2025 Altova GmbH



Weiter unten finden Sie eine Beschreibung der verfügbaren Einstellungen.

Codegenerierung

- *Applikationsname:* Definiert das Präfix der generierten XSLT-Datei oder den Namen der generierten Java-, C#- oder C++-Applikation (*Professional und Enterprise Edition*).

- *Java Base Package-Name (Professional und Enterprise Edition)* Diese Option ist anwendbar, wenn als Transformationssprache Java ausgewählt ist. Sie definiert den Base Package-Namen für die Java-Ausgabe.

- Pfade im generierten Code absolut machen: Dieses Kontrollkästchen wirkt sich auf alle Pfade in Mapping-Komponenten mit Ausnahme von Pfaden zu externen Bibliotheksdateien (wie z.B. XSLT-Bibliotheken) aus. Über das Kontrollkästchen wird definiert, ob die Dateipfade im generierten Programmcode relativ oder absolut sein sollen. Nähere Informationen dazu finden Sie unter Pfade in Ausführungsumgebungen 43.

- *Bibliotheken relativ zu den generierten XSLT / XQuery-Dateien referenzieren:* Dieses Kontrollkästchen wird angewendet, wenn die Mapping-Sprache entweder XQuery (*Professional und Enterprise Edition*) oder XSLT ist. Normalerweise ist diese Option nützlich, wenn in Ihrem Mapping eine XSLT- oder XQuery-Bibliothek referenziert wird und Sie beabsichtigen, anhand des Mappings XSLT- oder XQuery-Dateien zu generieren. Aktivieren Sie dieses Kontrollkästchen, wenn die Bibliothekspfade relativ zum Verzeichnis des generierten XSLT- oder XQuery-Code gemacht werden sollen. Wenn das Kontrollkästchen deaktiviert ist, werden die Bibliothekspfade im generierten Code absolut. Siehe auch Bibliothekspfade im generierten Code 43.

- *Windows-Pfadkonvention für Dateipfadausgaben von einem lokalen Dateisystem verwenden:* Dieses Kontrollkästchen wird angewendet, wenn die Mapping-Sprache entweder XQuery

(*MapForce Professional und Enterprise Edition*), XSLT 2.0 oder XSLT 3.0 ist. Durch Aktivieren des Kontrollkästchens stellen Sie sicher, dass die Windows-Pfadkonventionen eingehalten werden. Bei der Ausgabe von XSLT 2.0-, XSLT 3.0- oder XQuery-Dokumenten wird die aktuell verarbeitete Datei intern über die **`document-uri`**-Funktion aufgerufen, die einen Pfad im Format `file://URI` für lokale Dateien zurückgibt. Wenn dieses Kontrollkästchen aktiviert ist, wird eine `file://URI`-Pfadspezifikation automatisch in einen vollständigen Windows-Dateipfad (z.B. `C:\...`) konvertiert, um die weitere Verarbeitung zu vereinfachen.

- Ausgabedateieinstellungen (Professional und Enterprise Edition)

  Über die Auswahlliste **Zeilenenden** können Sie die Zeilenenden der Ausgabedateien definieren. *Betriebssystemstandardeinstellung* ist die jeweilige Standardeinstellung für das Zielbetriebssystem, z.B. Windows (CR+LF), macOS (LF), oder Linux (LF). Sie können auch manuell ein bestimmtes Zeilenende auswählen. Die hier ausgewählten Einstellungen sind wichtig, wenn Sie ein Mapping zu einer MapForce Server-Ausführungsdatei (`.mfx`) kompilieren oder wenn Sie ein Mapping auf einem FlowForce Server bereitstellen, der auf einem anderen Betriebssystem installiert ist.

- XML-Schema-Version

  Diese Option dient zum Definieren der in der Mapping-Datei verwendeten XML-Schema-Version. Beachten Sie, dass nicht alle Version 1.1-spezifischen Funktionen derzeit unterstützt werden. Wenn die Deklaration `xs:schema vc:minVersion="1.1"` vorhanden ist, wird Version 1.1 verwendet; falls nicht, wird Version 1.0 verwendet.

  Wenn das XSD-Dokument kein `vc:minVersion` Attribut hat oder der Wert des Attributs `vc:minVersion` nicht `1.0` oder `1.1` ist, wird XSD 1.0 als Standardmodus verwendet. Verwechseln Sie das `vc:minVersion` Attribut nicht mit dem `xsd:version` Attribut. Das erste Attribut hat die XSD-Versionsnummer, während das zweite die Dokument-Versionsnummer enthält. Wenn diese Einstellung in einem vorhandenen Mapping geändert wird, werden alle Schemas der ausgewählten XML-Schema-Version neu geladen, was sich auch auf die Gültigkeit auswirken kann.

- Webservice Operation - Einstellungen (Enterprise Edition)

  Die Felder **WSDL-Definitionen**, **Service**, **Endpoint** und **Operation** werden automatisch ausgefüllt, wenn das Mapping-Dokument Teil einer Webservice-Implementierung ist.

© 2019-2025 Altova GmbH

2.4 Grundlegende Regeln und Strategien

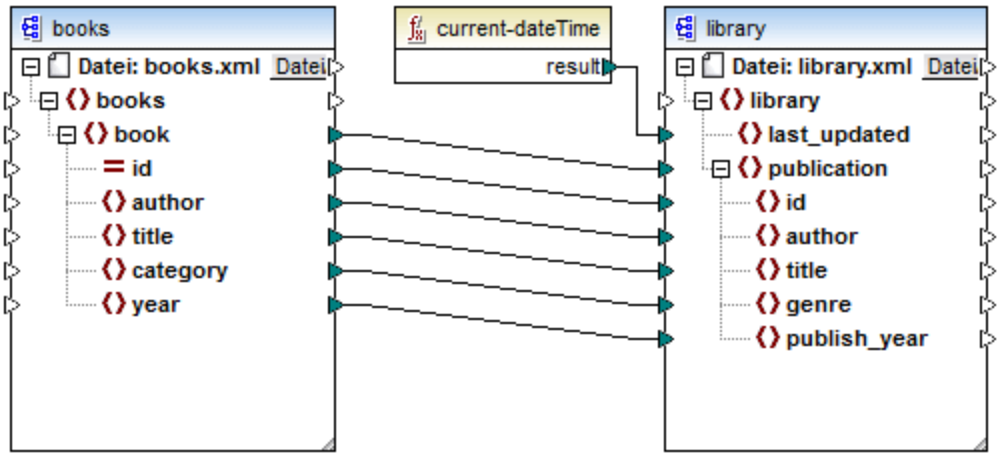
MapForce mappt Daten im Allgemeinen auf intuitive Art, doch gibt es einige Situationen, in denen die erzeugte Ausgabe zu viele oder zu wenige Datenelemente enthält. In diesem Kapitel werden die wichtigsten Mapping-Regeln beschrieben und Sie erfahren, wie Sie Situationen vermeiden, in denen infolge falscher Verbindungen oder des falschen Kontexts unerwünschte Ergebnisse erzeugt werden.

Damit ein Mapping gültig ist, muss es mindestens eine Quellkomponente enthalten, die mit mindestens einer Zielkomponente verbunden ist. Eine Quellkomponente ist eine Komponente, mit der Daten normalerweise aus einer Daten oder Datenbank ausgelesen werden. Eine Zielkomponente ist eine Komponente, mit der Daten normalerweise in eine Datei oder Datenbank geschrieben werden.

Alle erstellten Mapping-Verbindungen bilden gemeinsam einen Mapping-Algorithmus. Zur Laufzeit des Mappings wertet MapForce den Algorithmus aus und verarbeitet Daten anhand dessen. Ob und wie effizient der Mapping-Algorithmus funktioniert, hängt in erster Linie von den Verbindungen ab. Einige Einstellungen können außerdem auf Mapping- 74, Komponenten- 38 oder sogar Verbindungsebene 49 angepasst werden, doch bestimmen die *Verbindungen*, wie Ihre Daten verarbeitet werden.

Mapping-Regeln

In diesem Abschnitt werden die wichtigsten Datenverarbeitungsregeln beschrieben. Zur Veranschaulichung der Regeln haben wir das unten gezeigte Mapping erstellt.



Regel 1

Für jedes Datenelement in der Quellkomponente wird ein Datenelement in der Zielkomponente erstellt. **Dies ist die wichtigste Mapping-Regel in MapForce.** Wenn also die XML-Quelldatei wie im Mapping oben drei `book`-Elemente enthält,so werden in der Zielkomponente drei `publication`-Elemente erstellt. Beachten Sie, dass es auch einige Sonderfälle gibt, siehe Sequenzen 78.

Regel 2

MapForce beginnt immer beim *Root-Element der Zielkomponente* mit der Verarbeitung. Die Mapping-Ausführung beginnt immer hier. Nachdem der Ziel-Node verarbeitet wurde, arbeitet sich MapForce von oben nach unten durch die Hierarchie durch.

Regel 3
Für jede Verbindung wird ein *aktueller Mapping-Kontext* erstellt. Der Kontext legt fest, welche Daten *für den aktuellen Ziel-Node und seine Nachfahren* zur Verfügung stehen, d.h. mit dem Kontext wird festgelegt, welche Quelldatenelemente tatsächlich aus der Quell- in die Zielkomponente kopiert werden. Indem Sie eine Verbindung ziehen oder nicht erstellen, ändern Sie eventuell unabsichtlich den aktuellen Kontext, was sich auf die Ausgabe des Mappings auswirkt. So könnte z.B. eine Datenbank oder ein Webservice unnötigerweise mehrmals während derselben Mapping-Ausführung aufgerufen werden. Eine genauere Beschreibung dieses Prinzips finden Sie im Kapitel Kontext und Verarbeitungsreihenfolge 79.

Regel 4
Wenn Sie eine Verbindung von einem Quelldatenelement zu einen Zieldatenelement ziehen, liest das Mapping die Daten aus diesem Quelldatenelement aus, übergibt sie an das Zieldatenelement und generiert Instanzdaten im Zielformat.

Das Quelldatenelement kann null, einmal oder mehrmals in der Quellinstanzdatei vorkommen. Unabhängig davon, wie oft es vorkommt, werden in MapForce in Strukturkomponenten (z.B. XML) immer nur Schemahierarchien angezeigt. Aus diesem Grund hat die Quellkomponente im obigen Mapping nur ein `book`-Element.

2.4.1 Sequenzen

Das Verständnis von Sequenzen ist für die Beherrschung der Mapping-Techniken in MapForce unerlässlich. Die wichtigste Regel für Sequenzen lautet folgendermaßen:

> *Wenn eine Sequenz bestehend aus mehr als einem Node mit einem Ziel-Node verbunden wird, wird eine Schleife generiert, die so viele Ziel-Nodes generiert, wie Quell-Nodes vorhanden sind.*

Wenn ein Filter zwischengeschaltet ist, überprüft MapForce, ob in der Sequenz mindestens eine Boolesche Bedingung vorhanden ist, die true ergibt und mappt die Daten, die diese Bedingung erfüllen, auf die Zielkomponente.

Wenn eine Sequenz leer ist, wird auf Seite der Zielkomponente nichts generiert. Wenn die Zielkomponente z.B. ein XML-Dokument ist und die Quellsequenz leer ist, werden in der Zielkomponente gar keine XML-Elemente erstellt.

Ausnahmeereignisse

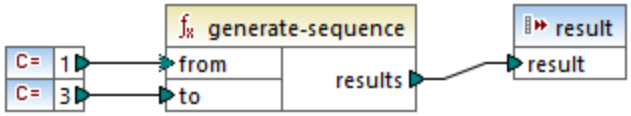
Beachten Sie, dass es für die Regel zu Sequenzen einige Ausnahmen gibt (*siehe unten*).

Ausnahme 1
Wenn es sich beim Zieldatenelement um ein XML-Root-Element handelt, wird dieses nur genau einmal erstellt. Wenn Sie damit eine Sequenz verbinden, ist das Ergebnis dem Schema gemäß möglicherweise nicht gültig. Wenn auch Attribute des Root-Elements verbunden werden, schlägt die XML-Serialisierung zur Laufzeit fehl. Vermeiden Sie es daher, eine Sequenz mit dem XML-Root-Element zu verbinden.

Ausnahme 2
Wenn im Zieldatenelement nur ein einziger Wert zulässig ist, wird es nur einmal erstellt. Beispiele für Datenelemente, in denen nur ein einziger Wert zulässig ist: XML-Attribute, Datenbankfelder, einfache Output-Komponenten. So wird etwa mit dem unten gezeigten Mapping mit Hilfe der **`generate-sequence-`**Funktion eine Sequenz von drei Ganzzahlen (1, 2, 3) generiert. Trotzdem enthält die Ausgabe nur eine Ganzzahl, da es

 © 2019-2025 Altova GmbH

sich bei der Zielkomponente um eine einfache Output-Komponente handelt, in der nur ein einziger Wert zulässig ist. Die anderen beiden Werte werden ignoriert.



Ausnahme 3
Wenn im Quellschema festgelegt ist, dass ein bestimmtes Datenelement nur einmal vorkommen darf, die Instanzdatei aber viele davon enthält, extrahiert MapForce unter Umständen das erste Datenelement aus der Quelldatei (das laut Schema vorhanden sein muss) und erstellt in der Zielkomponente nur ein Datenelement. Um dieses Verhalten zu verhindern, deaktivieren Sie in den Komponenteneinstellungen das Kontrollkästchen *Input-Verarbeitungsoptimierungen auf Basis von min/maxOccurs aktivieren,* siehe auch XML-Komponenteneinstellungen 131.

Funktionen

Wenn eine Nicht-Sequenz-Funktion als Input eine Sequenz erhält, wird die Funktion so oft aufgerufen (und erzeugt so viele Ergebnisse) wie Datenelemente in der Sequenz vorhanden sind. Wenn eine Funktion als Input eine leere Sequenz erhält, gibt sie ein leeres Ergebnis zurück und erzeugt somit gar keine Ausgabe.

Funktionen, die einen Wert zurückgeben, selbst wenn sie eine leere Sequenz erhalten
Es gibt jedoch einige Funktionskategorien, die aufgrund ihres Designs auch dann einen Wert zurückgeben, wenn Sie als Input eine leere Sequenz erhalten:

- **`exists`**, **`not-exists`**, **`substitute-missing`**
- **`is-not-null`**, **`is-null`**, **`substitute-null`** (Diese drei Funktionen sind Aliasnamen für die erstgenannten drei Funktionen)
- aggregate-Funktionen (**`sum`**, **`count`**, usw.)
- benutzerdefinierte Funktionen, die Sequenzen unterstützen und regulär sind (also nicht-inline-Funktionen)

Wenn Sie einen fehlenden Wert ersetzen müssen, fügen Sie die Funktion **`substitute-missing`** zum Mapping hinzu und ersetzen sie den fehlenden Wert durch den Ersetzungswert Ihrer Wahl.

Funktionen mit mehreren Inputs
Funktionen können mehrere Inputs haben. Wenn eine Sequenz mit jedem einzelnen Input verbunden wird, wird dadurch ein kartesisches Produkt aller Inputs erzeugt (eine Gruppe aller geordneten Paare, die durch Kombination von Elementen aus zwei oder mehr Gruppen erstellt werden kann), was normalerweise nicht das gewünschte Ergebnis ist. Um dies zu vermeiden, verbinden Sie nur eine Sequenz mit einer Funktion mit mehreren Parametern; alle anderen Parameter müssen von übergeordneten Elementen oder anderen Komponenten aus mit Einzeldatenelementen verbunden werden.

2.4.2 Kontext und Verarbeitungsreihenfolge

Der Kontext ist einer der wichtigsten Aspekte bei der Mapping-Ausführung. Den Kontext zu verstehen, ist der Schlüssel zur Erzielung des gewünschten Ergebnisses in der Ausgabe. Der Kontext legt fest, welche Daten *für*

den aktuellen Ziel-Node und seine Nachfahren zur Verfügung stehen und welche Quelldatenelemente tatsächlich ausgewählt und aus der Quell- in die Zielkomponente kopiert werden.

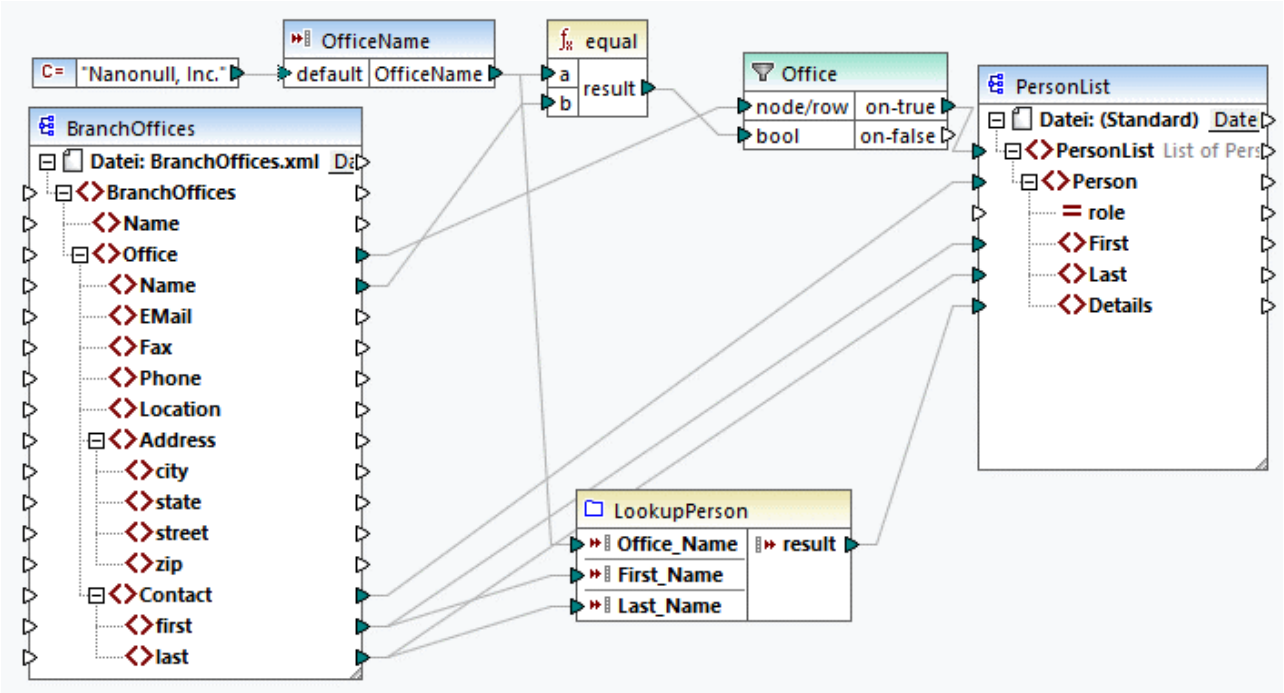
MapForce ermittelt den aktuellen Kontext immer vom *Root-Zieldatenelement (Node)* aus. Die Mapping-Ausführung beginnt immer hier. Nachdem der Ziel-Node verarbeitet wurde, arbeitet sich MapForce von oben nach unten durch die Hierarchie durch. Für jeden Ziel-Node werden die Verbindungen durch alle etwaigen zwischengeschalteten Funktionen hindurch zu den Quell-Nodes zurückverfolgt.

Für jeden Ziel-Node wird ein neuer Kontext hergestellt, der anfangs alle aktuellen Quell-Nodes des übergeordneten Datenelements und alle Funktionsergebnisse enthält. Während MapForce die direkt oder indirekt zum Ziel-Node führenden Verbindungen auswertet, werden zusätzliche Quell-Nodes und Funktionsergebnisse zum Kontext hinzugefügt. Gleichrangige Ziel-Nodes sind daher voneinander unabhängig, haben aber Zugriff auf alle Quell-Nodes ihrer übergeordneten Datenelemente.

Eine Zusammenfassung der grundlegenden Regeln der Mapping-Ausführung finden Sie unter Grundlegende Regeln und Strategien 77.

Beispiel

Um die oben beschriebenen Regeln in die Praxis umzusetzen, wird hier das folgende Mapping behandelt: **`MapForceExamples\PersonListByBranchOffice.mfd`** (*Abbildung unten*). Hauptaufgabe des Mappings ist es, eine Liste der Angestellten einer bestimmten Niederlassung mit ihren persönlichen Daten zu erhalten.



Quell- und Zielstruktur

Das Mapping in unserem Beispiel enthält eine XML-Quellkomponente und eine XML-Zielkomponente. Zusätzlich dazu gibt es eine einfache Input-Komponente, die ebenfalls als Quellkomponente dient und den Namen der gewünschten Niederlassung, deren Angestellten wir benötigen, bereitstellt. Außerdem enthält das Mapping verschiedene Transformationskomponenten wie Filter und einige Funktionen.

 © 2019-2025 Altova GmbH

Beachten Sie, dass in MapForce in Strukturkomponenten (z.B. XML) immer nur Schemahierarchien angezeigt werden. So wird etwa in der Komponente `BranchOffices` nur ein `Contact`-Element angezeigt, obwohl die Instanz-Quelldatei **`BranchOffices.xml`** mehrere `Contact`-Nodes mit unterschiedlichem Inhalt und in verschiedenen `Office` Parent-Nodes enthält. Unten sehen Sie einen Auszug aus der Datei **`BranchOffices.xml`**.

```
<results xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="Scores.xsd">
      <Name>Nanonull</Name>
      <Office>
            <Name>Nanonull, Inc.</Name>
            <EMail>office@nanonull.com</EMail>
            <Fax>+1 (321) 555 5155 - 9</Fax>
            <Phone>+1 (321) 555 5155</Phone>
            <Address>
                  <city>Vereno</city>
                  <state>CA</state>
                  <street>119 Oakstreet, Suite 4876</street>
                  <zip>29213</zip>
            </Address>
            <Contact>
                  <first>Vernon</first>
                  <last>Callaby</last>
            </Contact>
            <Contact>...</Contact>
            <Contact>...</Contact>
            <Contact>...</Contact>
            <...>
      </Office>
      <Office>
            <Name>Nanonull Partners, Inc.</Name>
            <EMail>nextoffice@nanonull.com</EMail>
            <Fax>+1 (927) 555 1845</Fax>
            <Phone>+1 (927) 555 0094</Phone>
            <Address>
                  <city>Brenton</city>
                  <state>MA</state>
                  <street>9865 Millenium Center, Suite 456</street>
                  <zip>5985</zip>
            </Address>
            <Contact>
                  <first>Steve</first>
                  <last>Meier</last>
            </Contact>
            <Contact>...</Contact>
            <Contact>...</Contact>
            <...>
      </Office>
</BranchOffices>
```

Kontext und Verarbeitungsreihenfolge

Die Verarbeitung beginnt in unserem Mapping beim Ziel-Root-Node `PersonList`. Wir können die Verbindung über den Filter und die `equal`-Funktion zurück zum Quelldatenelement `Office` verfolgen.

Eine einzelne Niederlassung (office) als Kontext für die Zielkomponente

Da unser Mapping einen Filter hat, werden nur die Daten, die die Boolesche Bedingung erfüllen, auf die Zielkomponente gemappt. Diese Bedingung trifft nur einmal zu, da die XML-Quelldatei nur eine Niederlassung (office) namens Nanonull, Inc. enthält. Folglich definiert die Verbindung von der `PersonList`-Liste durch den Filter zum Node `Office` eine einzige Niederlassung (office) als Kontext für das gesamte Zieldokument. D.h. alle weiteren Verbindungen zu den Nachfahren des Root-Elements sind vom Filter betroffen und haben nur Zugriff auf die Daten der Niederlassung Nanonull, Inc.. Im aktuellen Kontext gibt es keine weitere Niederlassung.

Ein Person-Element pro Contact-Element

Die nächste Verbindung ist die zwischen den Nodes `Contact` und `Person`. Gemäß der allgemeinen Mapping-Regel 77 wird mit dieser Verbindung für jedes `Contact`.-Quelldatenelement ein `Person`-Zieldatenelement erstellt. Bei jeder Iteration wird durch die Verbindung ein einziges aktuelles `Contact`-Element für die Verbindungen zu den Children des Elements `Person` ermittelt. Daher werden für die Child-Verbindungen (`first` mit `First`, `last` mit `Last`) der Vor- (first) und Nachname (last) des *aktuellen* `Contact`-Elements ausgewählt und in die Zielkomponente geschrieben.

Benutzerdefinierte Funktion

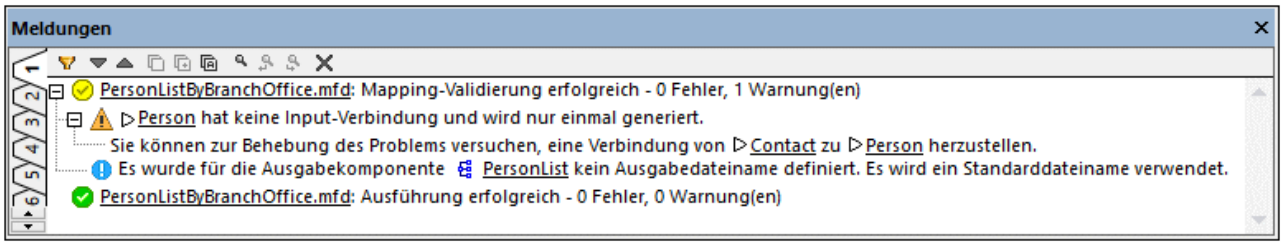
Das Weiteren enthält das Mapping eine benutzerdefinierte Funktion 219 namens `LookupPerson`. Aufgrund der übergeordneten Verbindung zwischen `Contact` und `Person` wird die benutzerdefinierte Funktion ebenfalls im Kontext der einzelnen `Person`-Datenelemente ausgeführt. So wird die Funktion jedes Mal, wenn in der Zielkomponente ein neues `Person`-Datenelement erstellt wird, aufgerufen, um das Element `Details` der jeweiligen Person zu befüllen.

Die Funktion hat drei Input-Parameter. Der Parameter `OfficeName` liest Daten aus der einfachen Input-Komponente aus. Die Quelldaten für diesen Parameter könnten genauso gut durch das Quelldatenelement `Name` bereitgestellt werden, ohne dass sich die Mapping-Ausgabe auf irgendeine Art ändern würde. Der Ausgangswert ist in beiden Fällen derselbe und stammt aus dem Parent-Kontext. Intern verkettet die Look-up-Funktion die als Argumente erhaltenen Werte und erzeugt einen einzigen Wert.

Nähere Informationen darüber, wie die Funktion **`LookupPerson`** konfiguriert wird, finden Sie unter Look-up-Implementierung 235.

Wenn die übergeordneten Elemente nicht verbunden werden

Gäbe es keine Verbindung zwischen `Contact` und `Person`, würde im Mapping nur ein `Person`-Datenelement mit mehreren `First-`, `Last-` und `Details`-Nodes erstellt. In solchen Fällen wird in MapForce im Fenster "Meldungen" eine Warnung angezeigt (*Abbildung unten*) und es wird vorgeschlagen, die Parent-Nodes zu verbinden, um das Problem zu beheben.

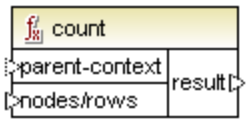


2.4.3 Parent-Kontext

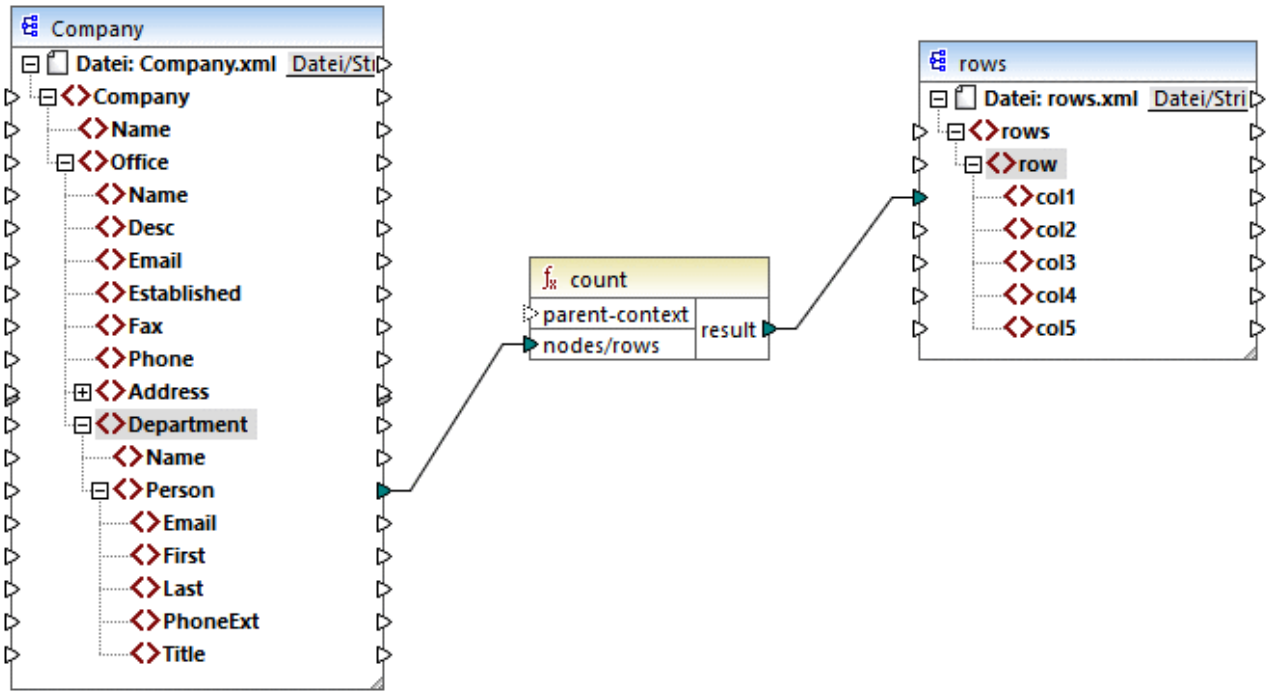
Einige Mapping-Komponenten haben ein optionales **parent-context**-Datenelement.

Das Argument `parent-context` ist ein optionales Argument in einigen MapForce Aggregatfunktionen der core-Bibliothek wie z.B. `min`, `max`, `avg`, `count`. Der parent-context bestimmt in einer Quellkomponente mit mehreren hierarchischen Sequenzen, an welcher Node-Gruppe die Funktion ausgeführt werden soll.

Mit Hilfe dieses Datenelements können Sie den Mapping-Kontext, in dem diese Komponente verwendet werden soll, beeinflussen und somit die Mapping-Ausgabe ändern. Die Komponenten, die einen optionalen **Parent-Kontext** haben, sind: Aggregatfunktionen, Variablen und Join-Komponenten.



Um ein Beispiel dafür zu sehen, wie Sie mit der Änderung des Parent-Kontexts arbeiten können, öffnen Sie das folgende Mapping: **<Dokumente>\Altova\MapForce2025\MapForceExamples\ParentContext.mfd**.



Die XML-Quelldatei im Mapping oben enthält einen einzigen **Company**-Node, der zwei **Office**-Nodes enthält. Jeder **Office**-Node enthält mehrere **Department**-Nodes und jeder **Department**-Node enthält mehrere **Person**-Nodes. Wenn Sie die XML-Datei in einem XML-Editor öffnen, sehen Sie folgende Aufteilung von Personen nach Büro (Office) und Abteilung (Department):

Office	Department	Personenzahl
Nanonull, Inc.	Administration	3

Office	Department	Personenzahl
	Marketing	2
	Engineering	6
	IT & Technical Support	4
Nanonull Partners, Inc.	Administration	2
	Marketing	1
	IT & Technical Support	3

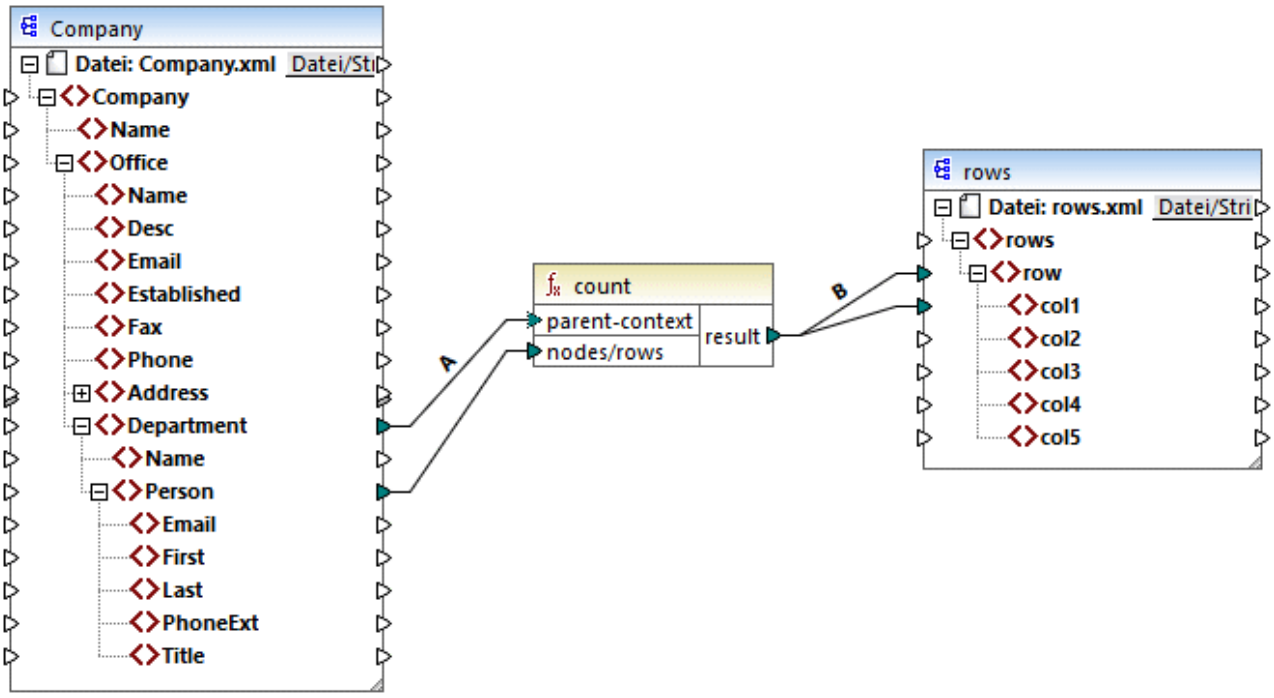
Im Mapping werden alle Personen in allen Abteilungen gezählt. Dazu wird die `count`-Funktion aus der `core`-Bibliothek verwendet. Wenn Sie auf das Fenster **Ausgabe** klicken, um eine Vorschau auf das Mapping anzuzeigen, sehen Sie, dass als Ergebnis ein einziger Wert, nämlich **21**, erzeugt wird, was der Gesamtanzahl aller Personen in der XML-Quelldatei entspricht.

Das Mapping funktioniert folgendermaßen:

- Wie gewöhnlich beginnt die Mapping-Ausführung beim obersten Node der Zielkomponente (in diesem Beispiel **rows**). **rows** hat keine eingehende Verbindung. Infolgedessen wird zwischen **Company** (oberstes Datenelement der Quellkomponente) und **rows** (oberstes Datenelement der Zielkomponente) ein impliziter Mapping-Kontext erstellt.
- Das Resultat der Funktion ist ein einziger Wert, da die Quelldatei nur eine Firma (Company) enthält.
- Um das Zieldatenelement **col1** zu befüllen, führt MapForce die `count`-Funktion im oben erwähnten *impliziten Parent-Kontext* aus, d.h. es werden alle **Person**-Nodes aus allen Büros und allen Abteilungen gezählt.

Mit Hilfe des Arguments **parent-context** der Funktion können Sie den Mapping-Kontext ändern. Dadurch kann z.B. die Anzahl der Personen in jeder einzelnen Abteilung ermittelt werden. Ziehen Sie dazu, wie unten gezeigt, zwei weitere Verbindungen:

 © 2019-2025 Altova GmbH

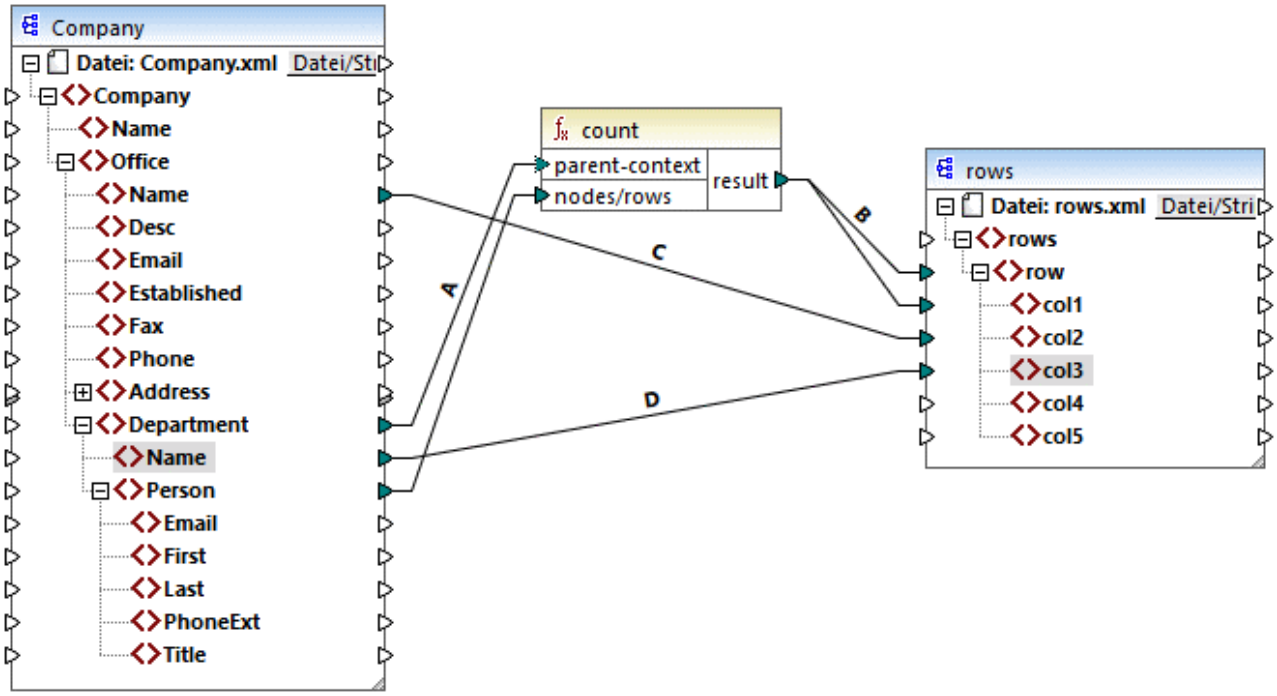


Im oben gezeigten Mapping wird durch die Verbindung A der Parent-Kontext der `count`-Funktion in **Department** geändert, wodurch die Funktion die Anzahl der Personen in jeder einzelnen Abteilung zählt. Beachten Sie, dass die Funktion nun eine *Sequenz* von Ergebnissen anstatt eines einzigen Ergebnisses zurückgibt, da in der Quelldatei mehrere Abteilungen (Department) vorhanden sind. Aus diesem Grund wurde die Verbindung B erstellt: Für jedes Datenelement in der erzeugten Sequenz wird in der Zieldatei eine neue Zeile (row) erstellt. Die Mapping-Ausgabe hat sich nun entsprechend geändert (Beachten Sie, dass die Zahlen genau der Anzahl der Personen in den einzelnen Abteilungen entspricht):

```
<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>
```

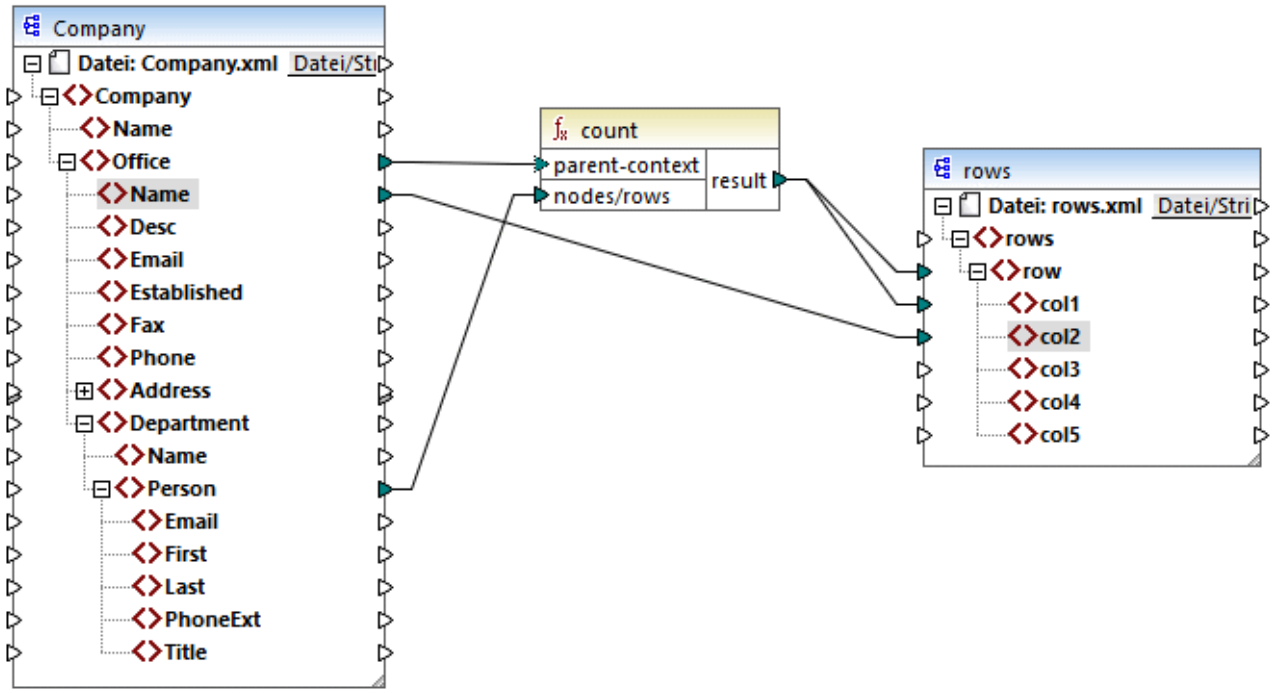
Da im aktuellen Mapping für jede Abteilung (Department) eine Zeile (row) erstellt wird, können Sie durch Ziehen der Verbindungen C und D den Office-Namen und den Abteilungsnamen ebenfalls in die Zieldatei kopieren:



Dadurch wird in der Ausgabe nicht nur die Anzahl der Personen, sondern auch der entsprechende Büro- und Abteilungsname angezeigt.

Wenn Sie die Anzahl der Personen in den einzelnen Büros (Office) zählen möchten, verbinden Sie den Parent-Kontext der **count**-Funktion mit dem Datenelement **Office** in der Quellkomponente.

 © 2019-2025 Altova GmbH



Mit den oben gezeigten Verbindungen gibt die **count**-Funktion ein Ergebnis für jedes Büro zurück. Die Quelldatei enthält zwei Büros, daher gibt die Funktion nun zwei Sequenzen zurück. Infolgedessen enthält die Ausgabe zwei Zeilen (row), von denen jede die Anzahl der Personen in diesem Büro enthält:

```
<rows>
    <row>
        <col1>15</col1>
        <col2>Nanonull, Inc.</col2>
    </row>
    <row>
        <col1>6</col1>
        <col2>Nanonull Partners, Inc.</col2>
    </row>
</rows>
```

2.4.4 Prioritätskontext

Der Prioritätskontext ist eine Methode, mit der Sie festlegen können, in welcher Reihenfolge die Input-Parameter einer Funktion ausgewertet werden. Unter Umständen muss ein Prioritätskontext definiert werden, wenn in Ihrem Mapping Daten aus zwei nicht miteinander in Zusammenhang stehenden Quelldateien verbunden werden müssen.

Um zu verstehen, wie der Prioritätskontext funktioniert, denken Sie daran, dass bei Ausführung eines Mappings die Verbindung zu einem Input-Datenelement eine *Sequenz* aus mehreren Werten übertragen kann. Bei Funktionen mit zwei Input-Parametern bedeutet dies, dass zwei Schleifen erstellt werden müssen, von denen

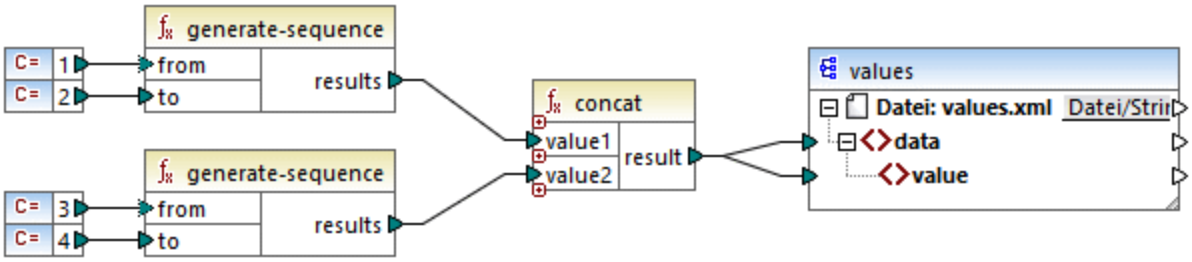
eine zuerst verarbeitet werden muss. Die zuerst verarbeitete Schleife ist die äußere Schleife. So erhält z.B. die **equal**-Funktion zwei Parameter: *a* und *b*. Wenn sowohl *a* als auch *b* eine Wertesequenz erhält, verarbeitet MapForce diese folgendermaßen:

- Für jede Instanz von *a*
  - Für jede Instanz von *b*
    - Ist *a* gleich *b*?

Wie Sie oben sehen, wird jedes *b* im Kontext eines jeden *a* ausgewertet. Mit Hilfe des Prioritätskontexts können Sie die Verarbeitungslogik ändern, sodass jedes *a* im Kontext eines jeden *b* ausgewertet wird, d.h. Sie können die innere Schleife mit der äußeren vertauschen, z.B:

- Für jede Instanz von *b*
  - Für jede Instanz von *a*
    - Ist *a* gleich *b*?

Betrachten wir nun ein Mapping, in dem sich der Prioritätskontext auf die Mapping-Ausgabe auswirkt. Die **concat**-Funktion im unten gezeigten Mapping hat zwei Input-Parameter. Jeder Input-Parameter ist eine Sequenz, die mit Hilfe der **generate-sequence**-Funktion generiert wurde. Die erste Sequenz ist "1,2"und die zweite Sequenz ist "3,4".



Führen wir das Mapping zuerst aus, ohne einen Prioritätskontext zu definieren. Die **concat-**Funktion beginnt zuerst mit der Auswertung der obersten Sequenz, daher werden die Werte in der folgenden Reihenfolge miteinander kombiniert:

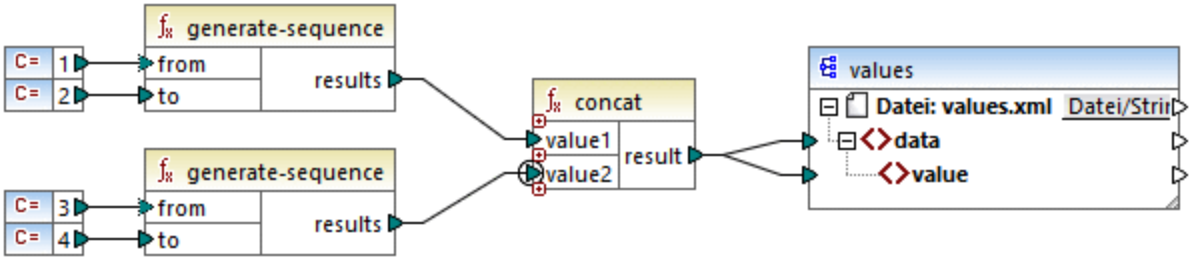
- 1 mit 3
- 1 mit 4
- 2 mit 3
- 2 mit 4

Dies wirkt sich auf die Mapping-Ausgabe folgendermaßen aus:

```
<data>
    <value>13</value>
    <value>14</value>
    <value>23</value>
    <value>24</value>
</data>
```

 © 2019-2025 Altova GmbH

Wenn Sie mit der rechten Maustaste auf den zweiten Input-Parameter klicken und im Kontextmenü den Eintrag **Prioritätskontext** auswählen, wird dieser zum Prioritätskontext. Wie Sie in der Abbildung unten sehen, ist der Prioritätskontext-Input mit einem Kreis umrandet.



Dieses Mal wird der zweite Input-Parameter zuerst ausgewertet. Die concat-Funktion verkettet nach wie vor dieselben Werte, diesmal wird jedoch die Sequenz '3,4' zuerst ausgewertet. Die Ausgabe wird folglich zu:

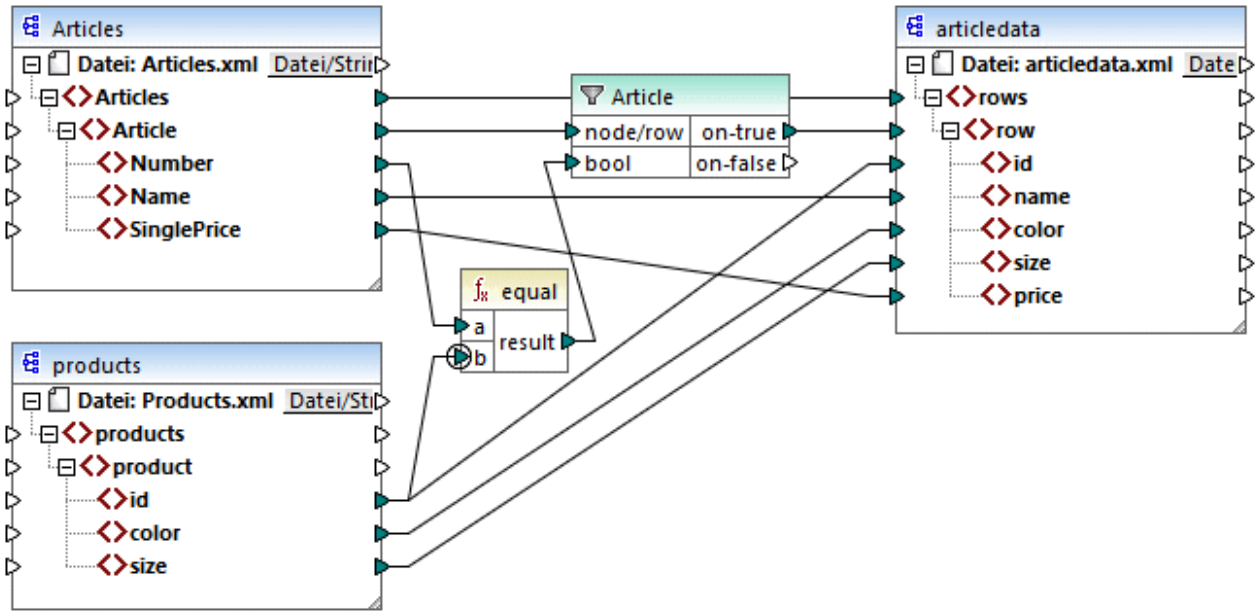
```
<data>
    <value>13</value>
    <value>23</value>
    <value>14</value>
    <value>24</value>
</data>
```

Bisher wurde nur der theoretische Hintergrund des Prioritätskontexts beschrieben. Ein praktisches Anwendungsszenario finden Sie im Beispiel: Filtern mit Prioritätskontext 89.

2.4.4.1 Beispiel: Filtern mit Prioritätskontext

Wenn eine Funktion mit einem Filter verbunden wird, wirkt sich der Prioritätskontext nicht nur auf die Funktion selbst, sondern auch auf die Auswertung des Filters aus. Im Mapping unten sehen Sie ein Beispiel für einen typischen Fall, in dem ein Prioritätskontext definiert werden muss, um das korrekte Ergebnis zu erhalten. Sie finden dieses Mapping unter dem folgenden Pfad:
<Dokumente>\Altova\MapForce2025\MapForceExamples\FilterWithPriority.mfd.

Anmerkung: In diesem Mapping werden XML-Komponenten verwendet, doch gilt dieselbe unten beschriebene Logik auch für allen anderen in MapForce verwendeten Komponententypen, darunter auch für EDI, JSON usw.



Ziel des oben gezeigten Mappings ist es, Daten aus **Articles.xml** in eine neue XML-Datei mit einem anderen Schema, nämlich **articledata.xml,** zu kopieren. Gleichzeitig sollen die Informationen zu jedem Artikel (Article) in der Datei **Products.xml** abgerufen und mit dem entsprechenden Artikeldatensatz verbunden werden. Beachten Sie, dass jeder Datensatz in **Articles.xml** eine Nummer (**Number)** und jeder Datensatz in **Products.xml** eine **id** hat. Wenn diese beiden Werte identisch sind, sollen alle andere Werte (**Name**, **SinglePrice**, **color**, **size**) in dieselbe Zeile (**row)** in der Zielkomponente kopiert werden.

Dieses Ziel wurde durch Hinzufügen eines Filters erreicht. Für jeden Filter wird als Input eine Boolesche Bedingung benötigt. Nur die Nodes/Zeilen, die die Bedingung erfüllen, werden in die Zielkomponente kopiert. Zu diesem Zweck gibt es im Mapping eine `equal`-Funktion. Mit der `equal`-Funktion wird überprüft, ob die Artikelnummer und die Produkt-ID in beiden Quelldateien identisch sind. Das Ergebnis wird anschließend als Input für den Filter bereitgestellt. Bei **true** wird das **Article**-Datenelement in die Zielkomponente kopiert.

Beachten Sie, dass für den zweiten Input-Parameter der zweiten `equal`-Funktion ein Prioritätskontext definiert wurde. Der Prioritätskontext macht in diesem Mapping einen großen Unterschied. Wenn er nicht definiert wird, führt dies zu einer falschen Mapping-Ausgabe.

Anfängliches Mapping: Kein Prioritätskontext

Hier die Mapping-Logik ohne Prioritätskontext:

- Gemäß der allgemeinen Mapping-Regel wird für jedes **Article**-Element, das die Filter-Bedingung erfüllt, in der Zielkomponente eine neuen Zeile (**row**) erstellt. Dafür sorgt die Verbindung zwischen **Article** und **row** (über die Funktion und den Filter).
- Mit dem Filter wird für jeden Artikel die Bedingung überprüft. Dazu iteriert der Filter durch alle Produkte und bringt mehrere Produkte in den aktuellen Kontext.
- Um in der Zielkomponente das Element **id** zu befüllen, geht MapForce nach der allgemeinen Regel vor (Erstelle für jedes Datenelement in der Quelldatei ein Datenelement in der Zielkomponente). Wie jedoch oben erläutert, befinden sich alle Produkte aus **Products.xml** im aktuellen Kontext. Es gibt keine Verbindung zwischen **product** und irgendeinem anderen Node in der Zielkomponente, damit nur

 © 2019-2025 Altova GmbH

die **id** eines bestimmten Produkts ausgelesen wird. Folglich werden für jeden Artikel (**Article**) mehrere **id**-Elemente in der Zielkomponente erstellt. Dasselbe geschieht mit **color** und **size**.

Zusammenfassung: Die Datenelemente aus **Products.xml** haben den Kontext des Filters (der durch jedes Produkt iterieren muss). Daher werden die **id-**, **color-** und **size**-Werte so oft in die einzelnen **row**-Elemente in der Zielkomponente kopiert, wie sich in der Quelldatei Produkte befinden und es wird eine falsche Ausgabe wie die unten gezeigte generiert:

```
<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>
```

Lösung A: Verwendung eines Prioritätskontexts

Das obige Problem wurde durch Hinzufügen eines Prioritätskontexts zur Funktion, die die Boolesche Bedingung des Filters berechnet, gelöst.

Wenn in diesem Fall der zweite Input-Parameter der `equal`-Funktion als Prioritätskontext definiert wird, erhält die Sequenz, die aus **Products.xml** eingeht, Priorität. Dies wird in die folgende Mapping-Logik übersetzt:

- Befülle für jedes Produkt den Input **b** der `equal`-Funktion (anders ausgedrückt: gib **b** Priorität). In dieser Phase befinden sich die Informationen des aktuellen Produkts im Kontext.
- Befülle für jeden Artikel den Input **a** der `equal`-Funktion und überprüfe, ob die Filterbedingung "true" ist. Falls ja, setze auch die Artikelinformationen in den aktuellen Kontext.
- Kopiere als nächstes die Artikel- und Produktinformationen aus dem aktuellen Kontext in die entsprechenden Datenelemente in der Zielkomponente.

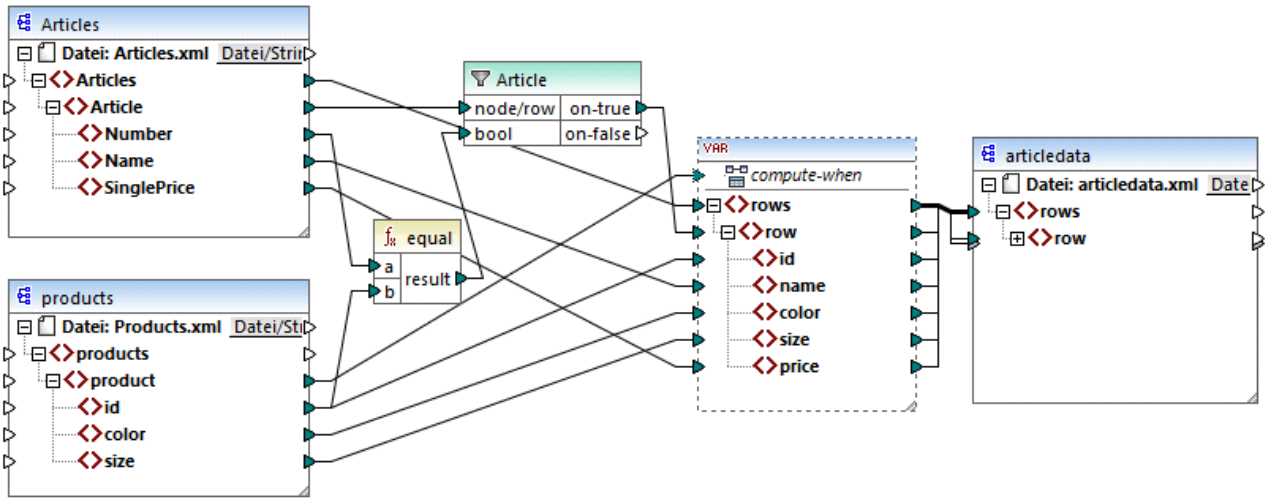
Mit der obigen Mapping-Logik wird eine korrekte Ausgabe erzeugt, z.B:

```
<rows>
    <row>
        <id>1</id>
        <name>T-Shirt</name>
        <color>red</color>
        <size>10</size>
        <price>25</price>
    </row>
</rows>
```

Lösung B: Verwendung einer Variablen

Als Alternativlösung dazu können Sie auch mit Hilfe einer Zwischenvariablen die einzelnen Artikel und Produkte, die die Filterbedingung erfüllen, in denselben Kontext bringen. Variablen eignen sich für derartige Szenarien, da Sie damit Daten temporär im Mapping speichern und dadurch bei Bedarf den Kontext ändern können.

In Szenarien wie diesem können Sie eine Variable zum Mapping hinzufügen, die dasselbe Schema wie die Zielkomponente hat. Klicken Sie im Menü **Einfügen** auf **Variablen** und geben Sie das Schema **articledata.xsd** als Struktur an, wenn Sie gefragt werden.



Im oben gezeigten Mapping geschieht Folgendes:

- Es wird kein Prioritätskontext mehr verwendet. Statt dessen gibt es eine Variable, die dieselbe Struktur wie die Zielkomponente hat.
- Das Mapping beginnt wie gewöhnlich am Ziel-Root-Node. Bevor die Zielkomponente befüllt wird, werden die Daten in der Variablen gesammelt.
- Die Variable wird im Kontext jedes einzelnen Produkts berechnet, da eine Verbindung von **product** zum **compute-when**-Input der Variablen besteht.
- Die Filterbedingung wird daher im Kontext jedes einzelnen Produkts überprüft. Nur, wenn die Bedingung "true" ist, wird die Variablenstruktur befüllt und an die Zielkomponente übergeben.

2.4.5 Mehrere Zielkomponenten

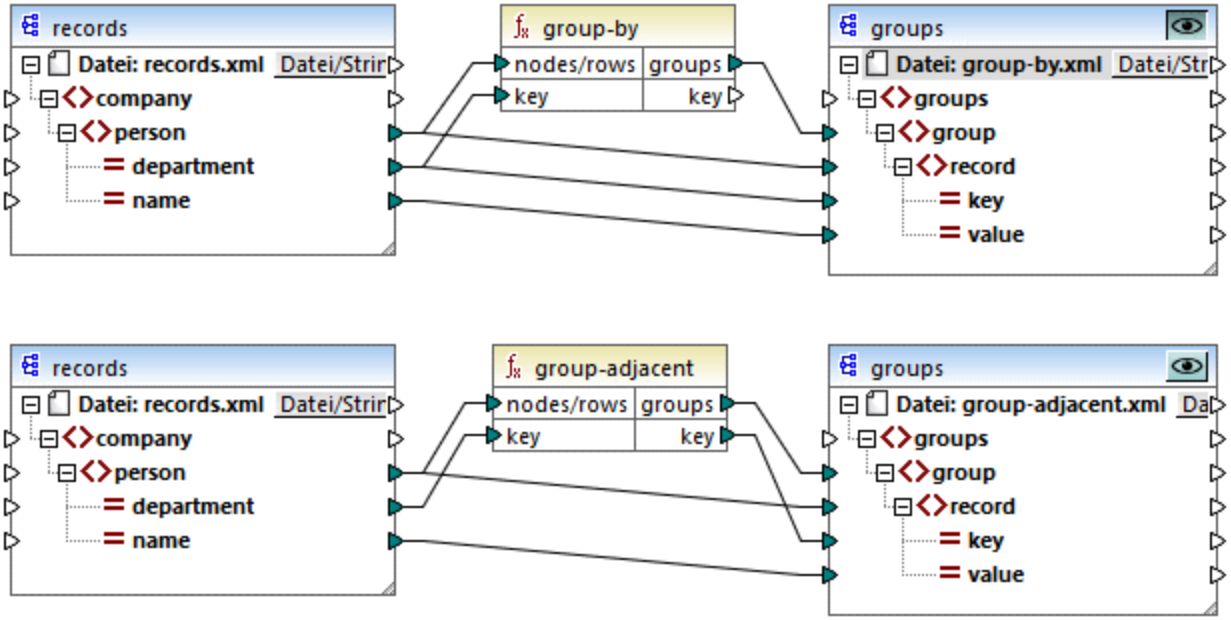
Ein Mapping kann mehrere Quell- und Zielkomponenten haben. Wenn mehrere Zielkomponenten vorhanden sind, können Sie in der MapForce-Vorschau immer nur eine Komponentenausgabe auf einmal anzeigen, nämlich diejenige, die Sie durch Klick auf die **Vorschau**-Schaltfläche markiert haben. In anderen Ausführungsumgebungen (MapForce Server oder generierter Code) werden alle Zielkomponente der Reihe nach ausgeführt und es wird die entsprechende Ausgabe für die einzelnen Komponente erzeugt.

 © 2019-2025 Altova GmbH

Standardmäßig werden Zielkomponenten von oben nach unten und von links nach rechts verarbeitet. Bei Bedarf können Sie diese Reihenfolge durch Ändern der Position der Zielkomponenten im Mapping-Fenster beeinflussen. Der Referenzpunkt ist jeweils die linke obere Ecke einer Komponente. Beachten Sie dazu Folgendes:

- Wenn zwei Komponenten dieselbe vertikale Position haben, so wird zuerst die oberste Komponente verarbeitet.
- Wenn zwei Komponenten dieselbe horizontale Position haben, so wird zuerst die am weitesten links liegende Komponente verarbeitet.
- In den seltenen Fällen, in denen zwei Komponenten sich an exakt derselben Stelle befinden, wird automatisch eine eindeutige interne Komponenten-ID verwendet. Damit ist eine genau definierte Reihenfolge definiert, die allerdings nicht geändert werden kann.

Um ein Beispiel dafür zu sehen, wie dies funktioniert, öffnen Sie das folgende Demo-Mapping: **<Dokumente>\Altova\MapForce2025\MapForceExamples\GroupingFunctions.mfd**. Dieses Mapping besteht aus mehreren Quell- und Zielkomponenten. Unten sehen Sie nur ein Fragment davon.



Gemäß der Regel ist die Standard-Verarbeitungsreihenfolge dieses Mappings in MapForce Server und im generierten Code von oben nach unten. Sie können dies überprüfen, indem Sie z.B. XSLT 2.0-Code generieren:

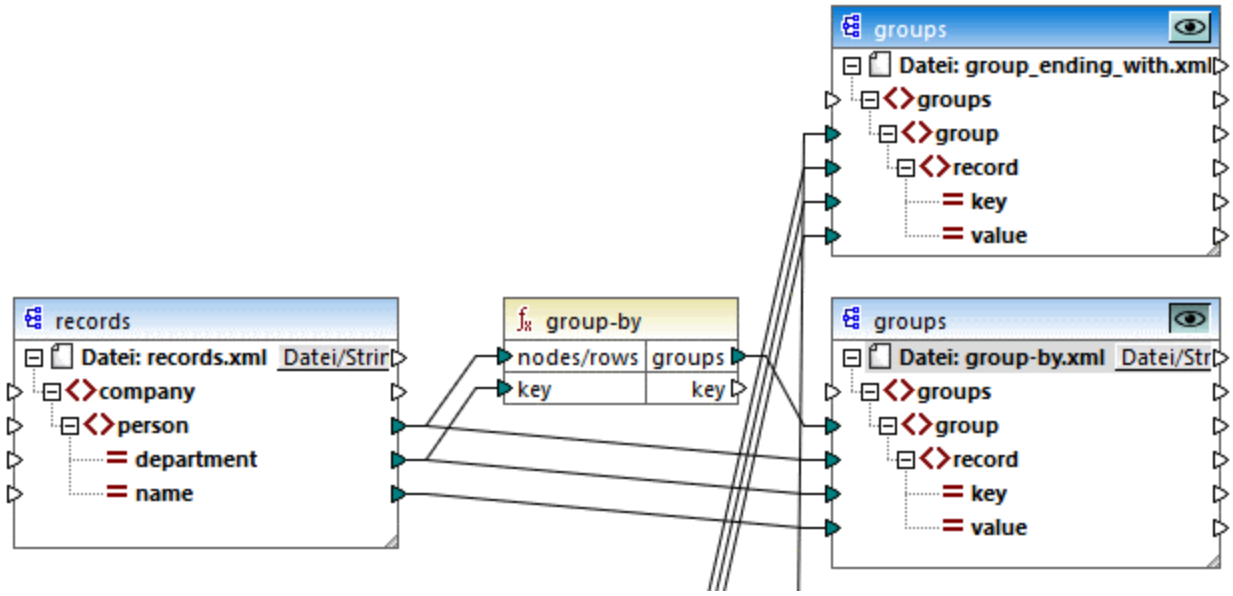
1. Klicken Sie im Menü **Datei** auf **Code generieren in | XSLT 2.0**.
2. Wenn Sie dazu aufgefordert werden, wählen Sie ein Zielverzeichnis für den generierten Code aus.

Das Zielverzeichnis enthält nach der Generierung mehrere XSLT-Dateien und eine **DoTransform.bat**-Datei. Letztere kann mit RaptorXML Server (eigene Lizenz erforderlich) ausgeführt werden. Die Datei **DoTransform.bat** verarbeitet Komponenten in derselben Reihenfolge, wie diese im Mapping definiert wurden, nämlich von oben nach unten. Überprüfen Sie dies, in dem Sie sich den `--output`-Parameter der einzelnen Transformationen ansehen.

```
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%
```

Mit der letzten Transformation wird eine Ausgabedatei namens **group-ending-with.xml** erzeugt. Verschieben wir diese Zielkomponente nun im Mapping ganz nach oben:



Wenn Sie nun erneut XSLT 2.0-Code generieren, ändert sich die Verarbeitungsreihenfolge entsprechend:

```
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%
```

 © 2019-2025 Altova GmbH

```
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%
```

Mit dem ersten Aufruf im oben gezeigten Codefragment wird nun **group-ending-with.xml** erzeugt.

Sie können die Verarbeitungsreihenfolge auch in anderen Codesprachen und in kompilierten MapForceServer-Ausführungsdateien (.mfx) auf ähnliche Weise ändern.

Verkettete Mappings

Dieselbe oben beschriebene Verarbeitungsreihenfolge gilt auch für verkettete Mappings. Die verkettete Mapping-Gruppe wird jedoch als eine einzige Einheit behandelt. Wenn Sie die Zwischen- oder Endkomponente eines einzigen verketteten Mappings verschieben, hat dies keine Auswirkung auf die Verarbeitungsreihenfolge. Die Position der Endkomponenten von einzelnen Gruppen hat nur dann einen Einfluss auf die Verarbeitungsreihenfolge, wenn mehrere Ketten bzw. mehrere Zielkomponenten in einem Mapping vorhanden sind.

- Wenn zwei Endkomponenten dieselbe vertikale Position einnehmen, so wird zuerst die weiter links gelegene verarbeitet.
- Wenn zwei Zielkomponenten dieselbe horizontale Position einnehmen, so wird zuerst die weiter oben gelegene verarbeitet.
- In den seltenen Fällen, in denen zwei Komponenten sich an exakt derselben Stelle befinden, wird automatisch eine eindeutige interne Komponenten-ID verwendet. Damit ist eine genau definierte Reihenfolge definiert, die allerdings nicht geändert werden kann.

3 Tutorials

Altova Website: MapForce Video-Demos

Die MapForce Tutorials sollen Ihnen dabei helfen, die grundlegenden Datentransformationsfunktionen von MapForce zu verstehen und verwenden zu können. Sie werden Schritt für Schritt durch die grundlegenden Funktionen geführt, wobei die Aufgaben stufenweise komplexer werden. Daher wird empfohlen, die Tutorials der Reihe nach durchzuarbeiten. Grundkenntnisse in XML und XML-Schema sind von Vorteil.

Beispieldateien
Die in diesen Tutorials gezeigten oder referenzierten Mapping-Dateien stehen im Ordner BasicTutorials 15 zur Verfügung. Wenn Sie sich nicht sicher sind, welche Auswirkungen Ihre Änderungen an den MapForce-Originaldateien haben werden, erstellen Sie Sicherungskopien der Originaldatei, bevor Sie damit zu arbeiten beginnen.

Liste der Tutorials

Eine Quellkomponente auf eine Zielkomponente
In diesem Tutorial 97 wird beschrieben, wie Sie die Nodes einer Quelldatei mit Hilfe grundlegender MapForce-Mechanismen auf die Nodes einer Zieldatei mappen. In diesem Tutorial wird anschließend erläutert, wie Sie eine durch ein bestimmtes XML-Schema definierte XML-Datei in eine durch ein anderes Schema definierte XML-Datei konvertieren.
Mehrere Quellkomponenten auf eine Zielkomponente
In diesem Tutorial 106 wird gezeigt, wie Sie Daten aus mehreren XML-Quelldateien in einer Zieldatei zusammenführen.
Verkettete Mappings
In diesem Tutorial 111 erstellen wir ein einfaches Mapping wie im zweiten Tutorial, filtern anschließend die mit diesem Mapping erzeugten Daten und übergeben diese dann an die zweite Zielkomponente.
Mehrere Quellkomponenten auf mehrere Zielkomponenten
In diesem Tutorial 120 erfahren Sie, wie Sie Daten aus mehreren XML-Instanzdateien aus demselben Ordner auslesen und diese in mehrere on-the-fly generierte XML-Dateien schreiben.

 © 2019-2025 Altova GmbH

3.1 Eine Quellkomponente auf eine Zielkomponente

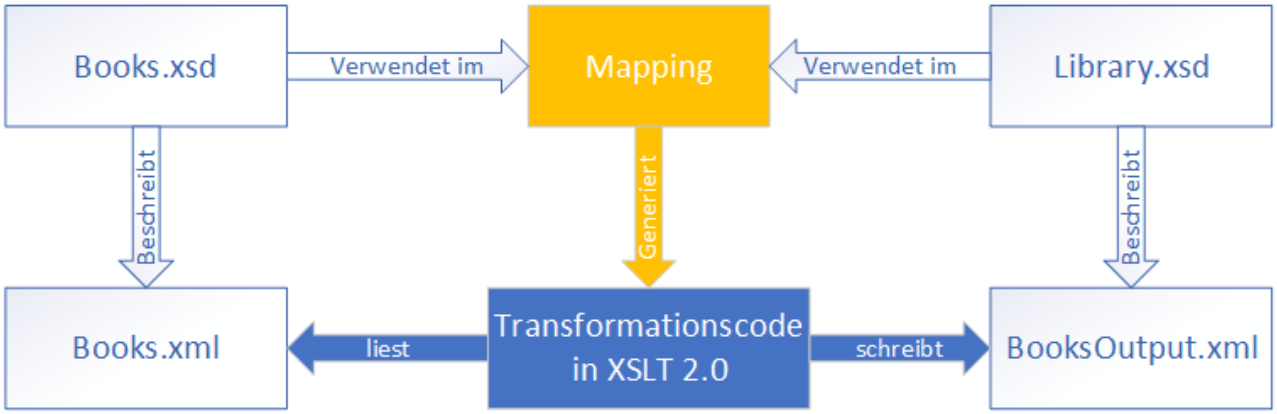
In diesem Tutorial wird beschrieben, wie Sie ein Mapping für eines der einfachsten Szenarien erstellen. Ziel ist es, die Daten aus der XML-Datei A, (der das XML-Schema A zugewiesen ist), in die XML-Datei B (der das XML-Schema B zugewiesen ist) zu übertragen. Im Prinzip wird dabei folgendermaßen vorgegangen:

1. Da wir zwei Datenstrukturen verwenden, werden wir in unserem Mapping-Design zwei Komponenten (eine *Quell-* und eine *Zielkomponente*) erstellen.
2. Um ein Dokument in ein anderes zu transformieren, wählen wir die geeignete Transformationssprache 16 aus.
3. Anschließend müssen wir die Quell-Nodes mit den gewünschten Ziel-Nodes verbinden. Diese Verbindungen bilden das Mapping und bestimmen, welcher Quell-Node auf welchen Ziel-Node gemappt wird.
4. Als Ergebnis des Mappings erhalten wir das XML-Zieldokument, das gemäß dem Zielschema gültig ist.
5. Schlussendlich können wir die XML-Ausgabedatei speichern.

Im abstrakten Modell unten sehen Sie, wie Datentransformationen durchgeführt werden.

Abstraktes Modell

Im unten gezeigten abstrakten Modell wird die Datentransformation in diesem Tutorial veranschaulicht:



Unser Mapping hat eine Quell- und eine Zielkomponente. Im Quellschema (**`Books.xsd`**) ist die Struktur der Quellinstanzdatei (**`Books.xml`**) beschrieben. Im Zielschema (**`Library.xsd`**) ist die Struktur der Zielinstanzdatei (**`BooksOutput.xml`**) beschrieben. Wenn Sie die Nodes der Quellkomponente mit den entsprechenden Nodes der Zielkomponente verbinden, generiert das Mapping Transformationscode in XSLT 3.0. Der Transformationscode liest Daten aus **`Books.xml`** aus und schreibt diese in **`BooksOutput.xml`**.

Quell- und Zieldatei

Das nachstehende Codefragment enthält die Beispieldaten aus der Datei **`Books.xml`**, die als Datenquelle verwendet wird.

```
<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>
```

So sollten die Daten in der Zieldatei namens **`BooksOutput.xml`** aussehen:

```
<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>
```

Unser Ziel ist, die Elemente `<author>`, `<title>`, `<genre>` und `<publish_year>` der Zieldatei mit dem Inhalt aus den entsprechenden Elementen in der Quelldatei (`<author>`, `<title>`, `<category>`, `<year>`) zu befüllen. Das Attribut `id` in der Quelldatei wird auf das Element `<id>` in der Zieldatei gemappt. Schließlich wird noch das Element `<last_updated>` der Zieldatei mit dem Datum und der Uhrzeit der letzten Aktualisierung der Datei befüllt.

Um die erforderliche Datentransformation durchzuführen, gehen Sie vor, wie in den nachfolgenden Unterabschnitten beschrieben.

3.1.1 Erstellen und Speichern eines Designs

In diesem Kapitel wird erläutert, wie Sie ein neues Design erstellen, eine Transformationssprache auswählen und Ihr Mapping validieren und speichern.

Erstellen eines neuen Designs

Um Daten transformieren zu können, müssen Sie ein neues Mapping-Design erstellen. Dies kann auf zwei Arten geschehen:

- Gehen Sie zum Menü **Datei** und klicken Sie auf **Neu**.

 © 2019-2025 Altova GmbH

- Klicken Sie in der Symbolleiste auf .

Wählen Sie eine Transformationssprache aus.

Je nach MapForce Edition stehen verschiedene Transformationssprachen 16 zur Verfügung. Für dieses Tutorial haben wir XSLT3 ausgewählt. Sie können diese Transformationssprache auf die folgenden zwei Arten auswählen:

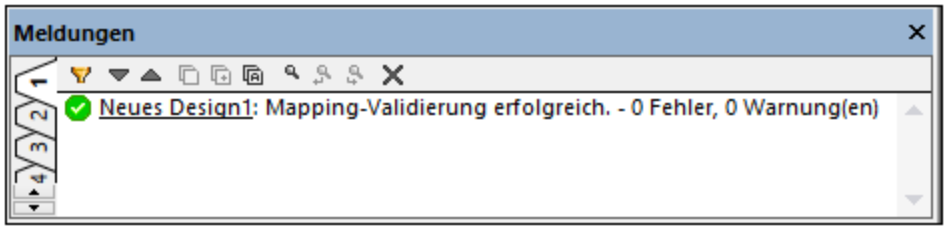
- Klicken Sie in der Symbolleiste auf **XSLT3**.
- Öffnen Sie das Menü **Ausgabe** und klicken Sie auf **XSLT 3.0**.

Validieren und Speichern des Designs

Die Validierung des Mappings ist ein optionaler Schritt, mit Hilfe dessen Sie potenzielle Mapping-Fehler- und Warnungen sehen und beheben können, bevor Sie das Mapping ausführen. Sie können Ihr Mapping jederzeit validieren. Um zu überprüfen, ob das Mapping gültig ist, wählen Sie eine der folgenden Methoden:

- Klicken Sie im Menü **Datei** auf **Mapping validieren**.
- Klicken Sie in der Symbolleiste auf .

Die Validierungsergebnisse werden im Fenster "Meldungen" folgendermaßen angezeigt:



Wählen Sie dazu eine der folgenden Methoden:

- Klicken Sie im Menü **Datei** auf **Speichern**.
- Klicken Sie in der Symbolleiste auf .

Das Mapping aus diesem Tutorial wurde unter dem Namen **`Tut1_OneToOne.mfd`** gespeichert.

3.1.2 Hinzufügen der Quellkomponente

Wir möchten nun eine XSD-Datei, die die Struktur der ersten Komponente bildet und eine XML-Datei, die die Daten für diese Komponente enthält, hinzufügen. Die Quelldatei namens **`Books.xsd`** kann auf eine der folgenden Arten zum Mapping hinzugefügt werden:

- Klicken Sie in der Symbolleiste auf (**XML-Schema/Datei einfügen**).
- Klicken Sie im Menü **Einfügen** auf **XML-Schema/Datei**.
- Ziehen Sie **`Books.xsd`** aus dem Windows Explorer in den Mapping-Bereich.

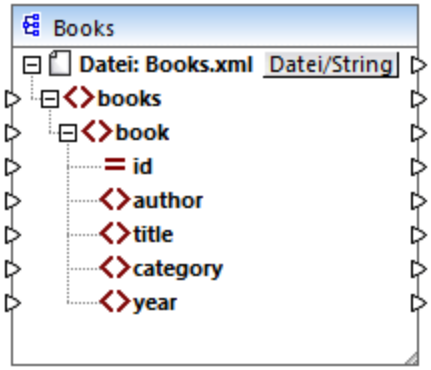
Wenn Sie eine der beiden ersten Optionen auswählen, werden Sie im Dialogfeld **XML-Schema-Datei einfügen** aufgefordert, zwischen einem vorinstallierten Schema und einer lokalen oder entfernten Datei zu wählen. In unseren Tutorials handelt es sich bei allen Dateien um lokale Dateien. Nähere Informationen zum Hinzufügen von XML-Dateien finden Sie unter XML- und XML-Schema 130.

Wenn eine Komponente anhand einer XSD-Datei erstellt wird, werden Sie aufgefordert, eine XML-Datei als Datendatei für diese Komponente anzugeben. Wenn eine Komponente anhand einer XML-Datei erstellt wird, wird die Struktur der Daten der Komponente anhand der von der XML-Datei referenzierten XSD-Datei definiert. Wenn keine Referenz auf eine XSD-Datei vorhanden ist, werden Sie gefragt, ob MapForce eine XSD-Datei für diese Komponente generieren soll.

Da wir zuerst die Schema-Datei hinzufügen, schlägt MapForce vor, eine XML-Beispieldatei hinzuzufügen. Klicken Sie auf **Durchsuchen** und navigieren Sie zur Datei **`Books.xml`** im selben Ordner. Unsere Quelldatei enthält nun sowohl ein Schema als auch Inhalt.

Anzeigen der Struktur

Nachdem Sie die Quelldatei nun zum Mapping-Bereich hinzugefügt haben, sehen Sie ihre Struktur. Diese Struktur wird in MapForce als Mapping-Komponente oder einfach Komponente 29 bezeichnet. Durch Klick auf das ⊞-Symbol können Sie Elemente in der Komponente erweitern. Alternativ dazu können Sie die **+**-Taste des Ziffernblocks drücken. In der Abbildung unten sehen Sie die Quellkomponente:



Der Name der Überschrift (`Books`) bezieht sich nur auf den Namen der Komponente und nicht auf den Namen des Schemas, auf dem diese Datei basiert. Um den Namen des Schemas und anderer Eigenschaften der Komponente zu sehen, doppelklicken Sie auf die Überschrift der Komponente. Daraufhin wird das Dialogfeld "Komponenteneinstellungen" 131 geöffnet.

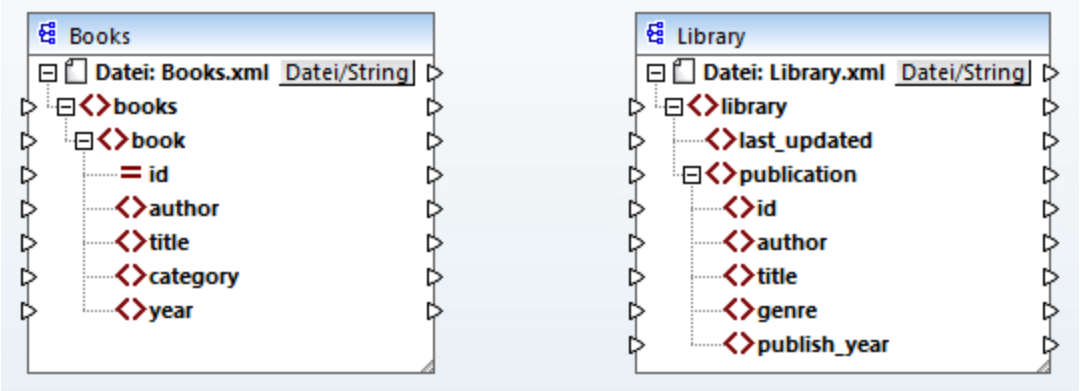
Der oberste Node der Komponente (**`Datei: Books.xml`**) ist der Name der XML-Instanzdatei. Die XML-Elemente in der Struktur werden durch das Symbol gekennzeichnet. XML-Attribute werden durch das Symbol gekennzeichnet. Die kleinen Dreiecke an beiden Seiten der Komponente repräsentieren auf der linken Seite Daten-Inputs und auf der rechten Seite Daten-Outputs. Diese Dreiecke werden in MapForce als *Input*- bzw. *Output-Konnektoren* bezeichnet.

Nähere Informationen zu Komponenten, Verbindungen, allgemeine Verfahren und Funktionalitäten finden Sie unter Mapping-Grundlagen 29.

 © 2019-2025 Altova GmbH

3.1.3 Hinzufügen der Zielkomponente

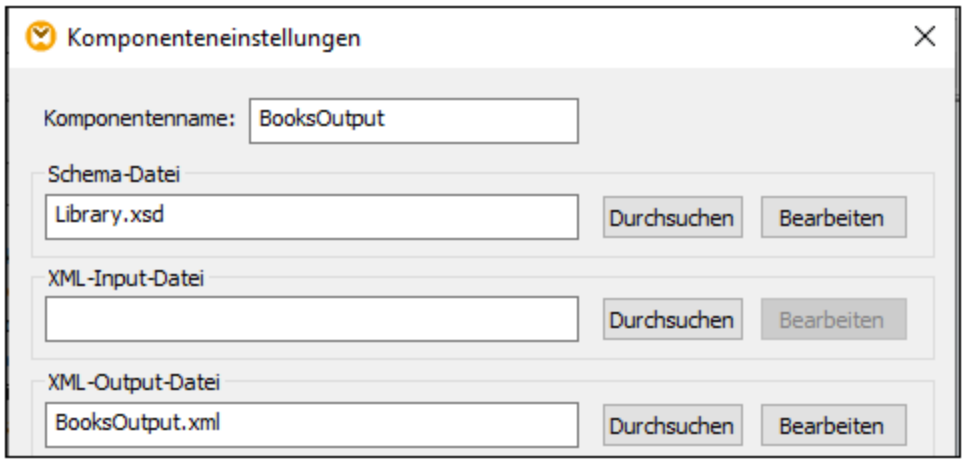
In nächsten Schritt wird nun eine Zielkomponente hinzugefügt und ihre Einstellungen definiert. Fügen Sie die Zieldatei **`Library.xsd`** zum Mapping hinzu. Klicken Sie auf **Überspringen**, wenn Sie von MapForce aufgefordert werden, eine Instanzdatei bereitzustellen. Zu diesem Zeitpunkt sieht das Mapping-Design folgendermaßen aus:



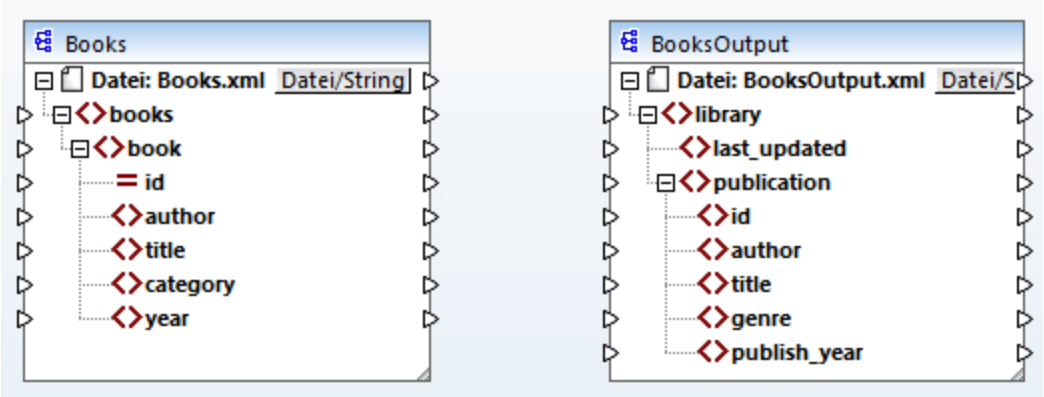
Wenn Sie **`Library.xsd`** öffnen, wird sie in der Komponente als XML-Datei angezeigt. Tatsächlich erzeugt MapForce nur eine Referenz auf die XML-Datei namens **`Library.xml`**, wobei diese XML-Datei selbst noch gar nicht existiert. Unsere Zielkomponente hat also ein Schema aber keinen Inhalt.

Komponenteneinstellungen

Wir müssen die Zielkomponente in **`BooksOutput`** umbenennen. Die Ausgabedatei erhält den Namen **`BooksOutput.xml`**. Dadurch vermeiden, wir, dass es in den nächsten Tutorials zu Verwirrung kommt, da wir dort eine separate Datei namens **`Library.xml`** verwenden werden, die ihren eigenen Inhalt hat und auf demselben Schema **`Library.xsd`** basiert. Um die Zielkomponente umzubenennen, doppelklicken Sie auf die Überschrift der Zielkomponente. Daraufhin wird das Dialogfeld Komponenteneinstellungen 131 (*siehe Abbildung unten*) geöffnet, in dem wir den Namen der Zieldatei und den Namen der Zieldatei folgendermaßen ändern:



Das Mapping-Design sieht nun folgendermaßen aus:

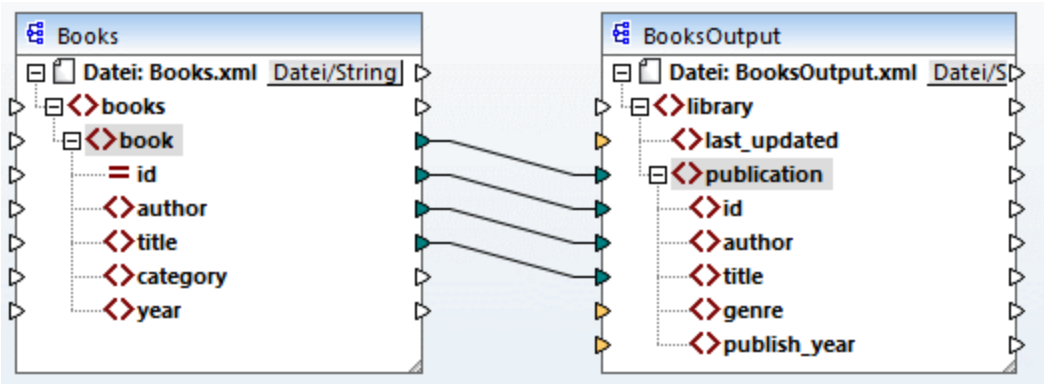


3.1.4 Verbinden von Quell- und Zielkomponente

In diesem Schritt werden wir die Daten in der Quelldatei auf die Zieldatei mappen. Außerdem stellen wir mit Hilfe der XPath-Funktion `current-dateTime` 344 Informationen über das aktuelle Datum und die aktuelle Uhrzeit bereit.

Automatische Verbindungen

Wir werden nun eine Mapping-Verbindung zwischen dem `<book>`-Element in der Quellkomponente und dem `<publication>`-Element in der Zielkomponente erstellen. Klicken Sie dazu auf den Output-Konnektor (das kleine Dreieck) rechts vom Element `<book>` und ziehen Sie die Linie auf den Input-Konnektor des Elements `<publication>` in der Zielkomponente. Bei Ziehen der Verbindungslinie verbindet MapForce unter Umständen automatisch alle Nodes von `<book>` in der Quelldatei mit den gleichnamigen Nodes in der Zieldatei. In unserem Beispiel wurden vier Verbindungen gleichzeitig erstellt (*siehe Abbildung unten*). Dieses Funktion hat den Namen *Idente Sub-Einträge automatisch verbinden* 52 und kann deaktiviert und gegebenenfalls angepasst werden.



Sie können **Idente Sub-Einträge automatisch verbinden** auf eine der folgenden Arten aktivieren bzw. deaktivieren:

 © 2019-2025 Altova GmbH

- Klicken in der Symbolleiste auf (**Aktiviert/Deaktiviert die automatische Verbindung von Sub-Einträgen**.
- Klicken Sie im Menü **Verbindung** auf den Befehl **Idente Sub-Einträge automatisch verbinden**.

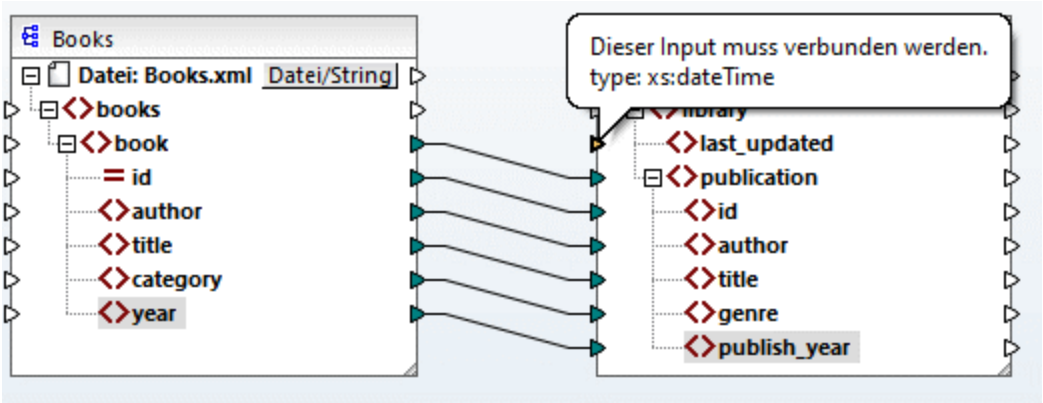
Verbinden obligatorischer Datenelemente

Beachten Sie, dass einige der Input-Konnektoren der Zielkomponente von MapForce orange markiert wurden. Dies bedeutet, dass diese Datenelemente zwingend erforderlich sind. Sie sind obligatorisch, da Sie im Schema der Datei als obligatorisch definiert wurden. Damit die XML-Zieldatei gültig ist, müssen Sie für die obligatorischen Datenelemente auf folgende Art Werte bereitstellen:

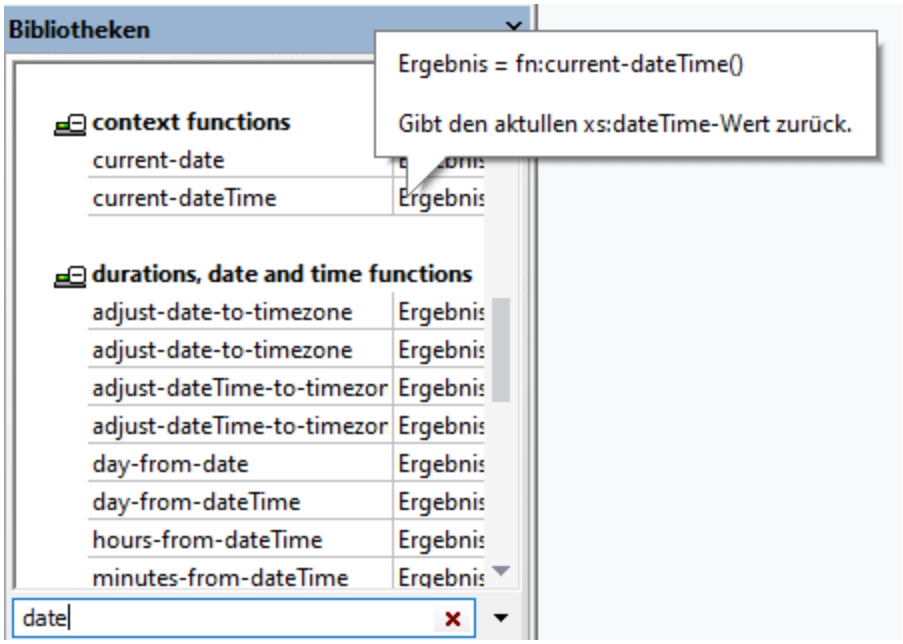
- Verbinden Sie das Datenelement `<category>`in der Quellkomponente mit dem Datenelement `<genre>` in der Zielkomponente.
- Verbinden Sie das Datenelement `<year>` in der Quellkomponente mit dem Datenelement `<publish_year>` in der Zielkomponente.

Hinzufügen des aktuellen Datums und der aktuellen Uhrzeit

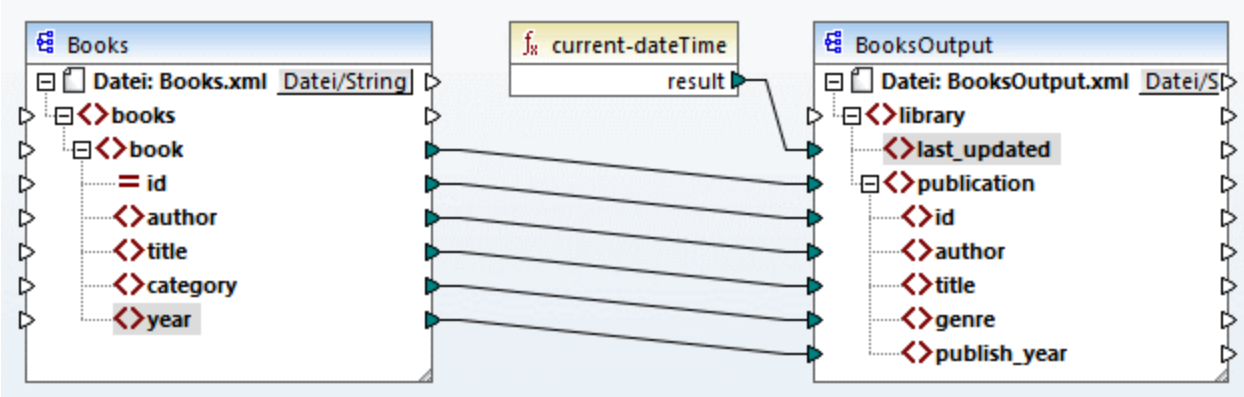
Sie müssen nun nur noch einen Wert für das Element `<last_updated>` bereitstellen. Wenn Sie die Maus über seinen Input-Konnektor platzieren, sehen Sie, dass das Element den Typ `xs:dateTime` hat (*siehe Abbildung unten*). Damit Tipps angezeigt werden, aktivieren Sie die Symbolleisten-Schaltfläche . Durch Klick auf (**Datentypen anzeigen**) in der Symbolleiste können die Datentypen der einzelnen Datenelemente jederzeit sichtbar gemacht werden.



Sie können das aktuelle Datum und die aktuelle Uhrzeit mit Hilfe der **`current-dateTime`**-Funktion abrufen. Um diese Funktion zu suchen, geben Sie den Namen der Funktion in das Textfeld im unteren Bereich des Fensters "Bibliotheken" [20] ein (*siehe Abbildung unten*). Doppelklicken Sie alternativ dazu auf einen leeren Bereich im Mapping und beginnen Sie mit der Eingabe von `current-date`.



Um die Funktion zum Mapping hinzuzufügen, ziehen Sie diese in den Mapping-Bereich. Verbinden Sie anschließend ihren Output mit dem Input des Elements `<last_updated>` (*siehe Abbildung unten*).



Sie können Ihr Mapping jetzt, wie in Erstellen und Speichern eines Designs 98 beschrieben, validieren und speichern.

3.1.5 Vorschau auf das Mapping-Ergebnis

Mit Hilfe seiner integrierten Prozessoren können Sie in MapForce direkt im Ausgabefenster eine Vorschau auf das Mapping-Ergebnis anzeigen (*siehe Abbildung unten*).

 © 2019-2025 Altova GmbH

```
3     <last_updated>2021-07-19T11:14:08+02:00</last_updated>
4     <publication>
5         <id>1</id>
6         <author>Mark Twain</author>
7         <title>The Adventures of Tom Sawyer</title>
8         <genre>Fiction</genre>
9         <publish_year>1876</publish_year>
10    </publication>
11    <publication>
12        <id>2</id>
13        <author>Franz Kafka</author>
14        <title>The Metamorphosis</title>
15        <genre>Fiction</genre>
16        <publish_year>1912</publish_year>
17    </publication>
```

Mapping | XSLT3 | DB-Abfrage | **Ausgabe**

Tut1-OneToOne.mfd

Ausgabe speichern

Standardmäßig werden die im Ausgabefenster angezeigten Dateien nicht auf der Festplatte gespeichert. MapForce erstellt statt dessen temporäre Dateien. Um die Ausgabe zu speichern, öffnen Sie das Ausgabefenster und wählen Sie den Menübefehl **Ausgabe | Ausgabedatei speichern** oder klicken Sie in der Symbolleiste auf (**Generierte Ausgabe speichern**).

Um MapForce so zu konfigurieren, dass die Ausgabe, anstatt in einer temporären Datei, direkt in einer endgültigen Datei gespeichert wird, gehen Sie zu **Extras | Optionen | Allgemein** und aktivieren Sie das Kontrollkästchen *Direkt in die endgültigen Output-Dateien schreiben*. Beachten Sie, dass nicht empfohlen wird, diese Option zu aktivieren, während Sie das Tutorial durchnehmen, da Sie dadurch die Originaldateien eventuell unabsichtlich überschreiben könnten.

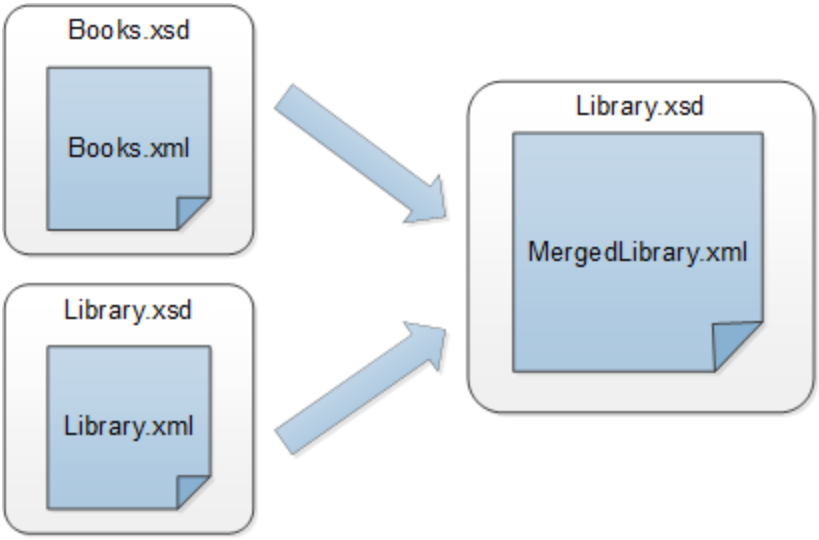
Vorschau auf den generierten Code

Sie können auch eine Vorschau des generierten XSLT-Codes, der die Transformation durchführt, anzeigen. Um eine Vorschau auf den Code zu sehen, öffnen Sie den XSLT3-Bereich am unteren Rand des Mapping-Fensters. Um den XSLT3-Code zu generieren und in einer Datei zu speichern, wählen Sie den Menübefehl **Datei | Code generieren in | XSLT 3.0**. Wenn Sie dazu aufgefordert werden, geben Sie einen Ordner an, in dem der generierte Code gespeichert werden soll. Nach Fertigstellung der Codegenerierung enthält der Zielordner die folgenden beiden Dateien:

1. Eine nach dem Zielschema benannte XSLT-Transformationsdatei. Diese Transformationsdatei hat das folgende Format: **`MappingMapTo<TargetFileName>.xslt`**.
2. Die Datei **`DoTransform.bat`**, mit der Sie die XSLT-Transformation mit Altova RaptorXML Server über die Befehlszeile ausführen können. Um den Befehl ausführen zu können, müssen Sie RaptorXML installieren.

3.2 Mehrere Quellkomponenten auf eine Zielkomponente

In diesem Tutorial erfahren Sie, wie Sie die Daten aus einer neuen Datei namens **Library.xml** mit den Daten aus **Books.xml** zusammenführen. Es wird eine Zieldatei namens **MergedLibrary.xml** erzeugt, die die Daten aus beiden Quelldateien enthält. Die Zieldatei basiert auf dem Schema **Library.xsd**. Beachten Sie, dass beide Quelldateien unterschiedliche Schemas haben. Wenn die Quelldateien dasselbe Schema hätten, könnten Sie ihre Daten auch auf andere Art miteinander zusammenführen (siehe Mehrere Quellkomponenten auf mehrere Zielkomponenten 120). In der Abbildung unten sehen Sie ein abstraktes Modell der in diesem Tutorial beschriebenen Datentransformation.



Das nachstehende Codefragment enthält einen Ausschnitt aus der Datei **Books.xml**, der als die erste Datenquelle verwendet wird.

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

Das nachstehende Codefragment enthält einen Ausschnitt aus aus der Datei **Library.xml**, der als die zweite Datenquelle verwendet wird.

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

 © 2019-2025 Altova GmbH

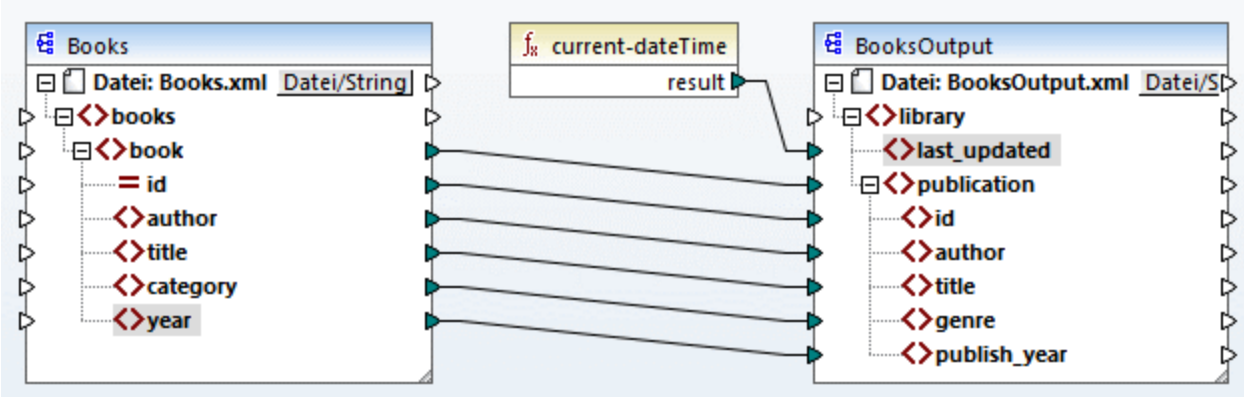
So sollten die zusammengeführten Daten in der Zieldatei namens **MergedLibrary.xml** aussehen:

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

Um die Datentransformation durchzuführen, gehen Sie vor, wie in den nachfolgenden Unterabschnitten beschrieben.

3.2.1 Vorbereiten des Mapping-Designs

Die Ausgangsbasis für dieses Tutorial bildet das Mapping **Tut1_OneToOne.mfd** (*Abbildung unten*), das im Tutorial 1 97 erstellt wurde. Bevor Sie Änderungen am Mapping vornehmen, sollten Sie das Design unter einem neuen Namen im Ordner **Basic Tutorials** speichern.



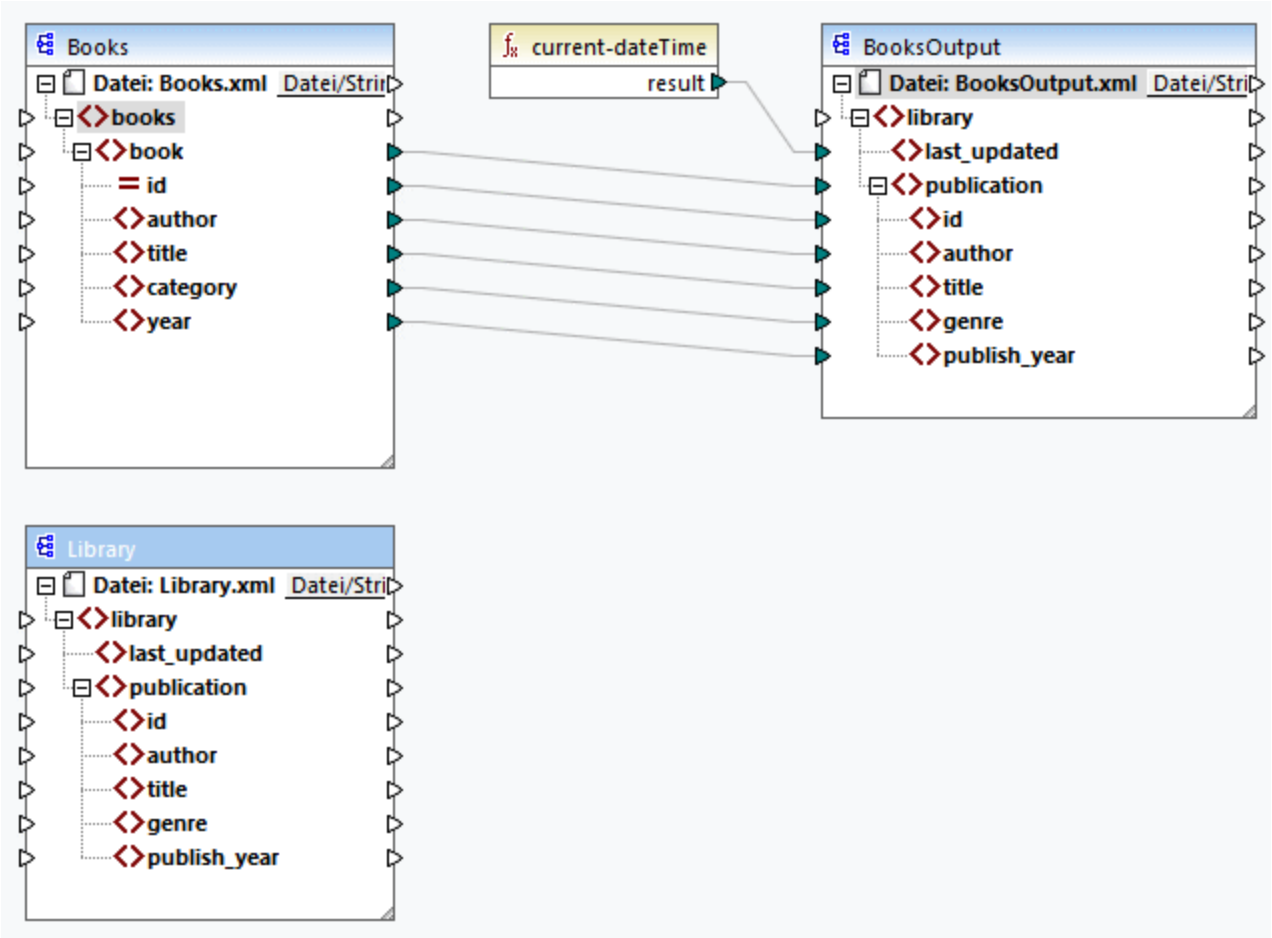
3.2.2 Hinzufügen einer zweiten Quellkomponente

Im nächsten Schritt wird nun eine zweite Quellkomponente hinzugefügt. Fügen Sie **Library.xml** zum Mapping hinzu. Da das Schema (**Library.xsd**) in dieser XML-Datei referenziert wird, müssen Sie es nicht in einem

separaten Schritt hinzufügen. Um zu überprüfen, ob die Schemareferenz korrekt ist, öffnen Sie die Komponenteneinstellungen 131.

Klicken Sie nun auf die Überschrift der neuen Komponente und ziehen Sie die Komponente mit der Maus unter die Komponente `Books`. Sie können Komponenten jederzeit beliebig verschieben. Wenn Sie eine Quellkomponente jedoch links von der Zielkomponente platzieren, wird das Mapping übersichtlicher und verständlicher. Dies ist auch die Konvention bei allen in dieser Dokumentation gezeigten Mappings und bei allen mit MapForce installierten Beispielmappingdateien.

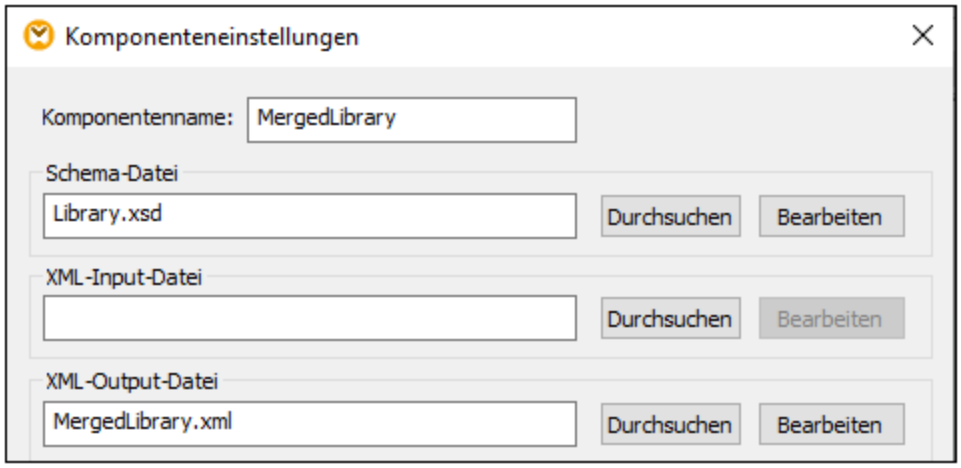
Zu diesem Zeitpunkt sieht das Mapping-Design folgendermaßen aus:



3.2.3 Konfigurieren der Ausgabe

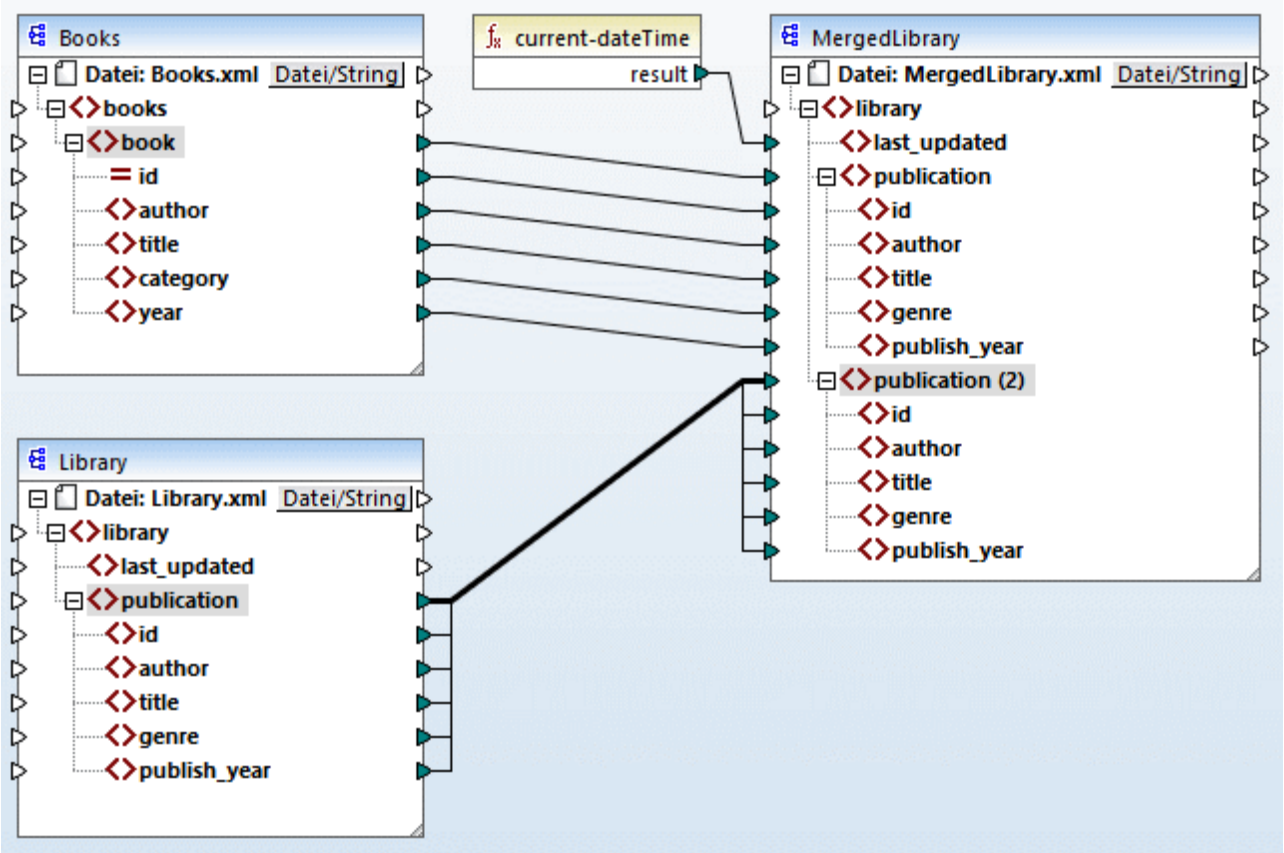
Das Mapping hat nun zwei Quellkomponenten: (`Books` und `Library`) und eine Zielkomponente (`BooksOutput`). Aus Gründen der Konsistenz und Klarheit müssen wir die Einstellungen der Komponente `BookOutput` ändern. Doppelklicken Sie auf die Überschrift der Zielkomponente. Daraufhin wird das Dialogfeld "Komponenteneinstellungen" 131 geöffnet. Konfigurieren Sie die Einstellungen, wie unten gezeigt.

 © 2019-2025 Altova GmbH



3.2.4 Verbinden der zweiten Quellkomponente mit der Zielkomponente

Im letzten Schritt des Tutorials wird die zweite Quellkomponente (`Library`) mit der Zielkomponente (`MergedLibrary`) verbunden. Verbinden Sie dazu des Element `<publication>` in **`Library.xml`** mit dem Element `<publication>` in **`MergedLibrary.xml`**. Da der Input-Konnektor der Zielkomponente bereits eine Verbindung hat, werden Sie aufgefordert, die Verbindung zu ersetzen oder ein Duplikat des Inputs zu erzeugen. Unser Ziel in diesem Tutorial ist es, Daten aus zwei Quellkomponenten auf eine Zielkomponente zu mappen. Klicken Sie daher auf **Duplikat erzeugen**. Dadurch konfigurieren Sie die Zielkomponente so, dass sie auch Daten aus der zweiten Quellkomponente erhält. Das Mapping sieht nun folgendermaßen aus:



In der Abbildung oben sehen Sie, dass das Element `publication` in der Zielkomponente nun dupliziert wurde. Der neue Node `publication(2)` erhält nun Daten aus **`Library.xml`**. Beachten Sie außerdem, dass als Name dieses Node im Mapping zwar `publication(2)` angezeigt wird, dass der Name in der XML-Zielkomponente aber, wie beabsichtigt, `publication` lautet.

Alles kopieren-Verbindung

Da die Child-Elemente des `publication`-Elements in der Komponente `Library` und die des `publication`-Elements in der Komponente `MergedLibrary` denselben Namen und Datentyp haben, werden diese Elemente mit einer einzigen dicken Linie miteinander verbunden. Eine solche Verbindung wird als *"Alles kopieren"-Verbindung* 54 bezeichnet. Sie macht das Mapping übersichtlicher.

Ausgabevorschau

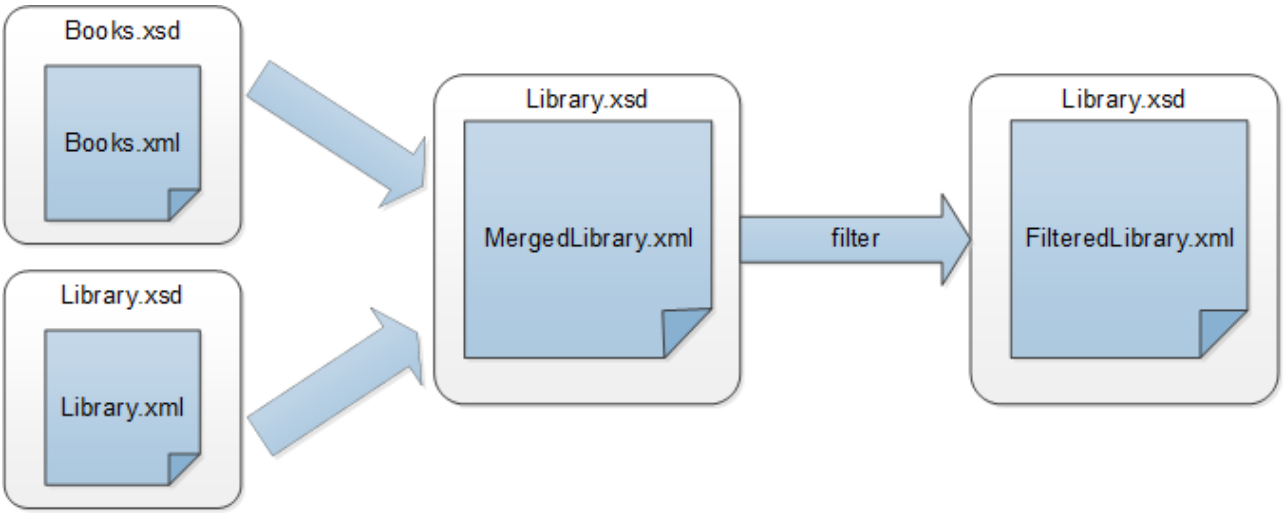
Öffnen Sie das Ausgabefenster, um das Ergebnis zu sehen. Sie werden sehen, dass die Daten aus den beiden Dateien **`Books.xml`** und **`Library.xml`** nun in der neuen Datei **`MergedLibrary.xml`** zusammengeführt wurden. Das Mapping-Design aus diesem Tutorial wurde unter dem Namen **`Tut2_MultipleToOne.mfd`** gespeichert. Dieses Mapping dient als Ausgangsbasis für das nächste Tutorial 111.

 © 2019-2025 Altova GmbH

3.3 Verkettetes Mapping

Altova Website: Video-Tutorial zu einem verketteten Mapping

In diesem Tutorial wird beschrieben, wie Sie mit mehreren Zielkomponenten arbeiten. Ziel dieses Tutorials ist es, Daten aus zwei Quellkomponenten in einer Zielkomponente zusammenzuführen, die Daten dieser Zielkomponente anschließend zu filtern und die gefilterten Daten an die zweite Zielkomponente zu übergeben. In der Abbildung unten sehen Sie ein abstraktes Modell der in diesem Tutorial beschriebenen Datentransformation.



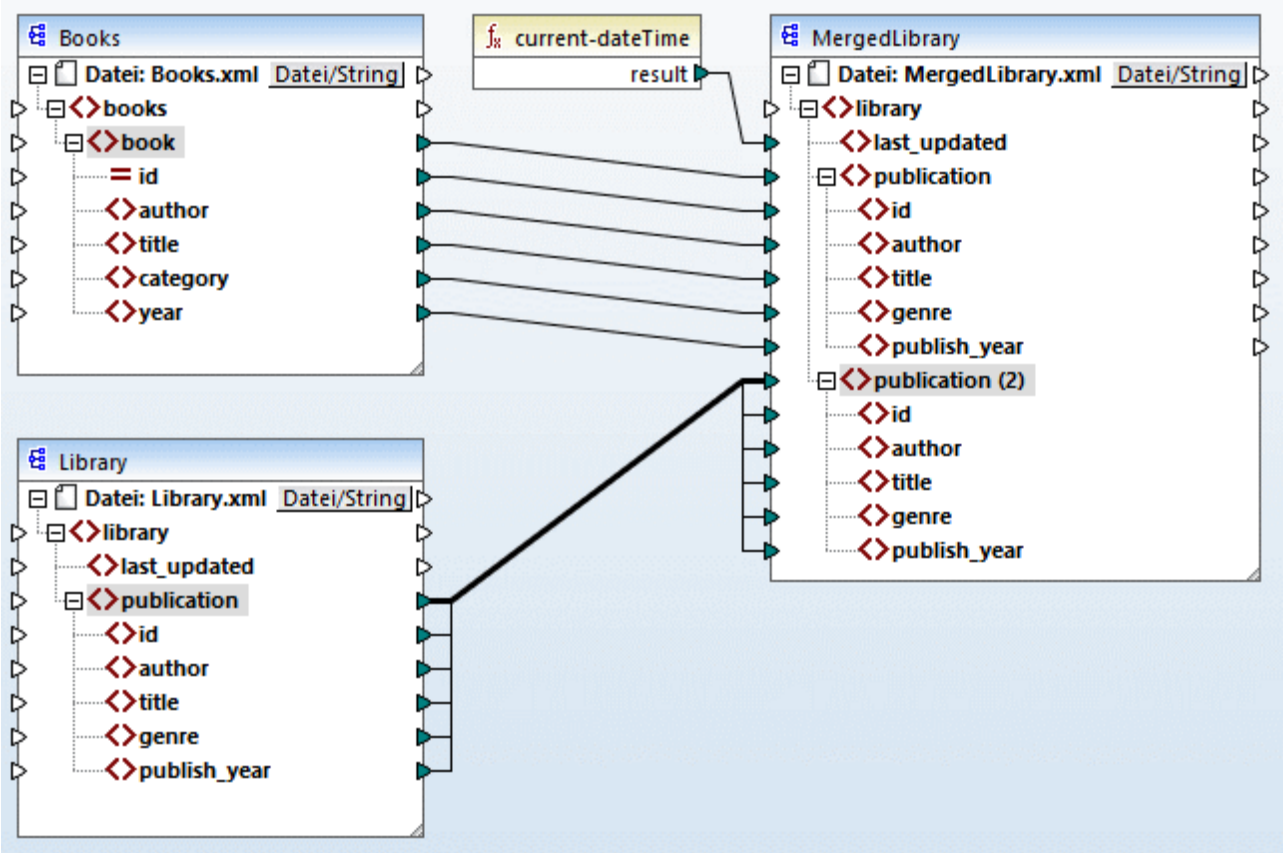
Im obigen Diagramm werden zuerst die Daten aus zwei Quelldateien (**`Books.xml`** und **`Library.xml`**) in einer einzigen Zieldatei namens **`MergedLibrary.xml`** zusammengeführt. Anschließend werden die Daten mit Hilfe einer Filterfunktion transformiert und an die nächste Komponente namens **`FilteredLibrary.xml`**übergeben. Beachten Sie, dass **`FilteredLibrary.xml`** auf dem Schema **`Library.xsd`** basiert. Die Zwischenkomponente dient sowohl als Datenziel als auch als Datenquelle. Dieses Verfahren wird in MapForce als *verkettetes Mapping* bezeichnet. Die Zwischenresultate von verketteten Mappings können in einer Vorschau angezeigt und das Zwischenergebnis (in unserem Fall **`MergedLibrary.xml`**) und das Ergebnis der letzten Zielkomponente (in unserem Fall **`FilteredLibrary.xml`**) können gespeichert werden.

Um die Datentransformation durchzuführen, gehen Sie vor, wie in den nachfolgenden Unterabschnitten beschrieben.

Ein weiteres Beispiel für ein verkettetes Mapping sehen Sie im am oberen Rand der Seite verlinkten Video-Tutorial.

3.3.1 Vorbereiten des Mapping-Designs

Als Ausgangsbasis für dieses Tutorial wird das Mapping **`Tut2_MultipleToOne.mfd`** (*siehe Abbildung unten*) verwendet. Dieses Mapping wurde im vorherigen Tutorial 106 erstellt. Bevor Sie Änderungen am Mapping vornehmen, sollten Sie das Design unter einem neuen Namen im Ordner **`Basic Tutorials`** speichern.



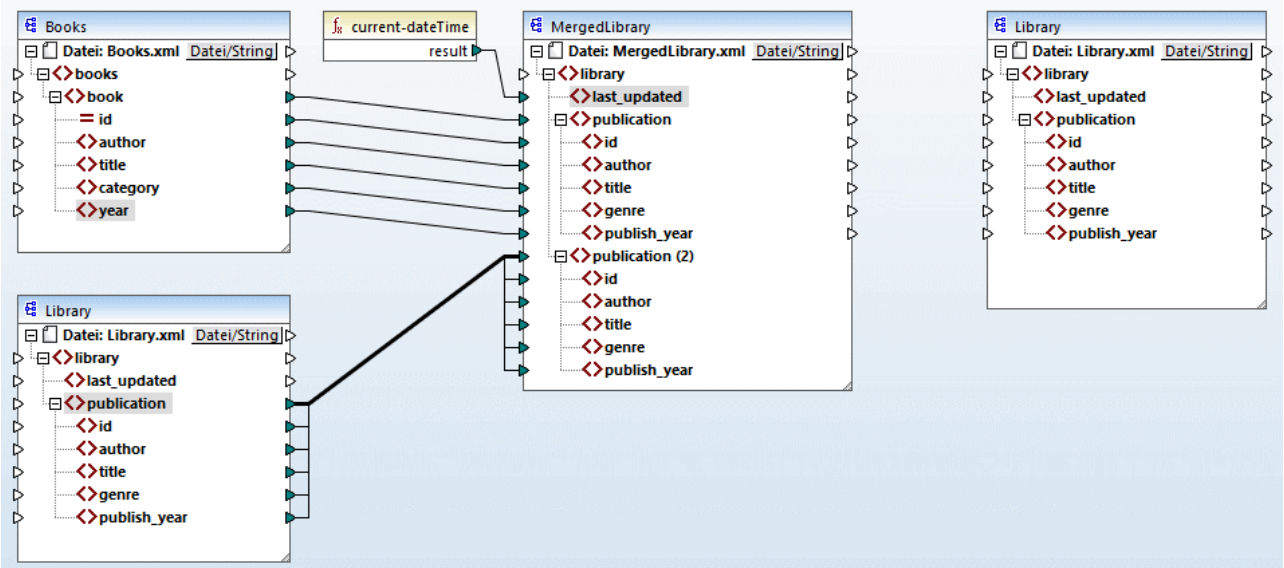
3.3.2 Konfigurieren der zweiten Zieldatei

Im nächste Schritt müssen wir nun die zweite Zieldatei, die nur eine Teilmenge der `publication`-Daten aus **`MergedLibrary.xml`** enthalten wird, hinzufügen und konfigurieren.

Hinzufügen der zweiten Zielkomponente

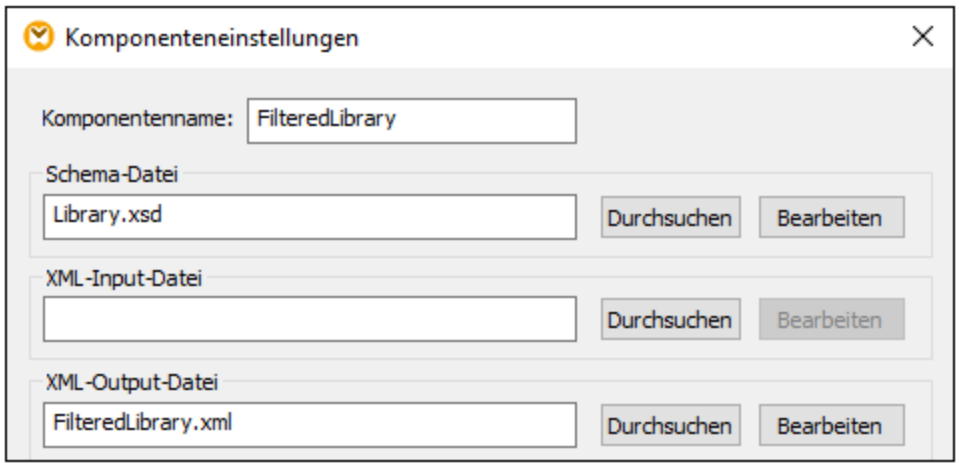
Fügen Sie **`Library.xsd`** zum Mapping hinzu und klicken Sie auf **Überspringen**, wenn Sie aufgefordert werden, eine Beispielinstanzdatei bereitzustellen. Die zweite Zielkomponente hat nur eine Struktur, aber keinen Inhalt. Wir werden die gefilterten Daten später auf diese Zieldatei mappen. Das Mapping-Design sieht nun folgendermaßen aus:

 © 2019-2025 Altova GmbH



Konfigurieren der zweiten Zielkomponente

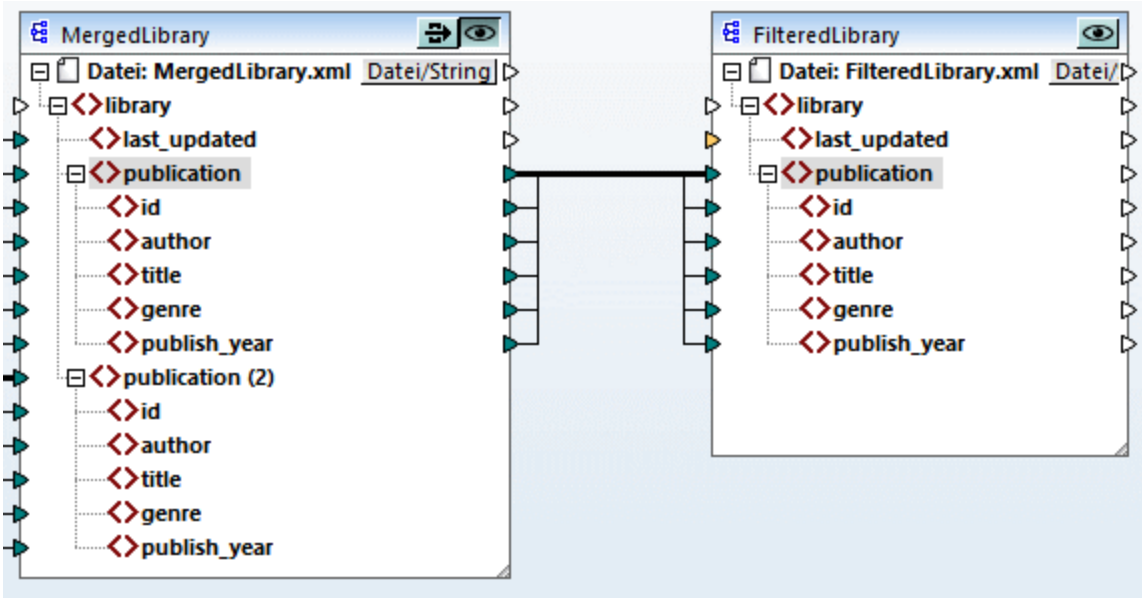
Wie oben gezeigt, hat das Mapping nun zwei Quellkomponenten (`Books` und `Library`) und zwei Zielkomponenten (`MergedLibrary` und `Library`). Aus Gründen der Klarheit benennen wir die neu hinzugefügte Komponente in `FilteredLibrary` *um.* Doppelklicken Sie dazu auf die Überschrift der Komponente ganz rechts und bearbeiten Sie die Komponenteneinstellungen 131 wie folgt:



3.3.3 Ziehen der Verbindungen

Im nächsten Schritt wird das Element `publication` in `MergedLibrary` auf das Element `publication` in `FilteredLibrary` gemappt. Wenn Sie den Output-Konnektor von `MergedLibrary` mit dem Input-Konnektor von `FilteredLibrary` verbinden, werden Sie informiert, dass Sie im Mapping mehrere Zielkomponenten erstellt

haben. Beachten Sie, dass in der rechten oberen Ecke beider Zielkomponenten nun neue Schaltflächen zur Verfügung stehen: (**Vorschau**) und (**Weiterleitung**). Diese Schaltflächen werden in den folgenden Schritten verwendet und erklärt.



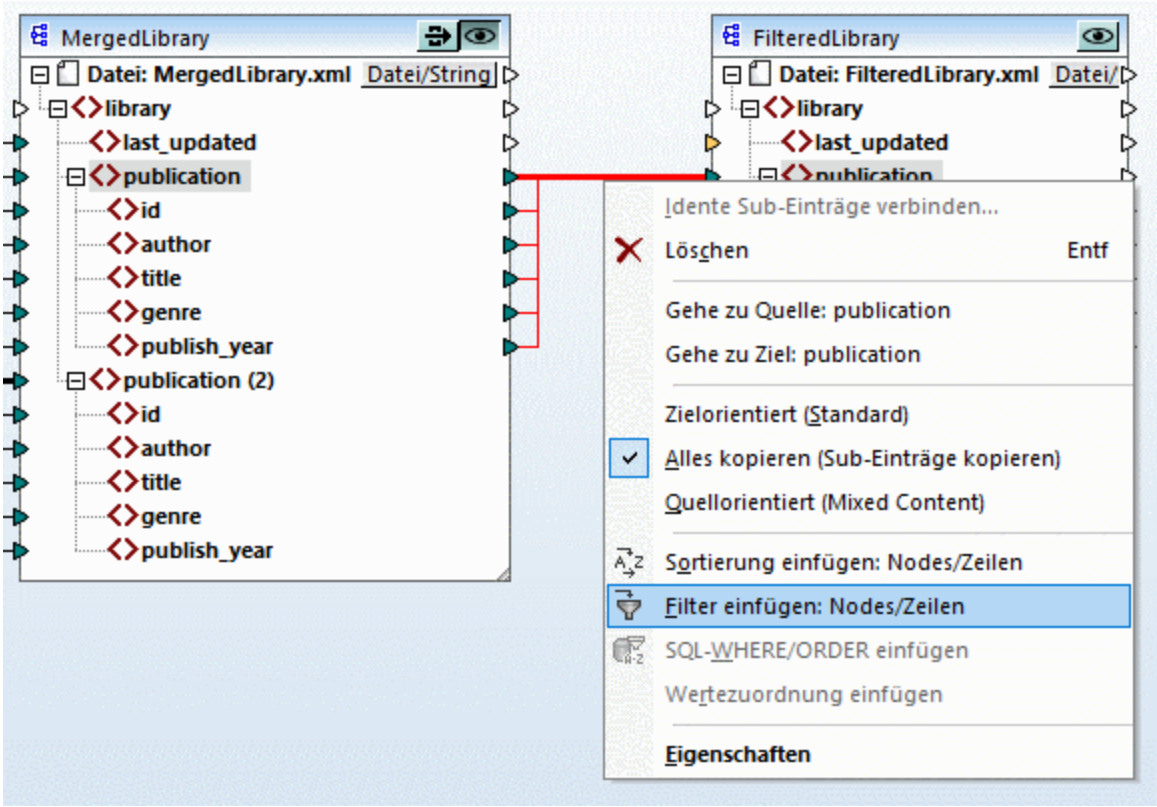
3.3.4 Filtern der Daten

In diesem Schritt werden wir die Daten aus `MergedLibrary` so filtern, dass nur die nach 1900 veröffentlichten Bücher an die Komponente `FilteredLibrary` übergeben werden. Zu diesem Zweck werden wir eine Filter-Komponente verwenden.

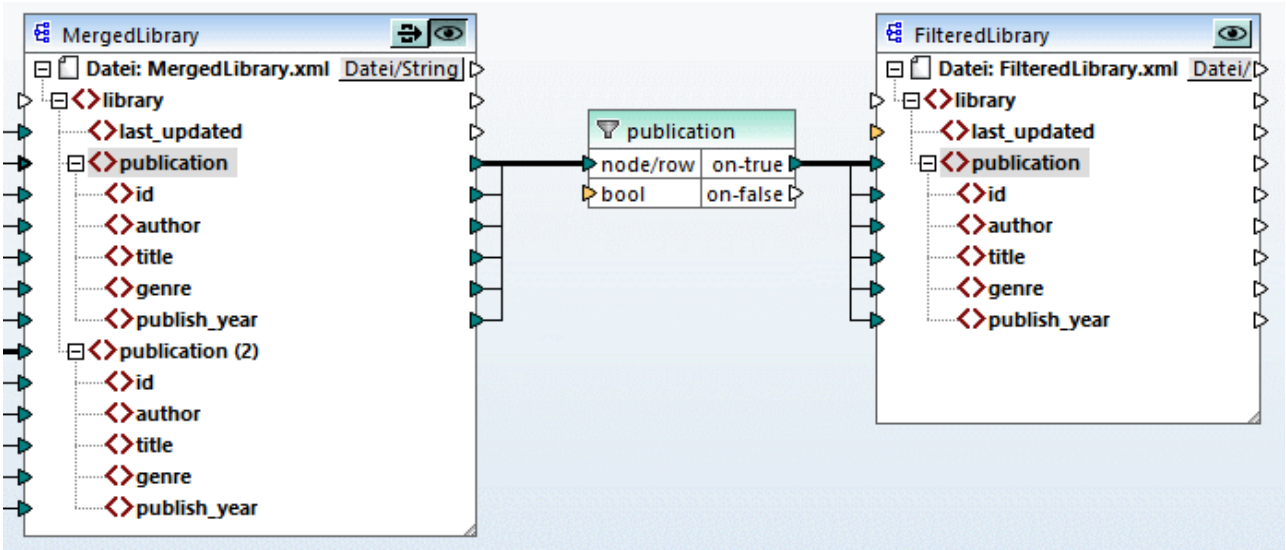
Hinzufügen eines Filters

Um einen Filter hinzuzufügen, klicken Sie mit der rechten Maustaste auf die Verbindung zwischen `MergedLibrary` und `FilteredLibrary` und wählen Sie im Kontextmenü den Befehl **Filter einfügen: Nodes/Zeilen** (*Abbildung unten*).

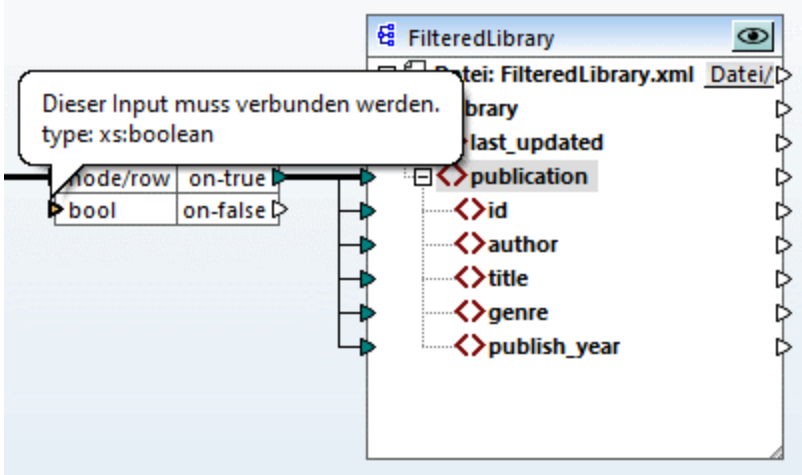
 © 2019-2025 Altova GmbH



Daraufhin wird die Filterkomponente zum Mapping hinzugefügt (*Abbildung unten*).



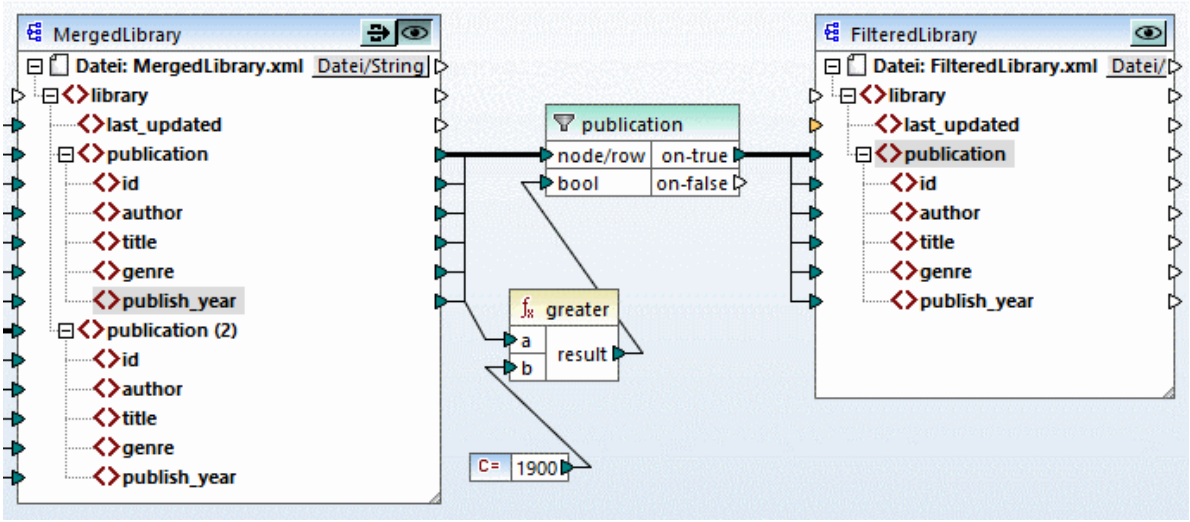
Wie in der Abbildung oben gezeigt, erscheint der Input-Konnektor `bool` orange markiert, was bedeutet, dass dieser Input zwingend erforderlich ist. Wenn Sie die Maus über den Konnektor platzieren, sehen Sie, dass ein Input vom Typ `xs:boolean` erforderlich ist (*siehe Abbildung unten*). Damit Tipps angezeigt werden, klicken Sie in der Symbolleiste auf (**Tipps anzeigen**).



Nur Bücher nach 1900

Für die Filterkomponente wird eine Bedingung benötigt, die `true` oder `false` zurückgibt. Wenn die Boolesche Bedingung `true` zurückgibt, werden die Daten der aktuellen `publication`-Sequenz in die Zielkomponente kopiert. Wenn die Bedingung `false` zurückgibt, werden die Daten nicht kopiert. In diesem Tutorial muss die Bedingung lauten, dass nur die nach 1900 veröffentlichten Bücher gemappt werden sollen. Gehen Sie folgendermaßen vor, um die Bedingung zu erstellen:

1. Klicken Sie in der Symbolleiste auf **Konstante** und geben Sie in die Textleiste *1900* ein. Wählen Sie als Typ *Zahl* aus.
2. Fügen Sie die Funktion **`greater`** zum Mapping hinzu.
3. Erstellen Sie die Mapping-Verbindungen zu und von der Funktion **`greater`**, wie unten gezeigt. Die Funktion **`greater`** vergleicht den Wert des Elements `publish_year` von jedem Werk mit dem Wert der Konstante. Nur diejenigen publication-Datensätze, deren Veröffentlichungsjahr größer als 1900 ist, werden auf die Zielkomponente gemappt.



 © 2019-2025 Altova GmbH

3.3.5 Anzeige einer Vorschau und Speichern der Ausgabe

Sie können nun eine Vorschau der Ausgaben der einzelnen Zielkomponenten anzeigen und speichern. Wenn im selben Mapping mehrere Zielkomponenten vorhanden sind, hat jede Zielkomponente eine Schaltfläche (**Vorschau**). Die Vorschau kann immer nur für jeweils eine Komponente aktiviert werden. Mit Hilfe der **Vorschau**-Schaltflächen können Sie das Mapping-Zwischenergebnis (in unserem Beispiel die auf die Komponente `MergedLibrary` gemappten Daten) sowie das Endergebnis des verketteten Mappings (die auf die Komponente `FilteredLibrary` gemappten Daten) anzeigen. Die Komponente `MergedLibrary` hat auch eine Schaltfläche (**Weiterleitung**). Mit Hilfe der **Weiterleitungs**-Schaltfläche können Sie festlegen, wie die Ausgabe generiert werden soll (*nähere Informationen weiter unten*).

Vorschau und Speichern der Zwischen- und Endausgabe
Wenn Sie sowohl die Ausgabe der Komponente `MergedLibrary` als auch die der Komponente `FilteredLibrary` anzeigen und speichern wollen, gehen Sie folgendermaßen vor:

1. Klicken Sie in der Komponente `MergedLibrary` auf die Schaltfläche **Weiterleitung**.
2. Stellen Sie sicher, dass die **Vorschau**-Schaltfläche der Komponente `FilteredLibrary` ebenfalls aktiviert ist.
3. Öffnen Sie das Ausgabefenster. Mit Hilfe der Schaltflächen **Zurück** und **Vorwärts** können Sie zwischen den Ausgaben wechseln.
4. Wechseln Sie zu der Ausgabe, die Sie in einer Datei speichern möchten und klicken Sie in der Symbolleiste auf **Ausgabe speichern**. Wenn Sie beide Ausgaben speichern möchten, klicken Sie in der Symbolleiste auf **Alle generierten Ausgaben speichern**.

Wenn die Weiterleitungsfunktion aktiv ist, so wird das Feld *XML-Input-Datei* der Zwischenkomponente automatisch deaktiviert. Der Grund dafür ist, dass die bei Anzeige des ersten Abschnitts des Mappings zwischen den Quellkomponenten (`Books` und `Library`) und der ersten Zielkomponente (`MergedLibrary`) generierte Ausgabe standardmäßig als Input verwendet wird, wenn Sie eine Vorschau auf den zweiten Abschnitt des Mappings zwischen der ersten (`MergedLibrary`) und der zweiten Zielkomponente (`FilteredLibrary`) anzeigen.

Anzeige einer Vorschau und Speichern der Zwischenausgabe
Zwischenkomponenten, bei denen die Schaltfläche "Weiterleitung" aktiv ist, können nicht in der Vorschau angezeigt werden. Die Vorschau-Schaltfläche der Zwischenkomponente ist automatisch deaktiviert, da eine Vorschau bei gleichzeitiger Datenweiterleitung nicht sinnvoll ist. Wenn Sie nur die Ausgabe der Zwischenkomponente (`MergedLibrary`) anzeigen und speichern wollen, gehen Sie folgendermaßen vor:

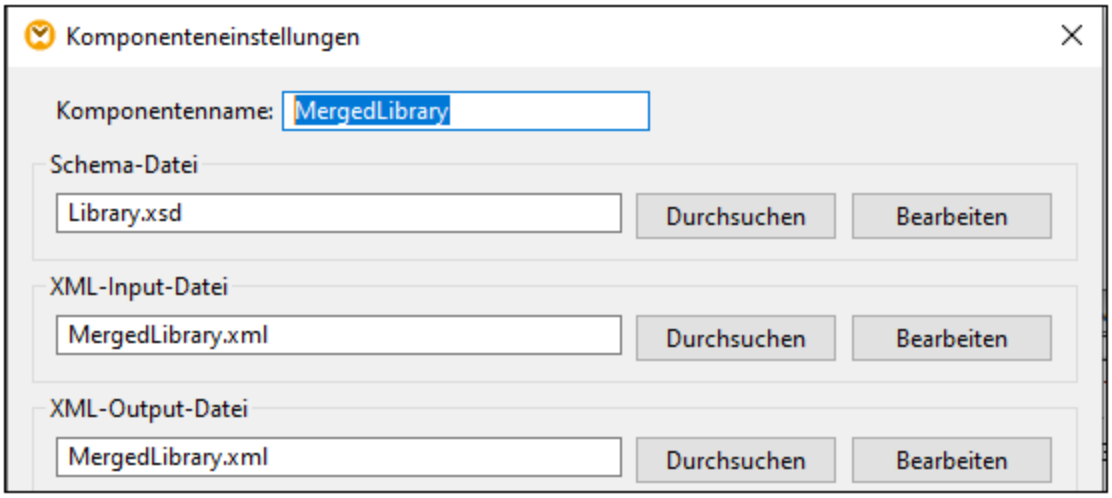
1. Deaktivieren Sie die **Weiterleitungs**-Schaltfläche der Komponente `MergedLibrary` , wenn die Schaltfläche zuvor aktiviert war.
2. Klicken Sie auf die **Vorschau**-Schaltfläche der Komponente `MergedLibrary`.
3. Öffnen Sie das Ausgabefenster.
4. Klicken Sie in der Symbolleiste auf die Schaltfläche **Ausgabedatei speichern**, um die Ausgabe in einer Datei zu speichern.

Anzeige einer Vorschau und Speichern der endgültigen Ausgabe
Wenn Sie nur die von der Zwischenkomponente auf die zweite Zielkomponente gemappten Daten anzeigen und speichern möchten, gehen Sie vor, wie unten beschrieben.

1. Deaktivieren Sie die **Weiterleitungs**-Schaltfläche der Komponente `MergedLibrary` , wenn die Schaltfläche zuvor aktiviert war.
2. Doppelklicken Sie auf die Überschrift der Komponente `MergedLibrary`.

3. Stellen Sie sicher, dass Sie im Feld *XML-Input-Datei* denselben Dateinamen wie im Feld *XML-Output-Datei* angeben (*Abbildung unten*). Wenn Sie die **Weiterleitungs**-Schaltfläche deaktivieren, können Sie auswählen, welche Input-Datei von der Zwischenkomponente gelesen werden soll. In den meisten Fällen sollte es sich hierbei um dieselbe Datei handeln, wie im Feld *XML-Output-Datei* definiert. Daher ist es normalerweise sinnvoll, dass die Zwischenkomponente eine einzige Datei für die Verarbeitung erhält und dieselbe Datei an das darauf folgende Mapping weiterleitet, anstatt nach einem anderen Dateinamen zu suchen.

   Es ist wichtig, dass die Zwischenkomponente dieselbe Input- und Output-Datei hat, wenn die **Weiterleitungs**-Schaltfläche der Zwischenkomponente deaktiviert ist. Damit stellen Sie sicher, dass die bei Anzeige des ersten Abschnitts des Mappings zwischen den Quellkomponenten (`Books` und `Library`) und der ersten Zielkomponente (`MergedLibrary`) generierte Ausgabe als Input verwendet wird, wenn Sie eine Vorschau auf den zweiten Abschnitt des Mappings zwischen der ersten (`MergedLibrary`) und der zweiten Zielkomponente (`FilteredLibrary`) anzeigen. Wenn Sie Ihr Mapping mit MapForce Server (*Professional und Enterprise Edition*) oder über generierten Code ausführen, gewährleisten identische Namen der Input- und Output.Datei der Zwischenkomponente, dass die Mapping-Kette nicht unterbrochen wird. Beachten Sie, dass die Verwendung *unterschiedlicher* Dateinamen für die Input- und Output-Datei in der Zwischenkomponente (bei deaktivierter **Weiterleitungsschaltfläche**) in MapForce, im generierten Code oder bei der MapForce Server-Ausführung zu Fehlern führen kann.



4. Klicken Sie in der Komponente `FilteredLibrary` auf die Schaltfläche **Vorschau**.
5. Öffnen Sie das Ausgabefenster.
6. Klicken Sie in der Symbolleiste auf die Schaltfläche **Ausgabedatei speichern**, um die Ausgabe in einer Datei zu speichern.

Achtung

- Die Funktion "Weiterleitung" steht nur für dateibasierte Komponenten wie XML-, CSV- und Textdateien zur Verfügung. Datenbankkomponenten (*Professional und Enterprise Editions*) können als Zwischenkomponenten verwendet werden, doch wird die Schaltfläche "Weiterleitung" nicht angezeigt.
- Wenn das Mapping mit MapForce ausgeführt wird, gibt die Einstellung *Direkt in die endgültigen Output-Dateien schreiben* (aus dem Menü **Extras | Optionen | Allgemein** 508) an, ob die Zwischendateien als temporäre oder als physischen Dateien gespeichert werden sollen. Beachten

 © 2019-2025 Altova GmbH

Sie, dass diese Einstellung nur dann gilt, wenn die Vorschau direkt in MapForce erfolgt. Wenn dieses Mapping mit MapForce Server oder durch generierten Code ausgeführt wird, werden in jeder Phase der Mapping-Kette tatsächlich Dateien erzeugt.

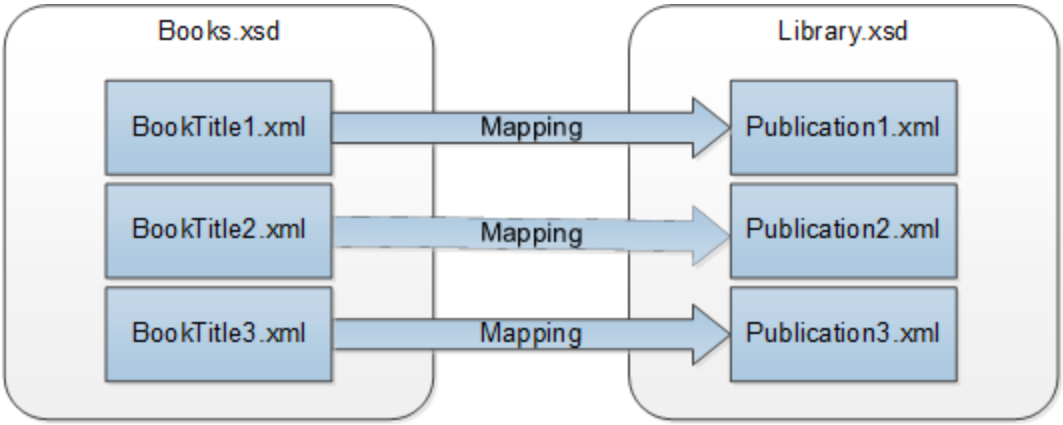
Sie haben nun ein verkettetes Mapping erstellt, das zwei Ausgabedateien erzeugt. Das Mapping-Design aus diesem Tutorial wurde unter dem Namen **`Tut3_ChainedMapping.mfd`** gespeichert.

3.4 Mehrere Quellkomponenten auf mehrere Zielkomponenten

In diesem Tutorial wird gezeigt, wie Sie mit derselben Transformation Daten aus mehreren Quelldateien auf mehrere Zieldateien mappen. Um dies zu veranschaulichen, werden wir nun ein Mapping erstellen, mit dem Folgendes bewerkstelligt werden soll:

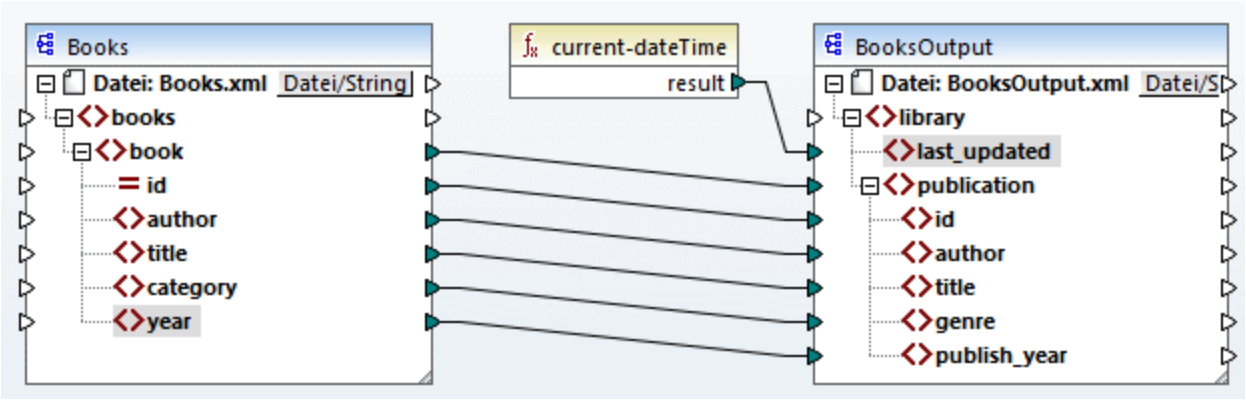
1. Auslesen der Daten aus mehreren XML-Dateien im selben Verzeichnis. Den Dateien liegt dasselbe Quellschema zugrunde.
2. Generieren einer neuen XML-Zieldatei für jede XML-Quelldatei. Die Zieldateien basieren auf dem neuen Zielschema.

In der Abbildung unten sehen Sie ein abstraktes Modell der in diesem Tutorial verwendeten Datentransformation:



Grundkonzept

Die Ausgangsbasis für dieses Tutorial bildet das Mapping **`Tut1_OneToOne.mfd`** aus dem ersten Tutorial 97 (*Abbildung unten*).



 © 2019-2025 Altova GmbH

Ändern der Quellkomponente
Wir ändern die Komponenteneinstellungen der Quellkomponente, sodass sie Daten aus mehreren Quelldateien ausliest: **BookTitle1.xml**, **BookTitle2.xml** und **BookTitle3.xml**. Jede dieser Dateien basiert auf **Books.xsd** und es ist darin ein einziger Buchdatensatz gespeichert (*siehe unten*).

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>
```

Ändern der Zielkomponente
Außerdem konfigurieren wir die Zielkomponente so, dass die Daten in mehrere Zieldateien geschrieben werden. Die Zieldateien basieren auf demselben Schema **Library.xsd**. Die generierten Zieldateien erhalten jeweils den Namen **Publication1.xml**, **Publication2.xml** und **Publication3.xml** (*Codefragmente unten*).

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>
```

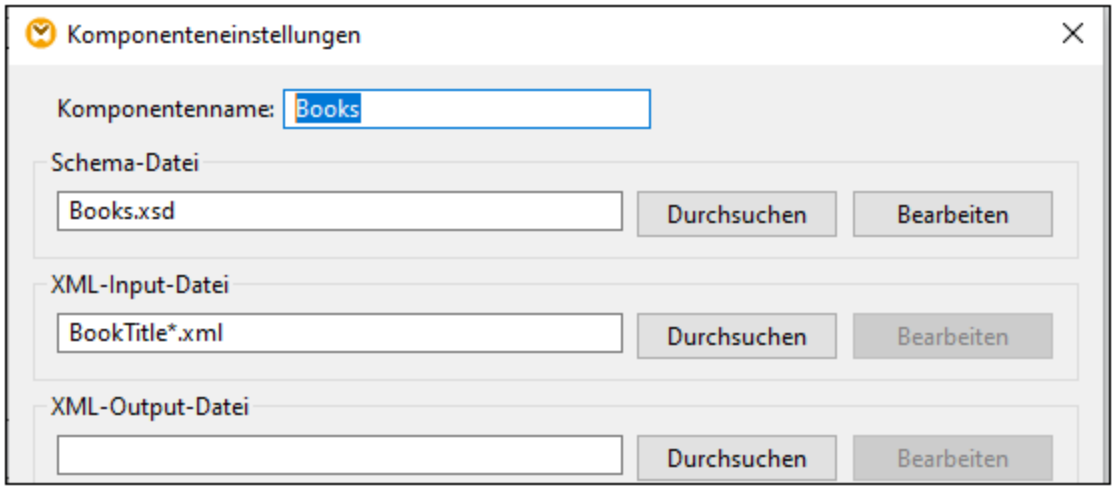
Um die erforderliche Datentransformation durchzuführen, gehen Sie vor, wie in den nachfolgenden Unterabschnitten beschrieben.

3.4.1 Konfigurieren des Input

Im ersten Schritt werden die Komponenteneinstellungen der Quellkomponente geändert. Bevor Sie Änderungen an den Komponenteneinstellungen vornehmen, sollten Sie das Mapping unter einem neuen Namen im Ordner **`Basic Tutorials`** speichern.

Damit in MapForce mehrere XML-Instanzdateien verarbeitet werden, doppelklicken Sie auf die Überschrift der Quellkomponente und geben Sie **`BookTitle*.xml`** in das Feld *XML-Input-Datei* ein (*Abbildung unten*). Aufgrund des Sternchens (*) im Dateinamen verwendet MapForce alle Dateien mit dem Präfix `BookTitle` als Mapping-Input. Da es sich um einen relativen Pfad handelt, sucht MapForce nach allen `BookTitle`-Dateien im selben Verzeichnis wie die Mapping-Datei. Sie können gegebenenfalls auch einen absoluten Pfad eingeben.

 © 2019-2025 Altova GmbH

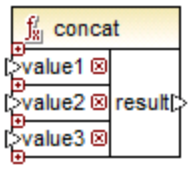


3.4.2 Konfigurieren des Ausgabeteils 1

In nächsten Schritt werden nun die Dateinamen für die einzelnen Ausgabedateien erstellt. Dazu verwenden wir die **concat** 329-Funktion, die alle bereitgestellten Werte miteinander verbindet. Wenn diese Werte miteinander verbunden werden, wird daraus ein Ausgabedateiname gebildet (z.B. **Publication1.xml**). Um die Dateinamen mit Hilfe der **concat**-Funktion zu generieren, gehen Sie folgendermaßen vor:

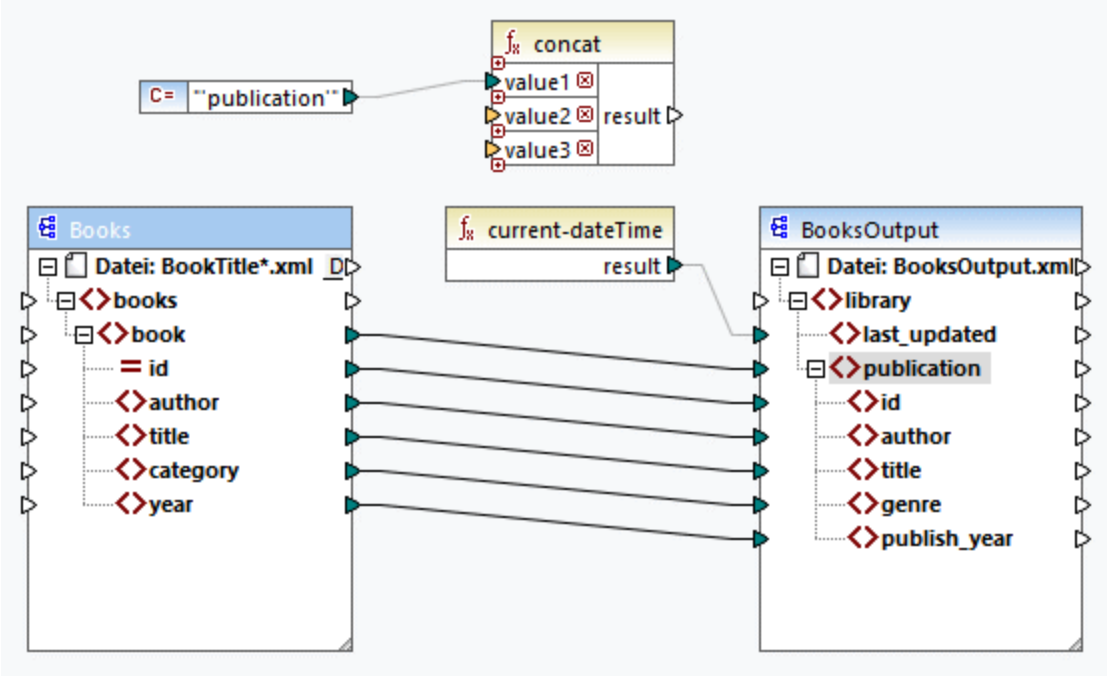
Schritt 1: Hinzufügen der concat-Funktion

Fügen 102 Sie die **concat**-Funktion (*Abbildung unten*) zum Mapping-Bereich hinzu. Diese Funktion hat standardmäßig zwei Parameter, wenn Sie zum Mapping hinzugefügt wird. Wir benötigen in unserem Beispiel drei Parameter. Klicken Sie in der Funktionskomponente auf ⊡ (**Parameter hinzufügen**) und fügen Sie einen dritten Parameter hinzu. Beachten Sie, dass Sie einen Parameter durch Klicken auf ⊠ (**Parameter löschen**) löschen können.



Schritt 2: Einfügen einer Konstante

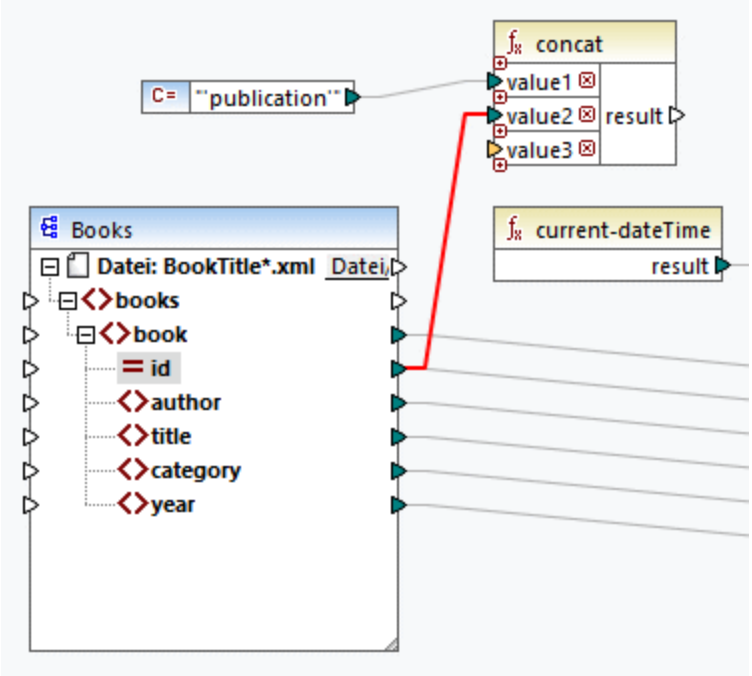
Im nächsten Schritt wird nun eine Konstante hinzugefügt. Wenn Sie aufgefordert werden, einen Wert anzugeben, geben Sie `publication` ein und belassen Sie die Option *String* aktiviert. Verbinden Sie die Konstante mit `value1` der **concat**-Funktion, wie in der Abbildung unten gezeigt:



Schritt 3: Angabe von IDs

Verbinden Sie das Attribut `id` der Quellkomponente mit `value2` der **`concat`**-Funktion (*Abbildung unten*). Das Attribut `id` der XML-Quelldatei liefert den eindeutigen ID-Wert für die einzelnen Dateien. Dadurch wird verhindert, dass die Dateien mit demselben Namen generiert werden. Die Verbindung wird rot angezeigt, wenn Sie darauf klicken.

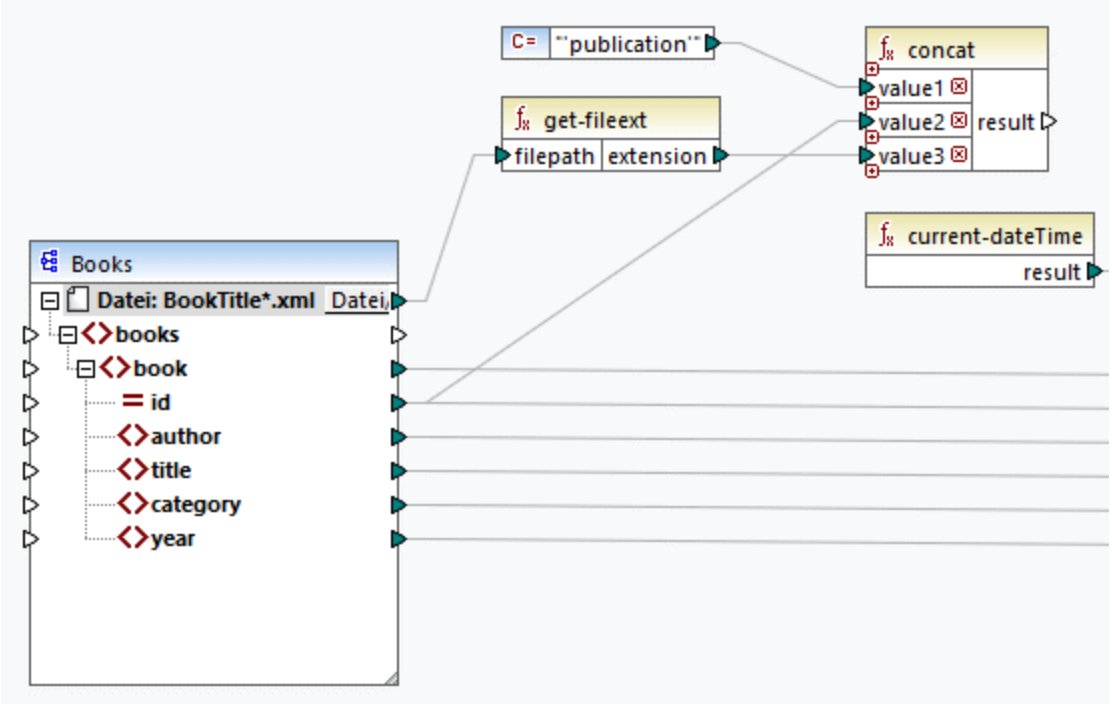
 © 2019-2025 Altova GmbH



Schritt 4: Extraktion der Dateierweiterung

Fügen Sie die Funktion **get-fileext** 271 zum Mapping-Bereich hinzu. Erstellen Sie anschließend eine Verbindung vom obersten Node der Quellkomponente (**Datei: BookTitle*.xml**) zum Parameter `filepath` dieser Funktion (*Abbildung unten*).

Verbinden Sie anschließend den Parameter `extension` der **get-fileext**-Funktion mit `value3` der **concat**-Funktion. Auf diese Art extrahieren Sie nur die Dateierweiterung (in diesem Fall **.xml**) aus dem Namen der Quelldatei und übergeben sie an den Namen der Ausgabedatei.



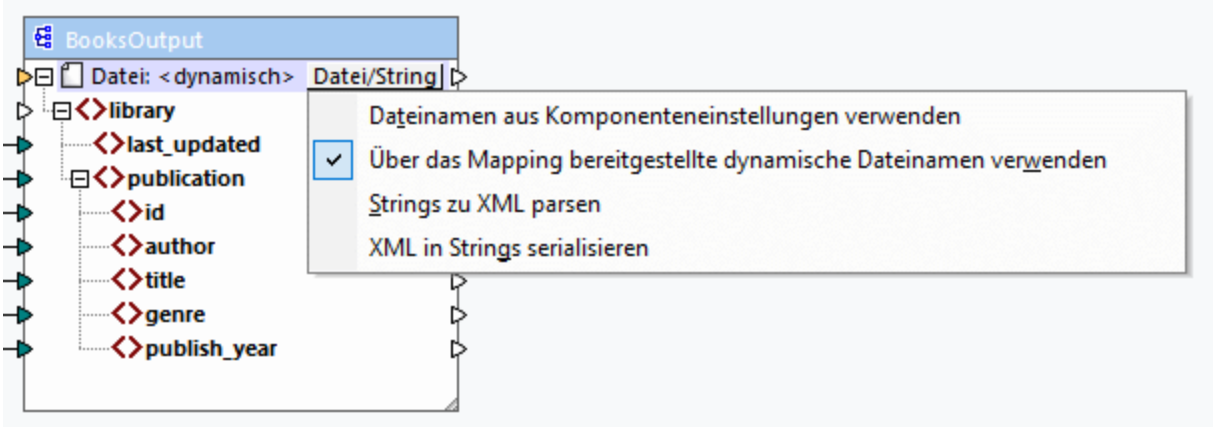
3.4.3 Konfigurieren des Ausgabeteils 2

Im zweiten Teil der Ausgabekonfiguration wollen wir nun, dass MapForce die Ausgabedateien dynamisch generiert, d.h. jede Ausgabedatei erhält ihren Namen auf Basis der durch die **`concat`**-Funktion bereitgestellten Argumente. Dazu verwenden wir dynamische Dateinamen (*siehe Unterabschnitte weiter unten*). Nähere Informationen zu dynamischen Dateinamen finden Sie unter Dynamische Verarbeitung mehrerer Input- und Output-Dateien 422.

Schritt 1: Definieren dynamischer Dateinamen

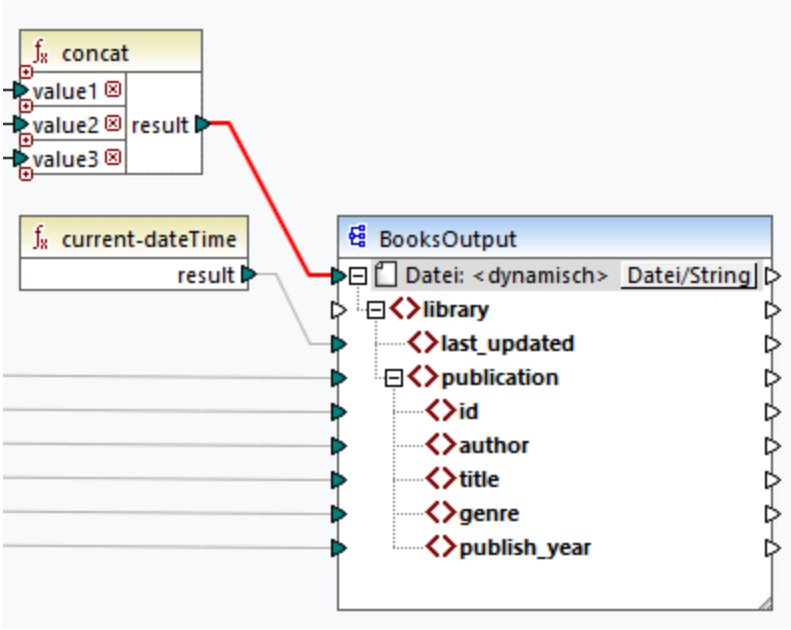
Damit MapForce die Instanzdateien dynamisch generiert, klicken Sie in der Zielkomponente auf die Schaltfläche Datei bzw. Datei/String neben dem Node `Datei` und wählen Sie im Kontextmenü den Befehl **Über das Mapping bereitgestellte dynamische Dateinamen verwenden** (*Abbildung unten*).

 © 2019-2025 Altova GmbH



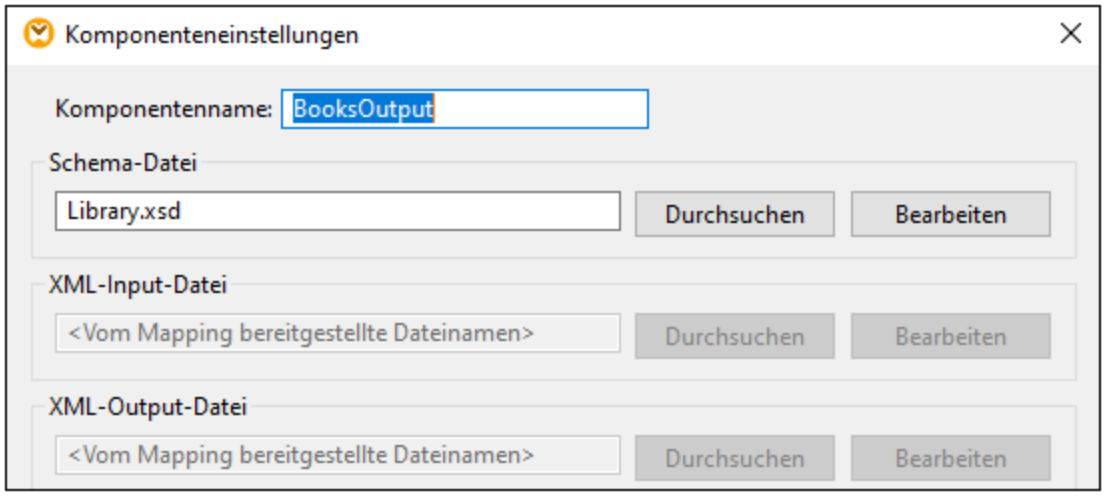
Schritt 2: Verbinden der concat-Funktion mit dem dynamischen Node

Im nächsten Schritt wird das Ergebnis der **`concat`**-Funktion mit dem Node `Datei: <dynamisch>` der Zielkomponente verbunden (*Abbildung unten*).



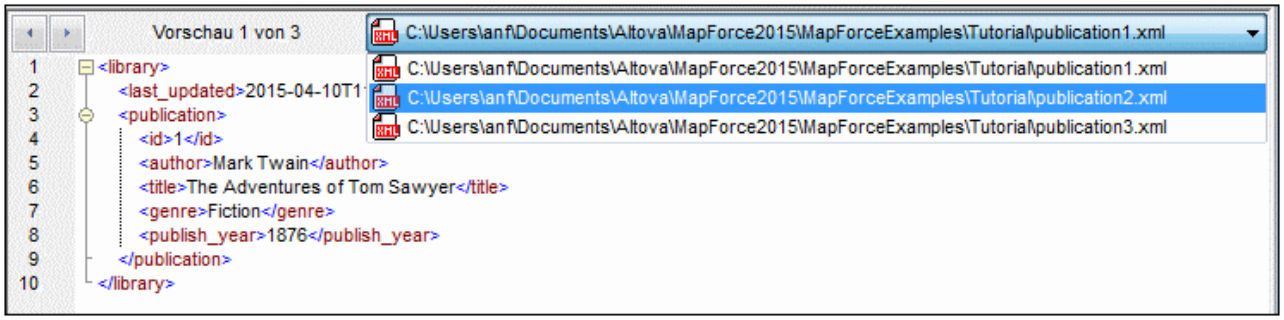
Schritt 3: Überprüfen der Komponenteneinstellungen

In den Komponenteneinstellungen sehen Sie, dass die Textfelder *XML-Input-Datei* und *XML-Output-Datei* deaktiviert sind und darin angezeigt wird <*Vom Mapping bereitgestellte Dateinamen*> (*Abbildung unten*). Dies gibt an, dass die Instanzdateinamen dynamisch über ein Mapping bereitgestellt werden, daher können diese nicht mehr in den Komponenteneinstellungen definiert werden.



Schritt 4: Generieren von Ausgabedateien

Sie können das Mapping nun ausführen und das Ergebnis sowie die Namen der generierten Dateien sehen. Um durch die Ausgabedateien zu navigieren, verwenden Sie die Pfeilschaltflächen in der linken oberen Ecke des Ausgabefensters oder wählen Sie eine Datei aus der daneben liegenden Dropdown-Liste aus (*Abbildung unten*).



Das Mapping-Design aus diesem Tutorial wurde unter dem Namen **`Tut4_MultipleToMultiple.mfd`** gespeichert.

 © 2019-2025 Altova GmbH

4 Strukturkomponenten

Dieser Abschnitt enthält Informationen über verschiedene Datenformate, die als Quell- und Zielkomponenten verwendet werden können.

- XML und XML-Schema 130

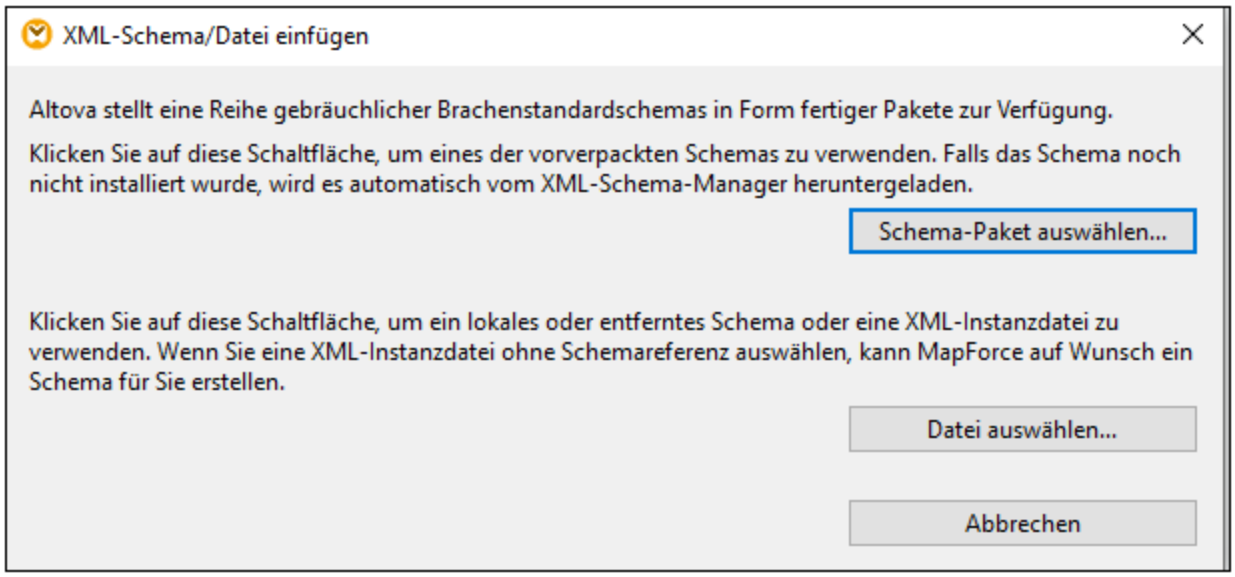
4.1 XML und XML-Schema

Altova Website: XML-Mapping

XML ist eine Markup-Sprache für Textdokumente. In XML Schema sind die Struktur und die Einschränkungen von XML-Dokumenten definiert. Bei XML-Komponenten in MapForce handelt es sich um Strukturkomponenten[31], die als Datenquellen und -ziele verwendet werden können. Informationen zu grundlegenden Datentransformationsszenarien finden Sie in den Tutorials[96].

Einfügen eines XML-Schemas/einer XML-Datei

Mit dem Menübefehl **Einfügen | XML-Schema/Datei** oder über die Symbolleisten-Schaltfläche können Sie ein XML-Schema/eine XML-Datei einfügen. Im Dialogfeld (*siehe Abbildung unten*) werden Sie gefragt, ob Sie ein vorverpacktes Branchenstandardschema oder ein lokales bzw. eine entfernte Schema/Instanzdatei einfügen möchten. Wenn Sie sich für ein vorverpacktes Schema entscheiden, werden Sie aufgefordert, einen Eintrittspunkt auszuwählen. Wenn das gewünschte Schema noch nicht installiert ist, wird der XML-Schema-Manager[147] aufgerufen, über den Sie es herunterladen können.



Generieren eines XML-Schemas
Wenn Sie eine lokale oder entfernte XML-Datei ohne eine Schemareferenz hinzufügen, wird Ihnen von MapForce vorgeschlagen, ein XML-Schema zu generieren. Anschließend werden Sie gefragt, in welchem Verzeichnis das generierte Schema gespeichert werden soll.

Bei Generierung eines Schemas anhand einer XML-Datei müssen die Datentypen für Elemente/Attribute anhand des XML-Instanzdokuments abgeleitet werden und entsprechen eventuell nicht genau Ihren Erwartungen. Überprüfen Sie bitte, ob das generierte Schema tatsächlich die Instanzdaten genau darstellt.

Wenn Elemente oder Attribute in mehr als einem Namespace vorhanden sind, generiert MapForce für jeden Namespace ein separates XML-Schema, es können daher mehrere Dateien auf der Festplatte generiert werden.

 © 2019-2025 Altova GmbH

DTD als Dokumentstruktur
Versionen ab MapForce 2006 SP2 unterstützen Namespace-fähige DTDs für Quell- und Zielkomponenten. Die Namespace URIs werden aus den DTD `xmlns`-Attributdeklarationen extrahiert, um Mappings zu ermöglichen. Einige DTDs enthalten allerdings `xmlns*`-Attributdeklarationen ohne Namespace-URIs (z.B. von StyleVision verwendete DTDs). Um solche DTDs in MapForce verwenden zu können, definieren Sie das `xmlns`-Attribut mit der Namespace-URI folgendermaßen:

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

Anmerkung zu Enumerationswerten

Bei Nodes, deren Datentypen Enumeration Facets haben, können Sie eine Wertezuordnung erstellen, bei der alle Enumerationswerte im Vorhinein ausgefüllt werden. Dank dieser Funktionalität lassen sich Enumerationswerte schneller und leichter verarbeiten und mappen. Nähere Informationen dazu finden Sie unter Wertezuordnungen 204.

In diesem Abschnitt

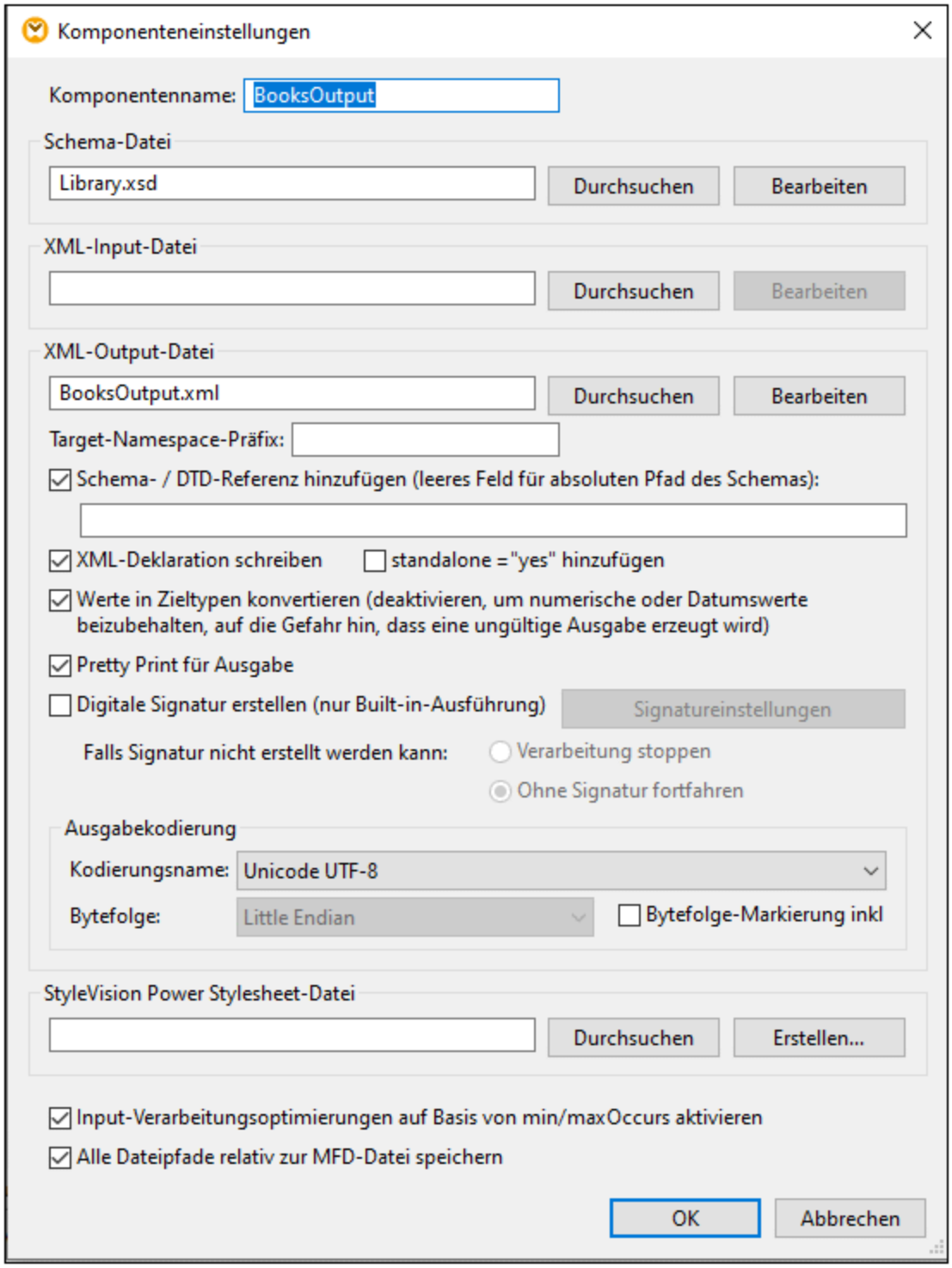
Dieser Abschnitt ist in die folgenden Kapitel gegliedert:

- XML-Komponenteneinstellungen 131
- Abgeleitete Typen 136
- NULL-Werte 138
- Kommentare und Processing Instructions 140
- CDATA-Abschnitte 140
- Wildcards: xs:any/xs:anyAttribute 142
- Benutzerdefinierte Namespaces 144
- XML-Schema-Manager 147

4.1.1 XML-Komponenteneinstellungen

Nachdem Sie eine XML-Komponente zum Mapping-Bereich hinzugefügt haben, können Sie die Einstellungen dafür über das Dialogfeld **Komponenteneinstellungen** (*siehe Abbildung unten*) konfigurieren. Sie können das Dialogfeld **Komponenteneinstellungen** auf eine der folgenden Arten öffnen:

- durch Doppelklick auf den Komponententitel
- durch Klick mit der rechten Maustaste auf den Komponententitel und Auswahl von **Eigenschaften**
- durch Auswahl der Komponente im Mapping und Klick auf **Eigenschaften** im Menü **Komponente**



Weiter unten finden Sie eine Beschreibung der verfügbaren Einstellungen.

 © 2019-2025 Altova GmbH

Allgemeine Einstellungen

Komponentenname

Der Komponentenname wird bei der Erstellung der Komponente automatisch generiert. Sie können den Namen jedoch jederzeit ändern. Der Komponentenname kann Leerzeichen und Punkte enthalten, darf aber keine Schrägstriche, umgekehrten Schrägstriche, Doppelpunkte, doppelten Anführungszeichen und voran- oder nachgestellte Leerzeichen enthalten. Beachten Sie beim Ändern des Namens einer Komponente Folgendes:

- Wenn Sie das Mapping auf FlowForce Server bereitstellen möchten, muss der Komponentenname eindeutig sein.
- Es wird empfohlen, nur Zeichen zu verwenden, die über die Befehlszeile eingegeben werden können. Länderspezifische Sonderzeichen sind in Windows und der Befehlszeile eventuell unterschiedlich kodiert.

Schema-Datei

Definiert den Namen und Pfad der XML-Schema-Datei, die von MapForce zum Validieren und Mappen der Daten verwendet wird. Um den Pfad der Datei zu ändern, klicken Sie auf **Durchsuchen** und wählen Sie eine neue Datei aus. Um die Datei in Altova XMLSpy zu bearbeiten, klicken Sie auf **Bearbeiten**.

XML-Input-Datei

Definiert die XML-Instanzdatei, aus der MapForce die Daten ausliest. Dieses Feld hat bei einer Quellkomponente eine Bedeutung und wird ausgefüllt, wenn Sie die Komponente erstellen und dieser Komponente eine XML-Instanzdatei zuweisen. In einer Quellkomponente werden anhand des Namens der Instanzdatei das XML-Root-Element und das referenzierte Schema, anhand dessen die Datei validiert werden soll, ermittelt. Um den Pfad der Datei zu ändern, klicken Sie auf **Durchsuchen** und wählen Sie eine neue Datei aus. Um die Datei in Altova XMLSpy zu bearbeiten, klicken Sie auf **Bearbeiten**.

XML-Output-Datei

Definiert die XML-Instanzdatei, in die MapForce die Daten schreibt. Dieses Feld hat bei einer Zielkomponente eine Bedeutung. Um den Pfad der Datei zu ändern, klicken Sie auf **Durchsuchen** und wählen Sie eine neue Datei aus. Um die Datei in Altova XMLSpy zu bearbeiten, klicken Sie auf **Bearbeiten**.

Target-Namespace-Präfix

Hier können Sie ein Präfix für den Target Namespace eingeben. Stellen Sie sicher, dass der Target Namespace im Zielschema definiert ist, bevor Sie ein Präfix zuweisen.

Schema-/DTD-Referenz hinzufügen

Fügt den Pfad der referenzierten XML-Schema-Datei zum Root-Element der XML-Ausgabe hinzu. Der in dieses Feld eingegebene Schemapfad wird in das Attribut `xsi:schemaLocation` oder (bei Verwendung einer DTD) die `DOCTYPE`-Deklaration der generierten Instanz-Zieldateien geschrieben.

MapForce Professional und Enterprise Edition: Bei der Generierung von Code in XQuery oder C++ wird das Hinzufügen einer DTD-Referenz nicht unterstützt.

Wenn Sie einen Pfad in dieses Feld eingeben, können Sie definieren, wo sich die von der XML-Instanzdatei referenzierte Schemadatei befindet. Damit stellen Sie sicher, dass die Ausgabeinstanz im Mapping-Ordner bei Ausführung des Mappings validiert werden kann. Sie können in dieses Feld eine **`http://`**-Adresse sowie einen absoluten oder relativen Pfad eingeben.

Wenn Sie diese Option deaktivieren, können Sie die XML-Instanz vom referenzierten XML-Schema oder der DTD entkoppeln. Verwenden Sie diese Option z.B., wenn Sie die erzeugte XML-Ausgabedatei an jemanden senden möchten, der keinen Zugriff auf das zugrunde liegende XML-Schema hat.

XML-Deklaration schreiben

Standardmäßig ist die Option aktiviert, d.h. die XML-Deklaration wird in die Ausgabe geschrieben. Diese Funktionalität wird in den folgenden MapForce-Zielsprachen und Ausführungsprozessoren unterstützt.

Zielsprache / Ausführungsprozessor	**Wenn die Ausgabe eine Datei ist**	**Wenn die Ausgabe ein String ist**
Built-in (*Professional und Enterprise Edition*)	Ja	Ja
MapForce Server (*Professional und Enterprise Edition*)	Ja	Ja
XQuery (*Professional und Enterprise Edition*), XSLT	Ja	Nein
Code Generator (C++, C#, Java) (*Professional und Enterprise Edition*)	Ja	Ja

standalone ="yes" hinzufügen

Bei Auswahl dieser Option wird die Deklaration `standalone="yes"` in die XML-Deklaration Ihrer XML-Zieldatei eingefügt. Nähere Informationen dazu finden Sie unter *Standalone Document Declaration*.

Beachten Sie die folgenden Punkte:

- Ist die Option *standalone="yes"* aktiviert, ist die Generierung einer Ausgabe kompatibel mit XSLT 1-3, Built-In und generiertem Code (C#, Java, C++ MSXML, C+ Xerces). Die Transformationssprache Built-In 16 sowie die Unterstützung von Codegenerierung stehen in der Professional und Enterprise Edition zur Verfügung.
- In Datenbankfelder und Webservice Requests eingebetteter XML-Code wird nicht unterstützt (*Professional und Enterprise Edition*).

Werte in Zieltypen konvertieren

Damit können Sie festlegen, ob (i) im Mapping die XML-Zielschematypen verwendet werden sollen oder (ii) ob alle Daten, die auf die Zielkomponenten gemappt werden, als Stringwerte behandelt werden sollen. Standardmäßig ist diese Einstellung aktiviert. Wenn Sie diese Option deaktivieren, können Sie die exakte Formatierung der Werte beibehalten, z.B. eignet sich die Option, wenn ein Inhalt genau einem Pattern Facet in einem Schema entsprechen muss, demzufolge ein numerischer Wert eine bestimmte Anzahl an Dezimalstellen aufweisen muss. Sie können die Zahl mit Hilfe von Mapping-Funktionen als String im gewünschten Format formatieren und diesen String dann auf die Zielkomponente mappen.

 © 2019-2025 Altova GmbH

Wenn Sie diese Option deaktivieren, wird auch die Erkennung ungültiger Werte deaktiviert, z.B. das Schreiben von Buchstaben in numerische Felder.

- Pretty Print für Ausgabe

  Formatiert Ihr XML-Ausgabedokument neu, um eine strukturierte Anzeige des Dokuments zu generieren. Jedes Subelement ist einen Tabstopp vom übergeordneten Element eingerückt.

- Digitale Signatur erstellen (*Enterprise Edition*)

  Gestattet Ihnen, eine digitale Signatur zur XML-Ausgabeinstanzdatei hinzuzufügen. Digitale Signaturen können nur generiert werden, wenn als Transformationssprache Built-In gewählt wurde.

Ausgabekodierung

Damit können Sie die folgenden Einstellungen der Output-Instanzdatei definieren:

- Kodierungsname
- Bytefolge
- Ob das Bytefolge-Markierungszeichen (BOM) inkludiert werden soll.

Standardmäßig haben alle neuen Komponenten die in der Option *Standardkodierung für neue Komponenten* definierte Kodierung. Sie können diese Option über **Extras | Optionen** (Abschnitt *Allgemein*) aufrufen.

Wenn mit dem Mapping XSLT 1.0/2.0-Code generiert wird, hat die Aktivierung des Kontrollkästchens *Bytefolge-Markierung* keine Auswirkung**,** da diese Sprachen Bytefolge-Markierungen nicht unterstützen.

StyleVision Power Stylesheet-Datei

Über diese Option können Sie eine Altova StyleVision Stylesheet-Datei auswählen oder erstellen. Mit Hilfe einer solchen Daten können Sie Daten aus der XML-Instanzdatei in die verschiedensten Berichtsformate wie z.B. HTML, RTF und andere transformieren. Siehe auch Verwenden relativer Pfade in einer Komponente 40.

Andere Einstellungen

- Input-Verarbeitungsoptimierungen auf Basis von min/maxOccurs aktivieren

  Diese Option ermöglicht die Sonderbehandlung bei Sequenzen, von denen bekannt ist, dass sie nur genau ein Datenelement enthalten, z.B. erforderliche Attribute oder Child-Elemente mit `minOccurs` und `maxOccurs="1"`. In diesem Fall wird das erste Datenelement der Sequenz extrahiert, anschließend wird das Datenelement direkt als atomarer Wert (und nicht als Sequenz) verarbeitet.

  Wenn die Input-Daten gemäß dem Schema **nicht gültig** sind, könnte eine leere Sequenz in einem Mapping vorkommen, sodass das Mapping mit einer Fehlermeldung abgebrochen wird. Damit auch ein solcher **ungültiger Input** verarbeitet werden kann, deaktivieren Sie dieses Kontrolllkästchen.

- Alle Dateipfade relativ zur MFD-Datei speichern

  Wenn diese Option aktiviert ist, speichert MapForce die im Dialogfeld **Komponenteneinstellungen** angezeigten Dateipfade relativ zum Ordner, in dem sich die MapForce Design (**`.mfd`**)-Datei befindet. Siehe auch Relative und absolute Pfade 40.

4.1.2 Abgeleitete Typen

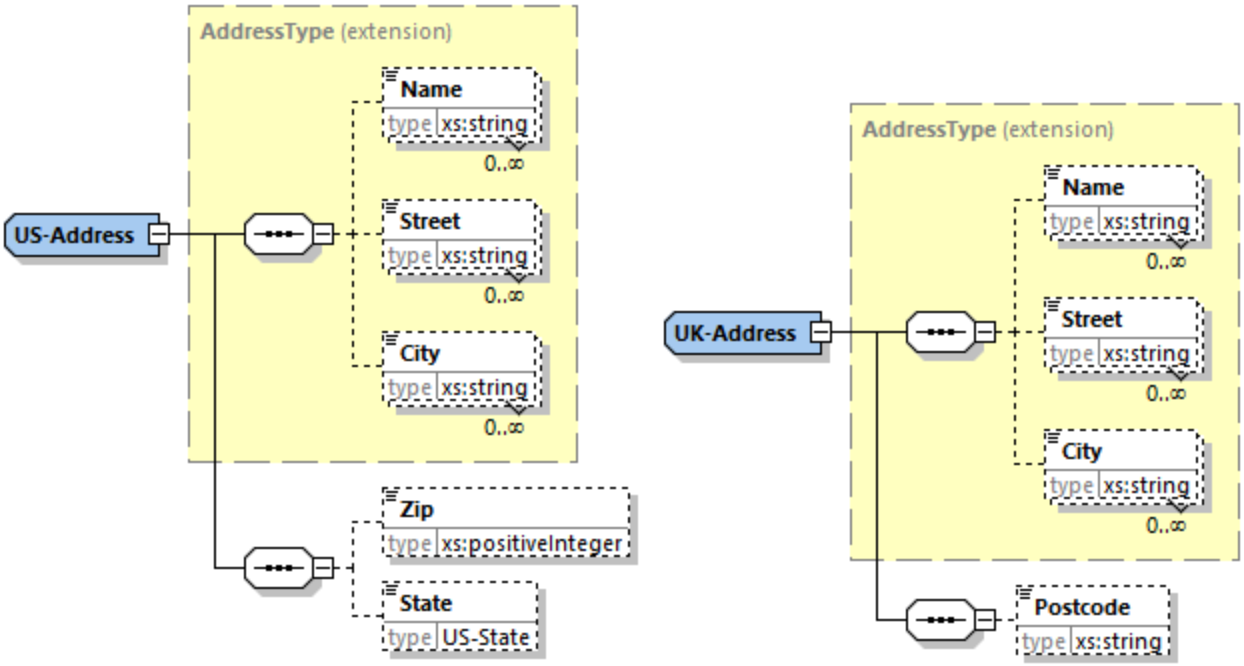
In diesem Kapitel wird beschrieben, wie Sie abgeleitete Typen in Mappings verwenden. Abgeleitete Typen sind in der W3C XML Schema-Spezifikation (Abschnitt 2.5.2) definiert. Eine kurze Übersicht über primitive und abgeleitete Typen finden Sie in der Microsoft-Dokumentation. Um abgeleitete Typen in einem Mapping verwenden zu können, müssen Sie das `xsi:type`-Attribut in Ihrer XML-Datei definieren (z.B., `<Address xsi:type="UK-Address">`).

Mögliches Szenario

In diesem Unterabschnitt wird ein mögliches Szenario für die Verwendung eines abgeleiteten Typs beschrieben. Angenommen, wir haben eine Firma mit zwei Niederlassungen, einer in GB und der anderen in den USA. Wir hätten nun gerne zwei Listen (`UKCustomers` und `USCustomers`). Jede davon soll Informationen über die Adresse der jeweiligen Niederlassung und alle Kunden im Zusammenhang mit dieser Niederlassung enthalten.

Definition von abgeleiteten Typen

In der Abbildung unten sehen Sie die Definition von abgeleiteten Typen namens `US-Address` und `UK-Address` (*XMLSpy Schema-Ansicht*). Die Elemente `UK-Address` und `UK-Address` haben denselben Basistyp namens `AddressType`, der die Elemente `Name`, `Street` und `City` enthält. Im Element `US-Address` wurde der Basistyp erweitert und enthält `Zip` und `State`, während das Element `UK-Address` den Basistyp und das Element `Postcode` enthält. Wir werden nun zur Veranschaulichung nur das Element `UK-Address` auf die Zieldatei mappen.

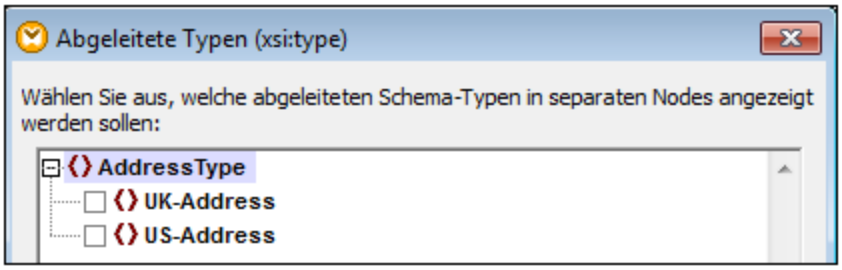


 © 2019-2025 Altova GmbH

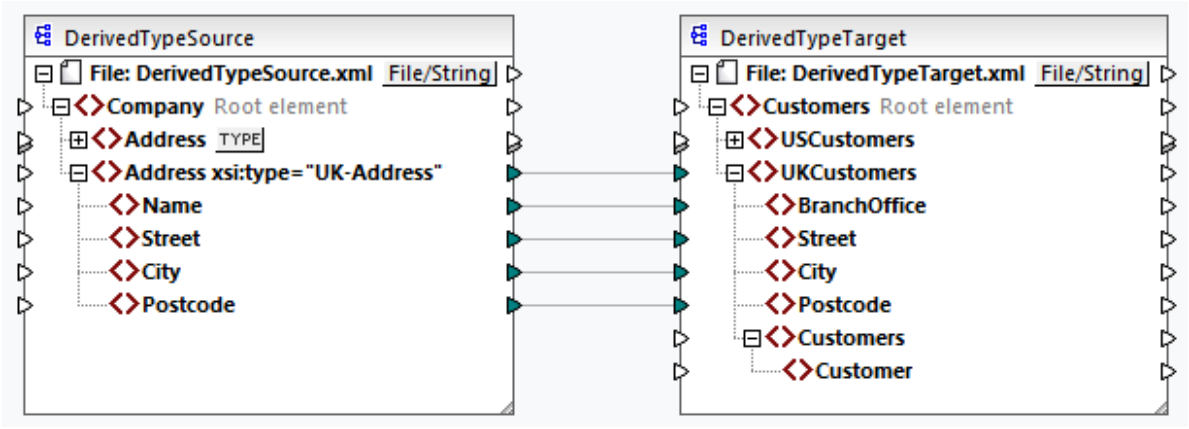
Abgeleiteter Typ in einem Mapping

In der Anleitung unten wird beschrieben, wie Sie Daten vom abgeleiteten Typ aus mappen. Ziel ist es, Informationen über das UK-Büro auf das Element `UKCustomers` zu mappen. Die Beispieldateien stehen im Ordner **`Tutorial`** zur Verfügung.

1. Klicken Sie im Menü **Einfügen** auf **XML-Schema/Datei**, und öffnen Sie die Datei **`DerivedTypeSource.xml`**. Diese XML-Datei basiert auf **`DerivedTypeSource.xsd`**.
2. Fügen Sie die Zieldatei **`DerivedTypeTarget.xsd`** ein. Beachten Sie, dass das Zielschema das Attribut `xsi:type` nicht enthalten muss.
3. Klicken Sie in der Quellkomponente neben dem Element `Address` auf die Schaltfläche TYPE. Diese Schaltfläche gibt an, dass es für dieses Element abgeleitete Typen im Schema gibt.
4. Über das Dialogfeld **Abgeleitete Typen** (*siehe Abbildung unten*) können Sie einen beliebigen abgeleiteten Typ für dieses bestimmte Element auswählen. In unserem Beispielmapping soll nur `UK-Address` gemappt werden.



5. Sobald Sie das Kontrollkästchen neben dem abgeleiteten `UK-Address`-Typ auswählen, steht in der Komponente ein neues Element namens `Address xsi:type="UK-Address"` zur Verfügung.
6. Verbinden Sie nun die Nodes, wie im Mapping unten gezeigt.



Ausgabe

Wenn Sie auf das **Ausgabe**-Fenster klicken, sehen Sie das folgende Resultat:

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

Das Beispielmapping ist unter dem Namen **`Tutorial\DerivedType.mfd`** gespeichert. Sie können auch eine weitere XML-Quellkomponente hinzufügen, die Informationen über die Kunden in UK enthält, und diese Daten auf den Node `Customers` in der Zielkomponente mappen. Dadurch enthält das Element `UKCustomers` in der Folge Informationen über die UK-Adresse und alle mit dieser Niederlassung verknüpften Kunden.

4.1.3 NULL-Werte

In diesem Abschnitt wird erläutert, wie MapForce NULL-Werte in Quell- und Zielkomponenten behandelt. Um das Attribut `xsi:nil="true"` in Ihrer XML-Datei verwenden zu können, müssen Sie in Ihrer Schema-Datei für das/die relevante(n) Element(e) das Attribut `nillable="true"` definieren. Nähere Informationen über die Attribute `nillable` und `xsi:nil` finden Sie in der W3C-Spezifikation. Beachten Sie, dass das Attribut `xsi:nil` im **Mapping**-Fenster in der Struktur einer Komponente nicht sichtbar ist.

In den folgenden Unterabschnitten werden einige mögliche Szenarien beim Mappen von NULL-Werten beschrieben.

NULL-Werte in XML-Komponenten

In diesem Unterabschnitt werden einige mögliche Szenarien beim Mappen von Elementen mit einem `xsi:nil="true"`-Attribut beschrieben.

Nur das Quellelement hat xsi:nil="true"/Sowohl Quell- als auch Zielelementen haben xsi:nil="true"
Für dieses Szenario gelten die folgenden Bedingungen:

- Die Verbindung ist zielorientiert[49].
- Das Quellelement hat ein `xsi:nil="true"`-Attribut. Das dazugehörige Zielelement hat dieses Attribut nicht.
- Alternativ dazu können sowohl Quell- als auch Zielelemente `xsi:nil="true"`-Attribute haben.
- Die `nillable="true"`-Attribute müssen im Quell- und im Zielschema definiert sein.
- Quell- und Zielelement haben den Typ simpleType.

In diesem Fall erhält das Zielelement in der Ausgabedatei das Attribut `xsi:nil="true"` , wie in der Beispiel-Ausgabedatei unten gezeigt (*gelb markiert*).

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

Anmerkung: Wenn das Attribut `nillable="true"` im Zielschema nicht definiert ist, bleibt das entsprechende Zielelement in der Ausgabe leer.

Nur das Zielelement hat xsi:nil="true"
Für dieses Szenario gelten die folgenden Bedingungen:

- Die Verbindung ist zielorientiert[49].
- Das Quellelement hat kein `xsi:nil="true"`-Attribut.
- Das dazugehörige Zielelement hat ein `xsi:nil="true"`-Attribut.

 © 2019-2025 Altova GmbH

- Quell- und Zielelement können den Typ simpleType oder complexType haben.

In diesem Fall überschreibt das Quellelement das Zielelement, das das Attribut `xsi:nil="true"` enthält. Unten sehen Sie ein Beispiel für eine Ausgabedatei. Das Element `<genre>` enthält im Zielelement das Attribut `xsi:nil="true"`. Dieses Element wurde zur Mapping-Laufzeit jedoch überschrieben. Daher hat das Element `<genre>` (*gelb markiert*) in der Ausgabe `Fiction`.

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

Das Quellelement vom ComplexType/beide Elemente vom ComplexType haben xsi:nil="true"
Für dieses Szenario gelten die folgenden Bedingungen:

- Die Verbindung ist zielorientiert [49].
- Das Quellelement hat den Typ complexType. Das Quellelement hat in unserem Beispiel ein Attribut `id="213"` und ein Attribut `xsi:nil="true"`. Das dazugehörige Zielelement hat ebenfalls den Typ complexType und ein Attribut `id="124"`, hat aber kein `xsi:nil="true"`-Attribut.
- Alternativ dazu können das Quell- und das Zielelement, beide vom Typ complexType, `xsi:nil="true"`-Attribute haben.

In diesem Fall überschreibt das Quellelement das Zielelement (*unten gelb markiert*). Das Attribut `xsi:nil="true"` wird jedoch nicht automatisch in die Ausgabe geschrieben. Damit das Attribut `xsi:nil="true"` in der Ausgabedatei im Zielelement aufscheint, verwenden Sie eine Alles kopieren [54]-Verbindung.

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

Nützliche Funktionen

Mit Hilfe der folgenden Funktionen können Sie NULL-Werte überprüfen, ersetzen und zuweisen:

- **`is-xsi-nil`** [290]: Damit können Sie explizit überprüfen, ob in einem Quellelement das Attribut `xsi:nil` auf `true` gesetzt ist.
- **`substitute-missing`** [325]: Ersetzt einen NULL-Wert im Quellelement durch etwas Bestimmtes.
- **`set-xsi-nil`** [293]: Weist einem Zielelement das Attribut `xsi:nil="true"` zu. Dies funktioniert bei Zieldatenelementen vom Typ "simpleType" und "complexType".
- **`substitute-missing-with-xsi-nil`** [294]: Wenn Inhalt vorhanden ist, wird dieser in das Zielelement geschrieben; wenn Werte fehlen, wird das Zielelement in der Ausgabe mit dieser Funktion mit dem Attribut `xsi:nil="true"` versehen.
- Wenn Sie die **`exists`** [299]-Funktion mit einem Quellelement mit einem NULL-Wert verbinden, wird `true` zurückgegeben, selbst wenn das Element keinen Inhalt hat.

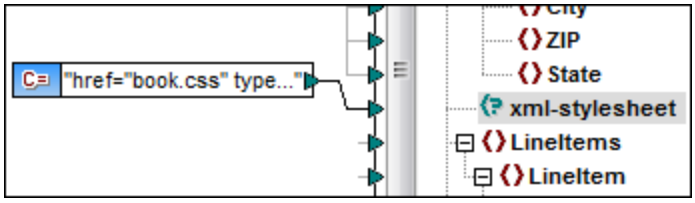
Beachten Sie, dass Funktionen, die `xsi:nil` generieren, nicht über Funktionen oder Komponenten übergeben werden können, die nur an Werten operieren (wie z.B. die **`if-else`**-Funktion).

4.1.4 Kommentare und Processing Instructions

In diesem Kapitel wird erläutert, wie Sie Kommentare und Processing Instructions in XML-Zielkomponenten einfügen. Beachten Sie, dass Kommentare und Processing Instructions nur Input-Verbindungen haben. Kommentare und Processing Instructions können nicht für Nodes definiert werden, die Teil einer "Alles kopieren" 54-Verbindung sind. Kommentare und Processing Instructions sind in der W3C-Spezifikation definiert

Einfügen eines Kommentars/einer Processing Instruction
Um eine Processing Instruction oder einen Kommentar einzufügen, gehen Sie folgendermaßen vor:

1. Klicken Sie mit der rechten Maustaste auf ein Element in der Komponente und wählen Sie **Kommentar/Processing Instruction davor/danach einfügen**. Wenn Sie eine Processing Instruction einfügen, müssen Sie auch ihren Namen eingeben. Im Beispiel unten wurde nach dem Element `State` eine Processing Instruction namens `xml-stylesheet` eingefügt.



3. Den Wert eines Kommentars oder einer Processing Instruction können Sie mit Hilfe einer Konstanten angeben, z.B. (*siehe Abbildung oben*).

Anmerkung: Sie können vor oder nach jedem Element in der Zielkomponente mehrere Processing Instructions hinzufügen.

Anmerkung: Vor oder nach einem einzigen Ziel-Node kann nur ein einziger Kommentar hinzugefügt werden. Um mehrere Kommentare zu erstellen, verwenden Sie die Funktion Duplikat einfügen 39.

Löschen eines Kommentars/einer Processing Instruction
Um einen Kommentar/eine Processing Instruction zu löschen, klicken Sie mit der rechten Maustaste auf den entsprechenden Node, wählen Sie **Kommentar/Processing Instruction** und anschließend aus dem Untermenü den Befehl **Kommentar/Processing Instruction löschen**.

4.1.5 CDATA-Abschnitte

CDATA-Abschnitte dienen dazu, Abschnitte eines Dokuments, die normalerweise als Markup interpretiert würden, als Zeichendaten darzustellen. Nähere Informationen zu CDATA-Abschnitten finden Sie in der W3C-Spezifikation. Die folgenden Ziel-Nodes können CDATA-Abschnitte erhalten: XML-Daten, in Datenbankfelder eingebettete XML-Daten und XML-Child-Elemente von typisierten Dimensions in einer XBRL-Zielkomponente. CDATA-Abschnitte können auch in duplizierten Nodes und `xsi:type`-Nodes definiert werden.

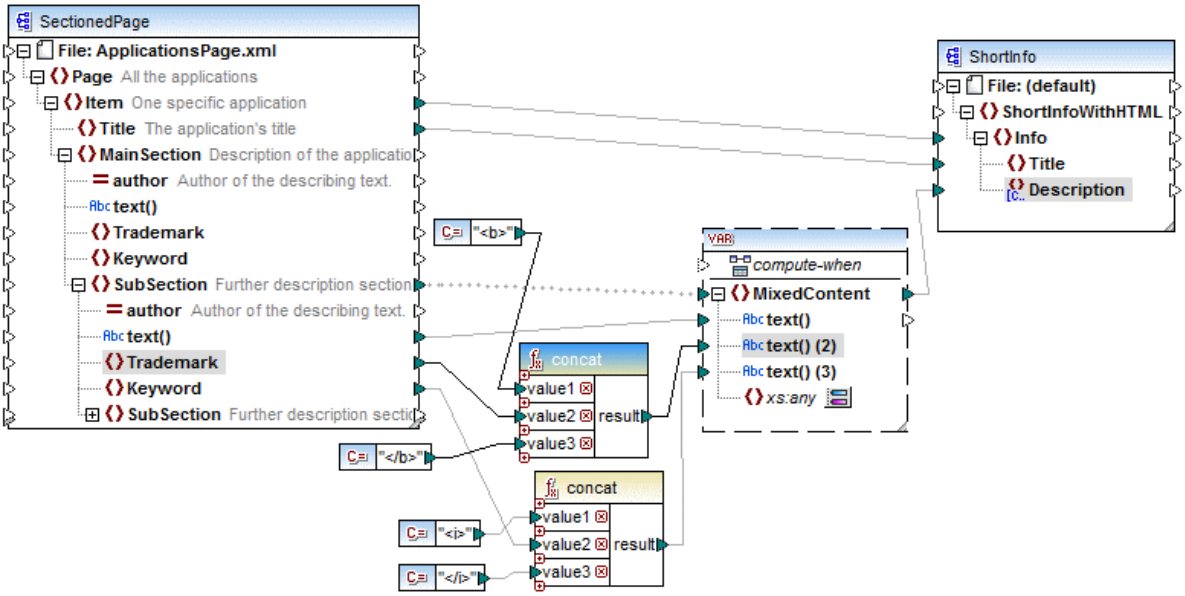
 © 2019-2025 Altova GmbH

Um einen CDATA-Abschnitt zu erstellen, klicken Sie mit der rechten Maustaste auf den entsprechenden Ziel-Node und wählen Sie die Befehl **Inhalt als CDATA-Abschnitt schreiben**. Daraufhin erscheint eine Meldung, in der Sie gewarnt werden, dass die Input-Daten das schließende CDATA-Abschnittstrennzeichen `]]>` nicht enthalten dürfen. Das Symbol `[C..` wird unterhalb des Element-Tags angezeigt und bedeutet, dass dieser Node nun als CDATA-Abschnitt definiert ist.

Beispiel

Im Beispiel unten sehen Sie ein Szenario, in dem sich ein CDATA-Abschnitt als nützlich erweisen könnte. Das Beispielmapping **`MapForceExamples\HTMLinCDATA.mfd`** (*siehe Abbildung unten*) hat die folgenden Aspekte:

- Das Element `SubSection` hat gemischten Inhalt. Nähere Informationen zu Nodes mit gemischtem Inhalt finden Sie unter Quellorientierte Verbindungen 49.
- Mit Hilfe der Funktion **`concat`** erhält der Inhalt des Elements `Trademark` die Tags `<b></b>`.
- Der Inhalt des Elements `Keyword` erhält die Tags `<i></i>`.
- Die Daten mit den neuen Tags werden in derselben Reihenfolge wie im Quelldokument an die duplizierten `text()`-Nodes übergeben.
- Die Ausgabe des `MixedContent` Node wird anschließend an den `Description` Node in der Zielkomponente `ShortInfo` übergeben. Der `Description`-Node wurde als CDATA-Abschnitt definiert.



Ausgabe

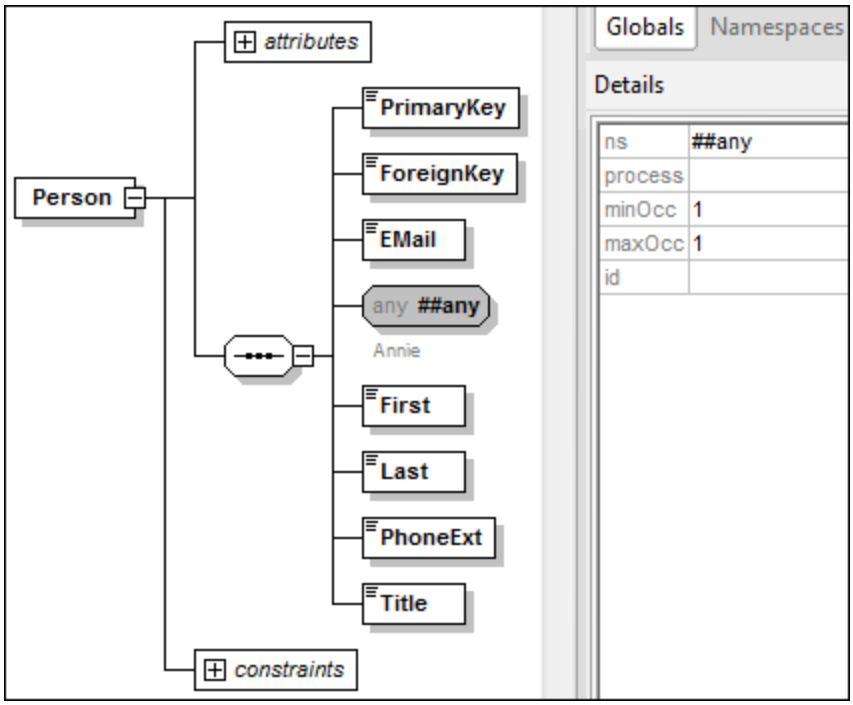
Klicken Sie auf das Fenster **Ausgabe**, um den CDATA-Abschnitt im Node `Description` zu sehen (*Abbildung unten*).

```
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>
```

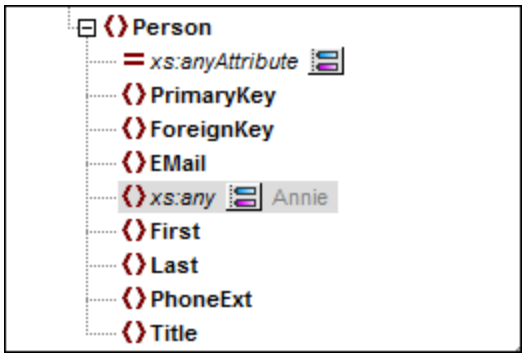
4.1.6 Wildcards - xs:any / xs:anyAttribute

In diesem Kapitel wird beschrieben, wie Sie Wildcards in Mappings behandeln können. Mit Hilfe der Wildcards `xs:any` und `xs:anyAttribute` können Sie in Ihrer Schema-Datei definierte any-Elemente/Attribute verwenden. Nähere Informationen zu Wildcards finden Sie in der W3C-Spezifikation.

Wildcards in der Schema-Definition
In der Abbildung unten wurde ein `xs:any`-Element als Child-Element des `Person`-Elements definiert (*Schema-Ansicht in Altova XMLSpy*).

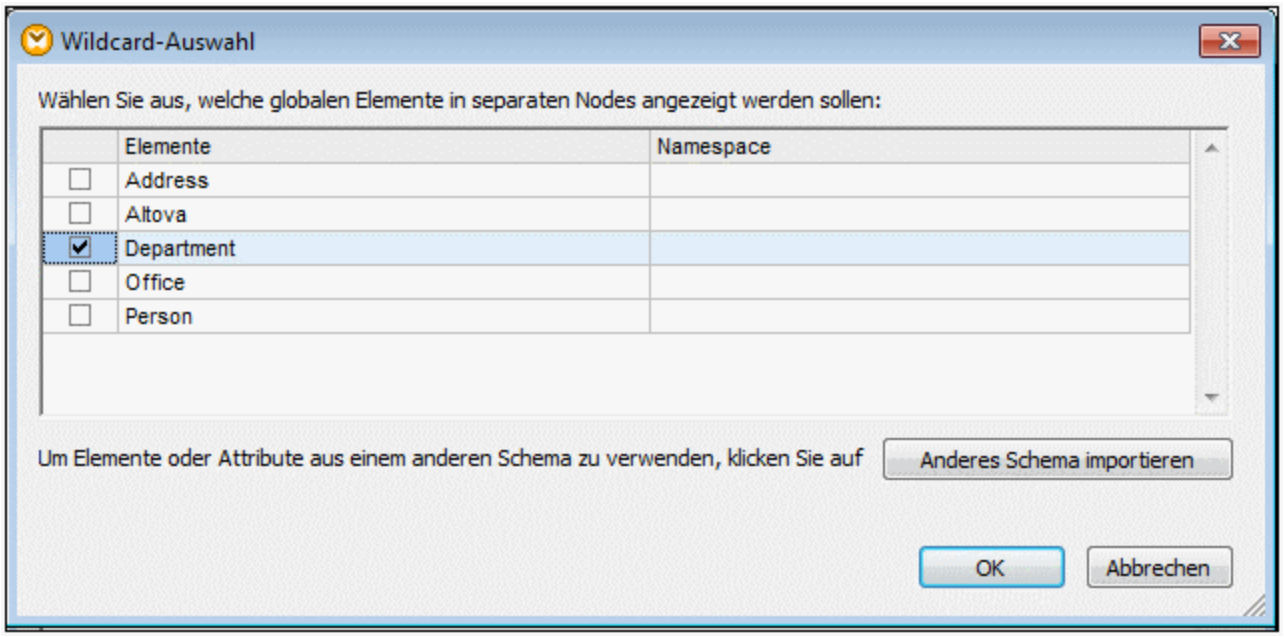


Wildcards in MapForce
Wenn für ein Element und/oder Attribut eine Wildcard definiert wurde, wird neben diesem Node eine -Schaltfläche (**Auswahl ändern**) angezeigt (*siehe Abbildung unten*).

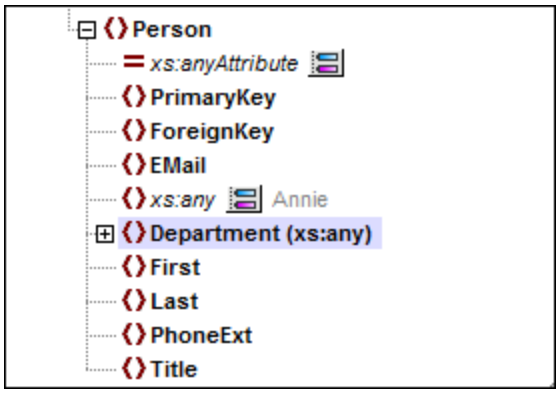


Wildcard-Auswahl

Wir möchten nun ein weiteres Element als separaten Node hinzufügen. Klicken Sie auf die Schaltfläche , um die Liste der Elemente, die zur Struktur hinzugefügt werden können, zu sehen. Beachten Sie, dass im Dialogfeld **Wildcard-Auswahl** nur Elemente und Attribute angezeigt werden, die in Ihrem Schema global deklariert wurden (*siehe Abbildung unten*).



Für dieses Beispiel haben wir `Department` ausgewählt. Beachten Sie, dass Wildcard-Elemente und -Attribute nach dem Node mit der Schaltfläche eingefügt werden. Unsere Komponente sieht nun folgendermaßen aus:



Sie können nun zwischen diesen Nodes wie gewöhnlich Mapping-Verbindungen erstellen. Wildcard-Elemente und -Attribute werden in einer Komponente mit (`xs:any`) bzw. (`xs:anyAttribute`) gekennzeichnet (*siehe Abbildung oben*).

Entfernen von Wildcards

Um einen Wildcard-Node zu entfernen, klicken Sie auf die und deaktivieren Sie das entsprechende Kontrollkästchen im Dialogfeld **Wildcard-Auswahl**.

Elemente/Attribute aus einem anderen Schema

Über das Dialogfeld **Wildcard-Auswahl** (*siehe oben*) können Elemente/Attribute aus einem anderen Schema verwendet werden. Wenn Sie auf die Schaltfläche **Anderes Schema importieren** klicken, haben Sie die folgenden Optionen: (i) Import einer Schema-Datei und (ii) Generierung eines Wrapper-Schemas (*siehe Beschreibung unten*).

Import eines Schemas
Mit der Option **Schema importieren** wird das externe Schema in das aktuelle, der Komponente zugewiesene Schema importiert. Beachten Sie, dass mit dieser Option das vorhandene Schema auf der Festplatte ersetzt wird. Wenn es sich beim aktuellen Schema nicht um ein entferntes über eine URL geöffnetes Schema (siehe Hinzufügen von Komponenten über eine URL 35) handelt und das Schema nicht von der lokalen Festplatte stammt, kann es nicht geändert werden. Verwenden Sie in diesem Fall die Option **Wrapper-Schema generieren**.

Generieren eines Wrapper-Schemas
Mit der Option **Wrapper-Schema generieren** wird eine neue als *Wrapper-Schema* bezeichnete Schema-Datei erstellt. Der Vorteil bei dieser Option ist, dass das bestehende Schema der Komponente nicht geändert wird. Stattdessen wird ein neues Schema erstellt, das sowohl das vorhandene Schema als auch das importierte Schema enthält. Wenn Sie diese Option auswählen, werden Sie gefragt, wo das Wrapper-Schema gespeichert werden soll. Standardmäßig hat das Wrapper-Schema das Format **`dateiname-wrapper.xsd`**.

Nachdem Sie das Wrapper-Schema gespeichert haben, wird es der Komponente standardmäßig automatisch zugewiesen. Außerdem werden Sie gefragt, ob Sie den Schemapfad anpassen möchten, um das vorherige Hauptschema referenzieren zu können. Klicken Sie auf **Ja**, um zum vorherigen Schema zurückzuwechseln oder auf **Nein**, damit das neu erstellte Wrapper-Schema der Komponente zugewiesen wird.

Wildcards oder dynamische Node-Namen

Es gibt Fälle, in denen eine Instanz zu viele Elemente und/oder Attribute enthält, als dass diese im Schema deklariert werden können. Betrachten Sie die folgende Beispieldatei:

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

In solchen Fällen wird empfohlen, anstelle von Wildcards dynamische Node-Namen zu verwenden. Nähere Informationen dazu finden Sie unter Mappen von Node-Namen 405.

4.1.7 Benutzerdefinierte Namespaces

Wenn mit einem Mapping eine XML-Ausgabe erzeugt wird, so übernimmt MapForce den Namespace (oder eine Gruppe von Namespaces) für die einzelnen Elemente und Attribute automatisch aus dem Ziel-Schema. Dies ist das Standardverhalten, das für die meisten Szenarien, in denen eine XML-Ausgabe generiert werden soll,

 © 2019-2025 Altova GmbH

geeignet ist. In einigen Fällen müssen Sie den Namespace eines Elements jedoch direkt über das Mapping manuell deklarieren.

Die Deklaration benutzerdefinierter Namespaces ist nur bei XML-Zielkomponenten sinnvoll und gilt nur für Elemente. Der Befehl **Namespace hinzufügen** steht für Attribute, Wildcard Nodes und für Nodes, die Daten aus einer "Alles kopieren"-Verbindung 54 erhalten, nicht zur Verfügung.

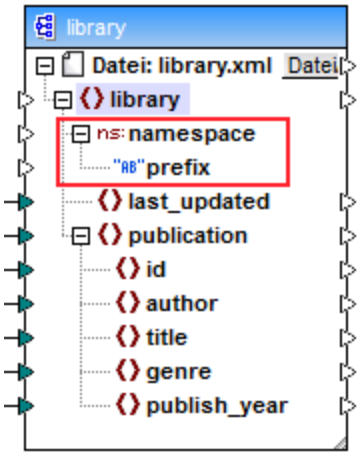
Um zu verstehen, wie benutzerdefinierte Namespaces funktionieren, folgen Sie der Anleitung in Unterabschnitt weiter unten.

Manuelle Deklaration eines Namespace

Sie benötigen für dieses Beispiel das folgende Mapping: **`BasicTutorials\Tut1-OneToOne.mfd`**.

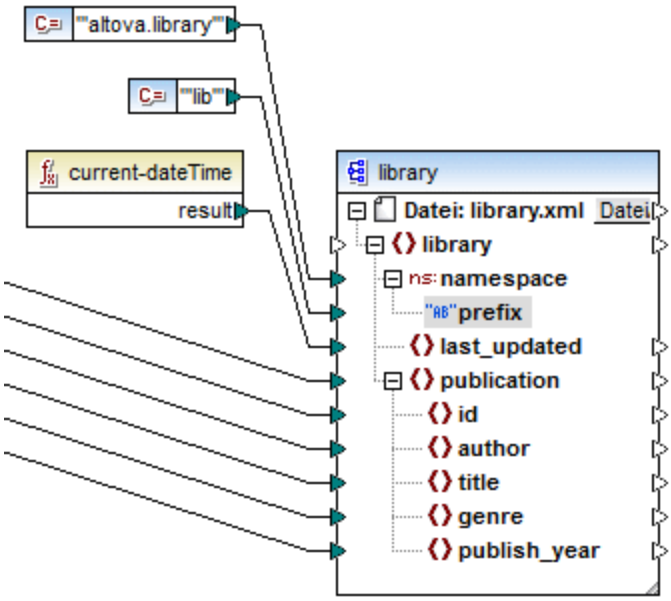
Hinzufügen eines Namespace

Öffnen Sie das Mapping, kllicken Sie in der Komponente `BooksOutput` mit der rechten Maustaste auf den Node `library` und wählen Sie im Kontextmenü den Befehl **Namespace hinzufügen**. Unter dem Element `library` stehen nun zwei neue Nodes zur Verfügung: `namespace` und `prefix` (*siehe Abbildung unten*).



Bereitstellen von Namespace-Werten

Im nächsten Schritt müssen nun Werte für die Nodes `namespace` und `prefix` angegeben werden. Zu diesem Zweck verwenden wir zwei Konstanten mit den folgenden String-Werten: `altova.library` und `lib` (*siehe Abbildung unten*).



Anmerkung: Sowohl der `namespace`- aus auch der `prefix`-Input-Konnektor muss gemappt werden, auch wenn Sie dafür leere Werte angeben.

Ausgabe
In der generieren Ausgabe wird zum Element ein `xmlns:<prefix>="<namespace>"`-Attribut hinzugefügt, wobei `<prefix>` und `<namespace>` Werte sind, die vom Mapping geliefert werden. Die Ausgabe sieht nun folgendermaßen aus (*Beachten Sie den markierten Teil*):

```
<?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">
...
```

Sie können für dasselbe Element auch mehrere Namespaces deklarieren, falls nötig. Klicken Sie dazu mit der rechten Maustaste auf den Node und wählen Sie im Kontextmenü den Befehl **Namespace hinzufügen**. Es steht nun ein neues Paar an "namespace" und "prefix"-Nodes zur Verfügung, mit dem Sie neue Präfix- und Namespace-Werte verbinden können.

Deklarieren eines Standard-Namespace
Wenn Sie einen Standard-Namespace deklarieren möchten, mappen Sie einen leeren String-Wert auf `prefix`. Die Ausgabe würden dann folgendermaßen aussehen (*Beachten Sie den markierten Teil*):

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

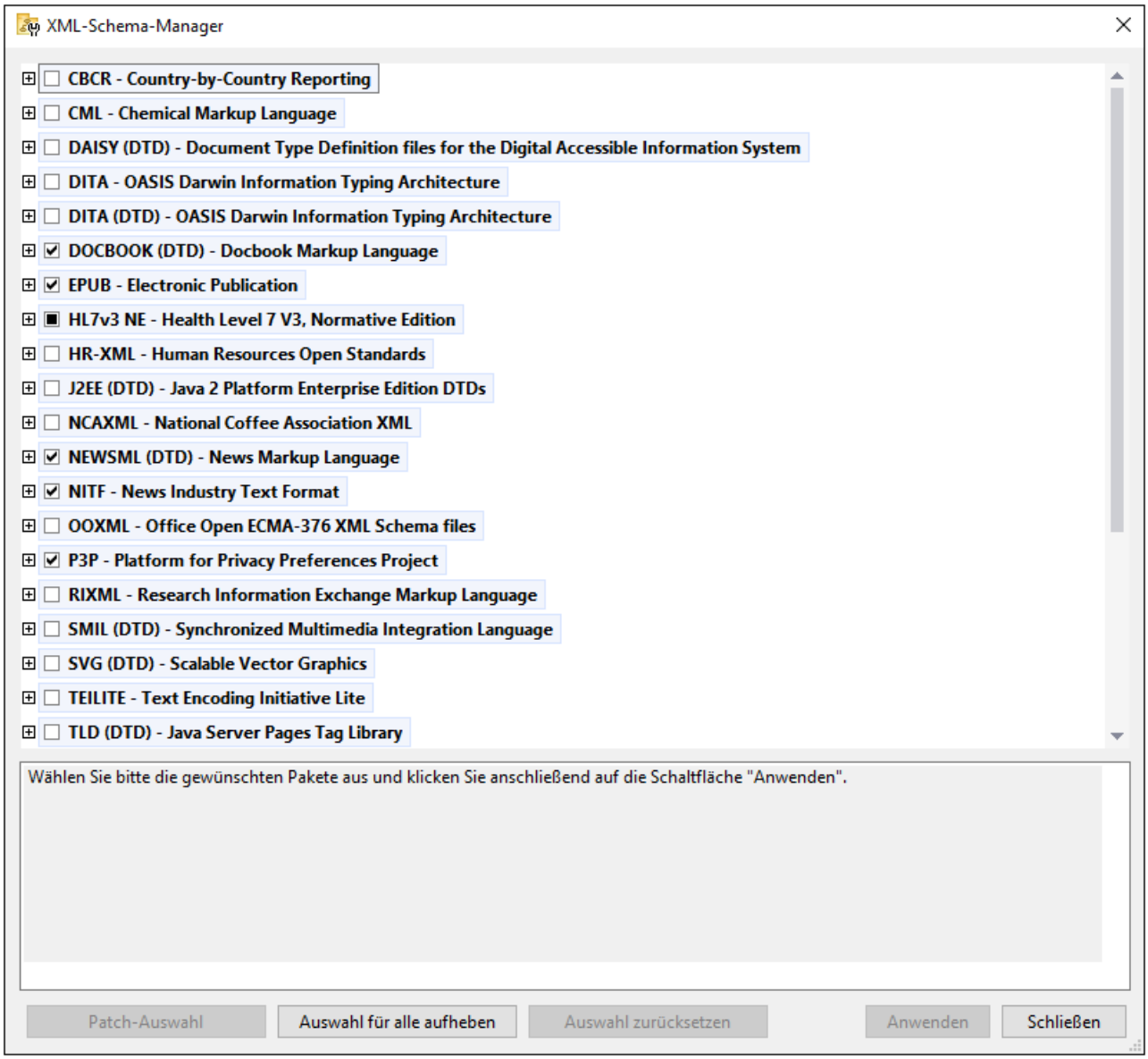
Wenn Sie Präfixe für Attributnamen erstellen müssen, z.B. `<number prod:id="prod557">557</number>`, können Sie dies durch dynamischen Zugriff auf die Attribute eines Node (siehe Mappen von Node-Namen. 405) oder durch Bearbeiten des Schemas tun, so dass es ein `prod:id`-Attribut für `<number>` hat.

Entfernen eines Namespace
Um eine zuvor hinzugefügte Namespace-Deklaration zu entfernen, klicken Sie mit der rechten Maustaste auf den Node `ns:namespace` und wählen Sie im Kontextmenü den Befehl **Namespace entfernen**.

4.1.8 Schema-Manager

Der XML-Schema-Manager ist ein Altova-Tool, mit dem Sie XML-Schemas (DTDs für XML-Dateien und XML-Schemas) zentral installieren und verwalten können, um diese in allen XML-Schema-fähigen Applikationen von Altova einschließlich MapForce verwenden zu können.

- Unter Windows hat der Schema-Manager eine grafische Benutzeroberfläche (*siehe Abbildung unten*) und steht auch über die Befehlszeile zur Verfügung. (Die Desktop-Applikationen von Altova stehen nur unter Windows zur Verfügung; *siehe Liste unten*).
- Unter Linux und MacOS steht der Schema-Manager nur über die Befehlszeile zur Verfügung. (Die Server-Applikationen von Altova stehen unter Windows, Linux und macOS zur Verfügung; *siehe Liste unten*).



Altova-Applikationen, die mit Schema-Manager arbeiten:

Desktop-Applikationen (nur Windows)	Server-Applikationen (Windows, Linux, macOS)
XMLSpy (alle Editionen)	RaptorXML Server, RaptorXML+XBRL Server
MapForce (alle Editionen)	StyleVision Server
StyleVision (alle Editionen)	
Authentic Desktop Enterprise Edition	

 © 2019-2025 Altova GmbH

Installation und Deinstallation des Schema-Managers

Der Schema-Manager wird bei der ersten Installation einer neuen Version des Altova Mission Kit oder einer der XML-Schema-fähigen Applikationen von Altova (*siehe Tabelle oben*) automatisch installiert.

Ebenso wird er auch automatisch entfernt, wenn Sie die letzte XML-Schema-fähige Applikation von Altova auf Ihrem Rechner deinstallieren.

Schema-Manager-Funktionalitäten

Im Schema-Manager stehen die folgenden Funktionalitäten zur Verfügung:

- Anzeigen der auf Ihrem Rechner installierten XML-Schemas und Überprüfung, ob neue Versionen zum Download zur Verfügung stehen.
- Download neuer Versionen von XML-Schemas unabhängig vom Altova Produkt-Release-Zyklus. (Die Schemas werden von Altova online bereitgestellt und können über den Schema-Manager heruntergeladen werden).
- Installation oder Deinstallation jeder beliebigen (oder ggf. aller) der zahlreichen Versionen eines bestimmten Schemas.
- Ein XML-Schema kann Abhängigkeiten von anderen Schemas aufweisen. Bei der Installation oder Deinstallation eines bestimmten Schemas informiert Sie der Schema-Manager über davon abhängige Schemas und installiert bzw. entfernt diese ebenfalls automatisch.
- Der Schema-Manager ordnet Schema-Referenzen mit Hilfe des XML-Katalogs lokalen Dateien zu. Dadurch lassen sich große XML-Schemas schneller verarbeiten, als wenn sie sich unter einem entfernten Pfad befinden.
- Alle wichtigen Schemas werden über den Schema-Manager bereitgestellt und regelmäßig auf die jeweils neuesten Version aktualisiert. Dadurch können alle Ihre Schemas zentral verwaltet werden und stehen allen XML-Schema-fähigen Applikationen von Altova jederzeit zur Verfügung.
- Im Schema-Manager vorgenommene Änderungen werden für alle auf dem Rechner installierten Altova-Produkte wirksam.
- Wenn Sie versuchen ein Dokument in einem Altova-Produkt anhand eines nicht installierten aber über Schema-Manager verfügbaren Schemas zu validieren, wird das Schema automatisch installiert. Wenn das Schema-Paket jedoch Namespace-Zuordnungen enthält, wird das Schema nicht automatisch installiert; in diesem Fall müssen Sie Schema-Manager starten, das/die gewünschte(n) Paket(e) auswählen und die Installation starten. Wenn Ihre offene Altova-Applikation nach der Installation nicht automatisch neu gestartet wird, müssen Sie sie manuell neu starten.

Funktionsweise

Alle in Altova-Produkten verwendeten XML-Schemas werden von Altova online bereitgestellt. Dieser Speicher wird bei Veröffentlichung neuer Versionen der Schemas aktualisiert. Im Schema-Manager werden sowohl bei Aufruf über die Benutzeroberfläche als auch über das CLI Informationen über die neuesten verfügbaren Schemas angezeigt. Sie können die gewünschten Schemas dann über den Schema-Manager installieren, aktualisieren oder deinstallieren.

Schemas können vom Schema-Manager auch auf eine weitere Art installiert werden. Sie können ein Schemas und die davon abhängigen Schemas auf der Altova Website (https://www.altova.com/de/schema-manager) auswählen. Daraufhin wird auf der Website eine Datei des Typs **`.altova_xmlschemas`** mit Informationen über Ihre ausgewählten Schemas zum Download vorbereitet. Bei Doppelklick auf diese Datei oder bei Übergabe an den Schema-Manager über das CLI als Argument des Befehls `install` 158 installiert der Schema-Manager die ausgewählten Schemas.

Lokaler Cache: Überprüfung Ihrer Schemas
Alle Informationen über installierte Schemas werden in einem zentralen Cache-Verzeichnis auf Ihrem Rechner aufgezeichnet. Das Verzeichnis befindet sich hier:

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

Dieses Cache-Verzeichnis wird regelmäßig mit dem neuesten Status der Schemas aus dem Online-Speicher von Altova aktualisiert. Diese Aktualisierungen finden unter den folgenden Bedingungen statt:

- bei jedem Start von Schema-Manager.
- Wenn Sie MapForce zum ersten Mal an einem bestimmten Kalendertag starten.
- Wenn MapForce länger als 24 Stunden geöffnet ist, findet alle 24 Stunden eine Aktualisierung des Cache statt.
- Sie können den Cache auch durch Ausführung des `update` 161-Befehls über die Befehlszeilenschnittstelle aktualisieren.

Der Schema-Manager kann somit Ihre installierten Schemas über den Cache ständig anhand der online verfügbaren Schemas auf der Altova Website überprüfen.

Nehmen Sie keine manuellen Änderungen am Cache vor!

Das lokale Cache-Verzeichnis wird automatisch auf Basis der installierten oder deinstallierten Schemas verwaltet; es sollte nicht manuell geändert oder gelöscht werden. Falls Sie den Schema-Manager je in seinen Originalzustand zurücksetzen möchten, (i) führen Sie den CLI-Befehl `reset` 160 der Befehlszeilenschnittstelle und (ii) anschließend den Befehl `initialize` 158 aus. (Führen Sie alternativ dazu den Befehl **`reset`** mit der Option **`-i`** aus).

4.1.8.1 Ausführen des Schema-Managers

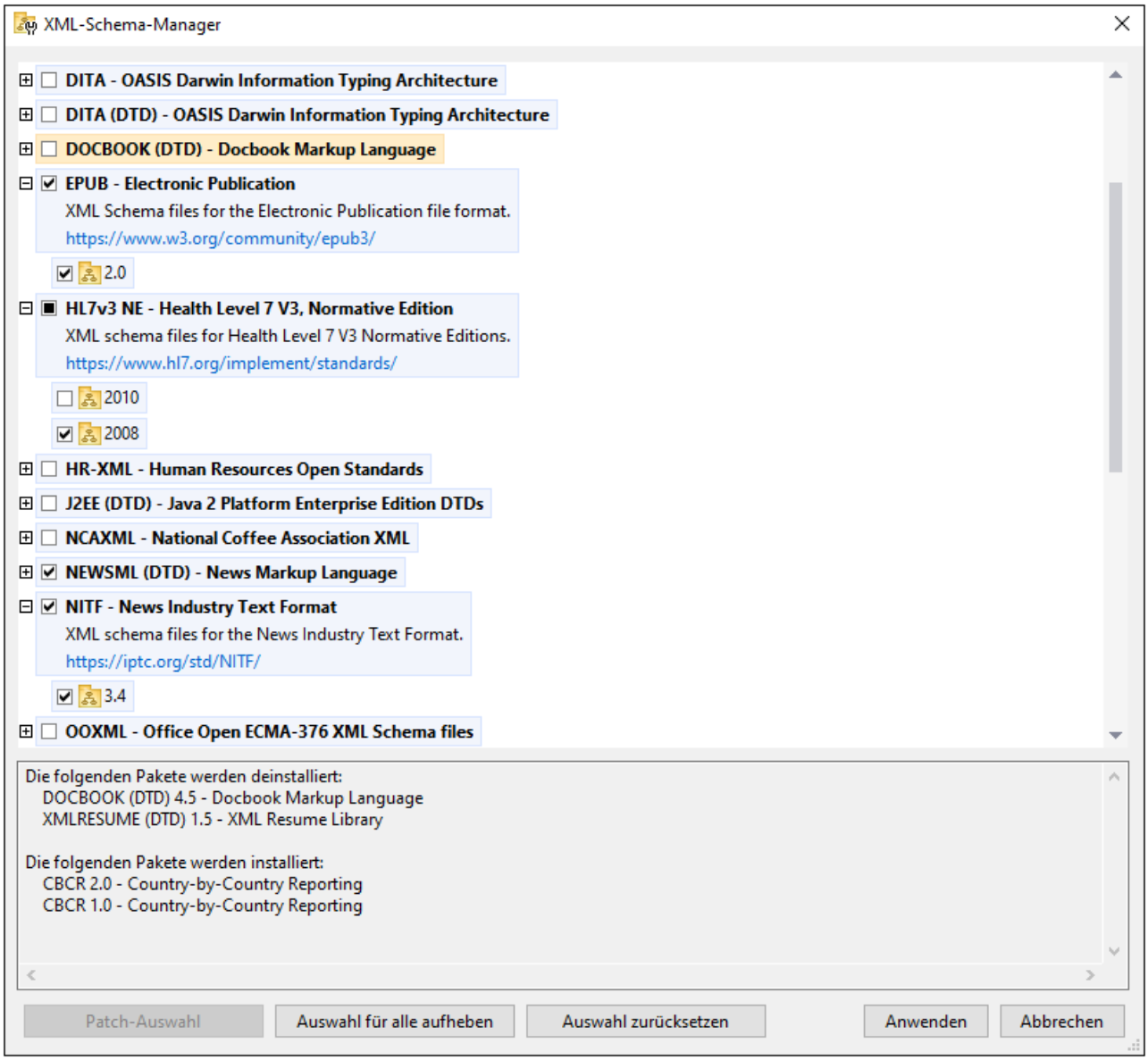
Grafische Benutzeroberfläche

Sie können die Benutzeroberfläche des Schema-Managers auf eine der folgenden Arten aufrufen:

- *Bei der Installationen von MapForce:* Aktivieren Sie gegen Ende der Installation das Kontrollkästchen *Altova-Schema-Manager aufrufen*, wodurch Sie die Benutzeroberfläche des Schema-Managers direkt aufrufen können. Auf diese Art können Sie Schemas während der Installation Ihrer Altova-Applikation installieren.
- *Nach der Installation von MapForce:* Nachdem die Applikation installiert wurde, können Sie die Benutzeroberfläche des Schema-Managers jederzeit über den Menübefehl **Tools | XML-Schemas-Manager** aufrufen.
- über die von der Altova Webseite heruntergeladene **`.altova_xmlschemas`**-Datei: Doppelklicken Sie auf die heruntergeladene Datei, um den Schema-Manager zu starten, der daraufhin die (auf der Website) ausgewählten Schemas installiert.

 © 2019-2025 Altova GmbH

Nachdem die Benutzeroberfläche des Schema-Managers geöffnet wurde (*Abbildung unten*), werden bereits installierte Schemas markiert angezeigt. Wenn ein zusätzliches Schema installiert werden soll, aktivieren Sie dieses. Wenn ein bereits installiertes Schema deinstalliert werden soll, deaktivieren Sie dieses. Nachdem Sie Ihre Auswahl getroffen haben, können Ihre Änderungen angewendet werden. Die Schemas, die installiert bzw. deinstalliert werden, werden markiert und im Fenster "Meldungen" am unteren Rand des Schema-Manager-Fensters (*siehe Abbildung*) erscheint eine Meldung über die bevorstehenden Änderungen.



Wenn Sie auf **Anwenden** klicken, wird der Fortschritt der Installation angezeigt. Bei Auftreten eines Fehlers (z.B. eines Verbindungsfehlers) wird eine Fehlermeldung angezeigt. Klicken Sie in diesem Fall auf die Schaltfläche **Erweitert**, die im Dialogfeld erscheint, überprüfen Sie die Schemaauswahl und versuchen Sie es erneut durch Klicken auf **Anwenden**.

Befehlszeilenschnittstelle

Sie können den Schema-Manager über eine Befehlszeilenschnittstelle starten, indem Sie Befehle an die ausführbare Datei **`xmlschemamanager.exe`** senden.

Die Datei **`xmlschemamanager.exe`** steht im folgenden Ordner zur Verfügung:

- *Unter Windows:* `C:\ProgramData\Altova\SharedBetweenVersions`
- *Unter Linux oder macOS (nur Server-Applikationen):* `%INSTALLDIR%/bin`, wobei `%INSTALLDIR%` das Installationsverzeichnis des Programms ist.

Anschließend können Sie jeden der im Abschnitt zur CLI-Befehlsreferenz 156 aufgelisteten Befehle verwenden.

Um die Hilfe zu den Befehlen anzuzeigen, führen Sie den folgenden Befehl aus:

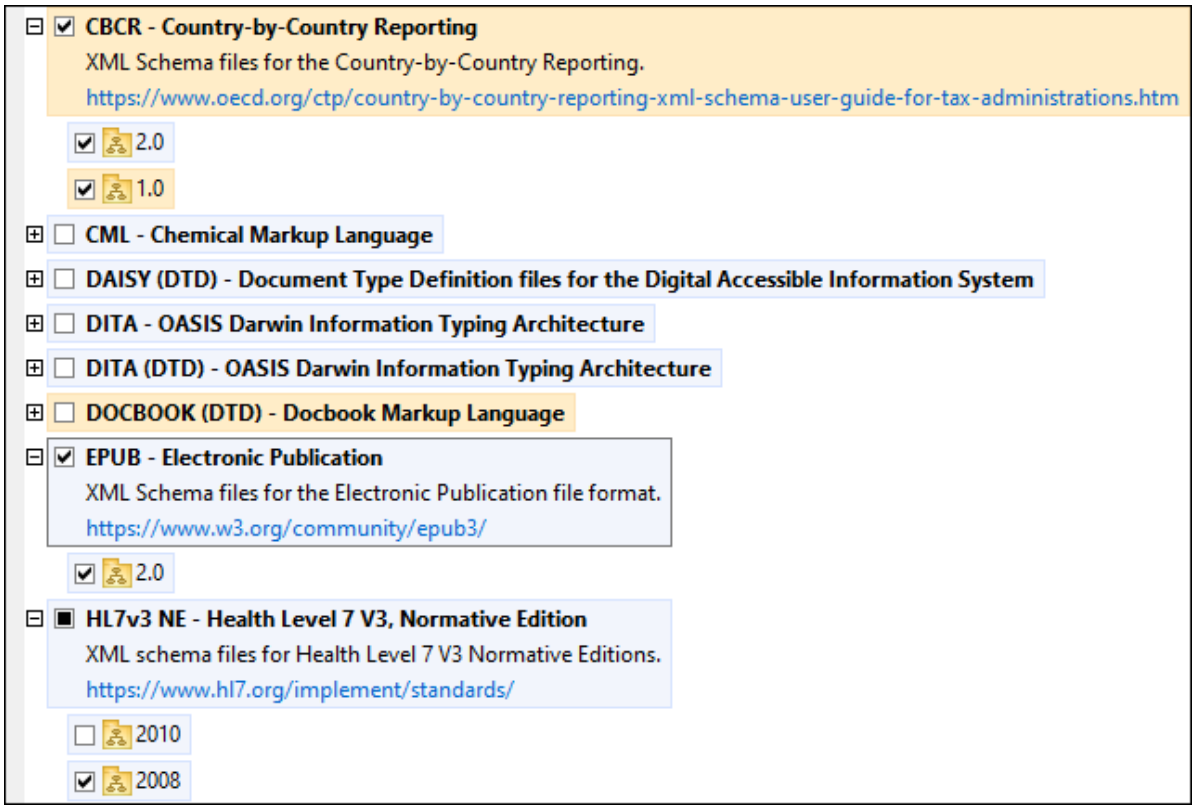
- *Unter Windows:* `xmlschemamanager.exe --help`
- *Unter Linux oder macOS (nur Server-Applikationen):* `sudo ./ xmlschemamanager --help`

4.1.8.2 Statuskategorien

Der Schema-Manager unterscheidet folgendermaßen zwischen den von ihm verwalteten Schemas:

- *installierte Schemas.* Diese werden auf der Benutzeroberfläche mit einem Häkchen angezeigt (*in der Abbildung unten sind die mit einem Häkchen versehenen und blau angezeigten Versionen der EPUB- und HL7v3 NE-Schemas installiert*). Wenn alle Versionen eines Schemas ausgewählt sind, wird ein Häkchen angezeigt. Wenn zumindest eine Version nicht ausgewählt ist, wird ein gefülltes Quadrat angezeigt. Sie können das Kontrollkästchen für ein installiertes Schema deaktivieren, um es zu **deinstallieren**; (*in der Abbildung unten ist die DocBook DTD installiert und wurde deaktiviert; sie wird daher für die Deinstallation vorbereitet*).
- *Nicht installierte verfügbare Schemas.* Diese werden auf der Benutzeroberfläche mit einem deaktivierten Kontrollkästchen angezeigt. Sie können die Schemas, die **installiert** werden sollen, auswählen.

 © 2019-2025 Altova GmbH



- *Schemas, für die ein Upgrade zur Verfügung steht* sind diejenigen, die seit ihrer Installation vom Herausgeber überarbeitet wurden. Sie werden auf der Benutzeroberfläche durch ein -Symbol gekennzeichnet. Sie können für ein installiertes Schema ein **Patch** der verfügbaren überarbeiteten Version installieren.

Wichtige Punkte

- In der Abbildung oben sind beide CBCR-Schemas ausgewählt. Dasjenige mit einem blauen Hintergrund ist bereits installiert. Dasjenige mit dem gelben Hintergrund ist nicht installiert und wurde für die Installation ausgewählt. Beachten Sie, dass das Schema "HL7v3 NE 2010" nicht installiert ist und nicht für die Installation ausgewählt wurde.
- Ein gelber Hintergrund bedeutet, dass das Schema auf irgendeine Art geändert wird, wenn Sie auf die Schaltfläche **Anwenden** klicken. Wenn ein Schema deaktiviert ist und einen gelben Hintergrund hat, bedeutet dies, dass es bei Klick auf die Schaltfläche **Anwenden** deinstalliert wird. In der Abbildung oben ist dies bei der DocBook DTD der Fall.
- Bei Ausführung des Schema-Managers über die Befehlszeile wird der Befehl `list` 159 mit verschiedenen Optionen verwendet, um verschiedene Schemakategorien aufzulisten:

`xmlschemamanager.exe list`	Listet alle installierten und verfügbaren Schemas auf; auch verfügbare Upgrades werden angezeigt.
`xmlschemamanager.exe list -i`	Listet nur installierte Schemas auf; auch verfügbare Upgrades werden angezeigt.

`xmlschemamanager.exe list -u`	Listet Schemas auf, für die Upgrades zur Verfügung stehen

Anmerkung: Verwenden Sie unter Linux und macOS **`sudo ./xmlschemamanager list`**

4.1.8.3 Anwenden eines Patch oder Installation eines Schemas

Anwenden eines Patch auf ein installiertes Schema

Von Zeit zu Zeit werden von den Herausgebern der XML-Schemas Patches (Upgrades oder Überarbeitungen) veröffentlicht. Wenn der Schema-Manager erkennt, dass Patches zur Verfügung stehen, werden diese in der Schemaliste des Schema-Managers angezeigt und Sie können diese Patches schnell installieren.

Über die Benutzeroberfläche

Patches werden mit dem Symbol gekennzeichnet. (*Siehe auch vorhergehendes Kapitel über Statuskategorien*[152]*).* Falls Patches zur Verfügung stehen, ist die Schaltfläche **Patch-Auswahl** aktiv. Klicken Sie darauf, um alle Patches für die Installation auszuwählen und vorzubereiten. Auf der Benutzeroberfläche ändert sich das Symbol von Schemas, für die ein Patch installiert wird von in , und im Fenster "Meldungen" am unteren Rand des Dialogfelds werden die Patches, die angewendet werden, aufgelistet. Sobald Sie mit der Auswahl fertig sind, klicken Sie auf **Anwenden**. Alle Patches werden gemeinsam angewendet. Beachten Sie, dass ein für die Installation eines Patch markiertes Schema deinstalliert wird, wenn Sie die Auswahl aufheben.

Über das CLI

So wenden Sie einen Patch über die Befehlszeilenschnittstelle an:

1. Führen Sie den Befehl `list -u`[159] aus. Daraufhin werden alle Schemas, für die Upgrades zur Verfügung stehen, aufgelistet.
2. Führen Sie den Befehl `upgrade`[162] aus, um alle Patches zu installieren.

Installieren eines verfügbaren Schemas

Sie können Schemas entweder über die Benutzeroberfläche des Schema-Managers oder durch Senden der Schema-Manager-Installationsbefehle über die Befehlszeile installieren.

Anmerkung: Wenn das aktuelle Schema andere Schemas referenziert, werden auch die referenzierten Schemas installiert.

Über die Benutzeroberfläche

Um Schemas über die Benutzeroberfläche des Schema-Managers zu installieren, wählen Sie die gewünschten Schemas aus und klicken Sie auf **Anwenden**.

Sie können die gewünschten Schemas auch auf der Altova Website auswählen und eine herunterladbare **`.altova_xmlschemas`**-Datei generieren. Bei Doppelklick auf diese Datei wird der Schema-Manager aufgerufen, in dem die gewünschten Schemas bereits vorausgewählt sind. Sie müssen nur mehr auf **Anwenden** klicken.

Über das CLI

Um Schemas über die Befehlszeile zu installieren, rufen Sie den Befehl `install`[158] auf:

 © 2019-2025 Altova GmbH

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

wobei es sich bei **Schema** um das/die gewünschte(n) Schema(s) bzw. eine **.altova_xmlschemas**-Datei handelt. Ein Schema wird von einem Identifier im Format **<name>-<version>** referenziert. (Die Identifier von Schemas werden angezeigt, wenn Sie den Befehl list[159] ausführen.) Sie können beliebig viele Schemas eingeben. Nähere Informationen dazu finden Sie unter der Beschreibung des Befehls install[158].

Anmerkung: Verwenden Sie unter Linux oder macOS den Befehl **sudo ./xmlschemamanager**.

Installation eines benötigten Schemas
Wenn Sie einen XML-Schema-Befehl in MapForce ausführen und MapForce erkennt, dass ein zur Ausführung des Befehls erforderliches Schema nicht vorhanden oder unvollständig ist, wird der Schema-Manager mit Informationen über das/die fehlende(n) Schema(s) aufgerufen. Sie können die gewünschten Schemas dann über den Schema-Manager direkt installieren.

Alle bereits installierten Schemas können jederzeit durch Aufruf des Schema-Managers über **Extras | Schema-Manager** über die Benutzeroberfläche des Schema-Managers angezeigt werden.

4.1.8.4 Deinstallieren eines Schemas, Zurücksetzen

Deinstallieren eines Schemas

Sie können Schemas entweder über die Benutzeroberfläche des Schema-Managers oder durch Senden der Schema-Manager-Deinstallationsbefehle über die Befehlszeile deinstallieren.

Anmerkung: Wenn das gewünschte Schema andere Schemas referenziert, so werden auch die referenzierten Schemas deinstalliert.

Über die Benutzeroberfläche
Um Schemas über die Benutzeroberfläche des Schema-Managers zu deinstallieren, deaktivieren Sie die Kontrollkästchen der entsprechenden Schemas und klicken Sie auf **Anwenden**. Daraufhin werden die ausgewählten Schemas und die davon referenzierten Schemas deinstalliert.

Um alle Schemas zu deinstallieren, klicken Sie auf **Auswahl für alle aufheben** und anschließend auf **Anwenden**.

Über das CLI
Um Schemas über die Befehlszeile zu deinstallieren, rufen Sie den Befehl uninstall[160] auf:

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

wobei es sich beim Argument **Schema** ein zu deinstallierendes Schema oder eine **.altova_xmlschemas**-Datei handelt. Ein Schema wird von einem Identifier im Format **<name>-<version>** definiert. (Die Identifier von Schemas werden angezeigt, wenn Sie den Befehl list[159] ausführen.) Sie können beliebig viele Schemas eingeben. Nähere Informationen dazu finden Sie unter der Beschreibung des Befehls uninstall[160].

Anmerkung: Verwenden Sie unter Linux oder macOS den Befehl **`sudo ./xmlschemamanager`**.

Zurücksetzen des Schema-Managers

Sie können den Schema-Manager zurücksetzen. Damit werden alle installierten Schemas und das Cache-Verzeichnis entfernt.

- Klicken Sie auf der Benutzeroberfläche auf **Auswahl zurücksetzen**.
- Führen Sie über die Benutzeroberfläche den Befehl `reset` 160 aus.

Nachdem Sie diesen Befehl ausgeführt haben, muss der Befehl `initialize` 158 ausgeführt werden, um das Cache-Verzeichnis neu zu erstellen. Führen Sie alternativ dazu den Befehl `reset` 160 mit der Option `-i` aus.

Beachten Sie, dass mit `reset-i` 160 die Originalinstallation des Produkts wiederhergestellt wird, daher wird empfohlen, nach dem Zurücksetzen auch den Befehl `update` 161 auszuführen. Führen Sie alternativ dazu den Befehl `reset` 160 mit den Optionen `-i` und `-u` aus.

4.1.8.5 Befehlszeilenschnittstelle (CLI)

Um den Schema-Manager über die Befehlszeile aufzurufen, müssen Sie den Pfad zur ausführbaren Datei kennen. Standardmäßig befindet sich die ausführbare Schema-Manager-Datei hier:

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

Anmerkung: Nachdem Sie auf Linux- und macOS-Systemen das Verzeichnis in dasjenige, das die ausführbare Datei enthält, geändert haben, können Sie die ausführbare Datei mit **`sudo ./xmlschemamanager`** aufrufen. Das Präfix **`./`** gibt an, dass sich die ausführbare Datei im aktuellen Verzeichnis befindet. Das Präfix **`sudo`** gibt an, dass der Befehl mit Root-Rechten ausgeführt werden muss.

Befehlszeilensyntax

Die allgemeine Syntax zur Verwendung der Befehlszeile lautet folgendermaßen:

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

Der senkrechte Balken **`|`** im Codefragment oben trennt eine Gruppe einander gegenseitig ausschließender Elemente. Optionale Elemente stehen innerhalb von eckigen Klammern **`[]`**. Im Prinzip können Sie den Pfad zur ausführbaren Datei, gefolgt von entweder `--h`, `--help` oder `--version`-Optionen oder gefolgt von einem Befehl eingeben. Jeder Befehl kann Optionen und Argumente haben. Die Liste der Befehle wird in den folgenden Abschnitten beschrieben.

 © 2019-2025 Altova GmbH

4.1.8.5.1 help

Mit diesem Befehl erhalten Sie Hilfe zu Befehlen zur ausführbaren Schema-Manager-Datei.

Syntax

```
<exec> help [Befehl]
```

`[Befehl]` ist hierbei ein optionales Argument zur Angabe jedes beliebigen gültigen Befehlsnamens.

Beachten Sie dazu Folgendes:

- Sie können die Hilfe zu einem Befehl auch durch Eingabe des Befehls, gefolgt von **`-h`** oder **`--help`** aufrufen, z.B: **`<exec> list -h`**
- Wenn Sie **`-h`** oder **`--help`** direkt nach dem Namen der ausführbaren Datei und vor einem Befehl eingeben, wird die allgemeine Hilfe (und nicht die Hilfe zu einem bestimmten Befehl) angezeigt, z.B: **`<exec> -h list`**

Beispiel

Mit dem folgenden Befehl wird Hilfe zum Befehl `list` angezeigt:

```
xmlschemamanager help list
```

4.1.8.5.2 info

Mit diesem Befehl werden ausführliche Informationen über die einzelnen als `Schema`-Argument angegebenen Schemas angezeigt. Darin enthalten sind Titel, Version, Beschreibung, Herausgeber der jeweils angegebenen Schemas und davon referenzierte Schemas sowie die Information, ob das Schema installiert ist oder nicht.

Syntax

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

- Das Argument **`Schema`** ist der Name eines Schemas oder Teil eines Schemanamens. (Die Paket-ID eines Schemas und detaillierte Informationen über ihren Installationsstatus erhalten Sie mit dem Befehl `list` 159.)
- Mit **`<exec> info -h`** können Sie die Hilfe zum Befehl anzeigen.

Beispiel

Mit dem folgenden Befehl werden Informationen über das jeweils neueste **`DocBook-DTD`**- und **`NITF`**-Schemas angezeigt.

```
xmlschemamanager info doc nitf
```

4.1.8.5.3 initialize

Mit diesem Befehl wird die Schema-Manager-Umgebung initialisiert. Sie erstellen damit ein Cache-Verzeichnis, in dem Informationen über alle Schemas lokal gespeichert werden. Die Initialisierung erfolgt automatisch bei der ersten Installation einer Schema-fähigen Altova-Applikation. Normalerweise muss dieser Befehl nicht ausgeführt werden. Nach Ausführung des **reset**-Befehls ist dies allerdings erforderlich.

Syntax

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

Optionen
Für den Befehl **initialize** stehen die folgenden Optionen zur Verfügung:

`--silent, --s`	Nur Fehlermeldungen anzeigen. Der Standardwert ist **false**.
`--verbose, --v`	Anzeige detaillierter Informationen während der Ausführung. Der Standardwert ist **false**.
`--help, --h`	Anzeige der Hilfe zum Befehl.

Beispiel

Mit dem folgenden Befehl wird der Schema-Manager initialisiert:

```
xmlschemamanager initialize
```

4.1.8.5.4 install

Mit diesem Befehl installieren Sie ein oder mehrere Schemas.

Syntax

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

Um mehrere Schemas zu installieren, fügen Sie das Argument **Schema** mehrmals hinzu.

Als **Schema**-Argument kann eines der folgenden verwendet werden:

- Ein Schema-Identifier (im Format **<name>-<version>**, z.B: **cbcr-2.0**). Um die Schema-Identifier der gewünschten Schemas zu eruieren, führen Sie den Befehl `list` 159 aus. Sie können auch einen abgekürzten Identifier verwenden, sofern dieser eindeutig ist, z.B. **docbook**. Falls Sie einen abgekürzten Identifier verwenden, wird die neueste Version dieses Schemas installiert.
- Der Pfad zu einer von der Altova-Website heruntergeladenen **.altova_xmlschemas**-Datei. Informationen zu diesen Dateien finden Sie in der *Einführung zu Schema-Manager: Funktionsweise* 147.

Optionen
Für den Befehl **install** stehen die folgenden Optionen zur Verfügung:

 © 2019-2025 Altova GmbH

`--silent, --s`	Nur Fehlermeldungen anzeigen. Der Standardwert ist **`false`**.
`--verbose, --v`	Anzeige detaillierter Informationen während der Ausführung. Der Standardwert ist **`false`**.
`--help, --h`	Anzeige der Hilfe zum Befehl.

Beispiel

Mit dem folgenden Befehl werden das CBCR 2.0 (Country-By-Country Reporting)-Schema und die neueste DocBook-DTD installiert:

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

4.1.8.5.5 list

Mit diesem Befehl werden vom Schema-Manager verwaltete Schemas aufgelistet. In der Liste wird eine der folgenden Informationen angezeigt:

- alle verfügbaren Schemas
- Schemas, die im Namen den im Argument **`Schema`** angegebenen String enthalten
- nur installierte Schemas
- Nur Schemas, für die ein Upgrade installiert werden kann

Syntax

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

Wenn kein **`Schema`**-Argument angegeben wird, werden alle verfügbaren Schemas aufgelistet. Andernfalls werden die durch die angegebenen Optionen definierten Schemas aufgelistet (*siehe Beispiel unten*). Beachten Sie, dass Sie das Argument **`Schema`** mehrfach angeben können.

Optionen
Für den Befehl **`list`** stehen die folgenden Optionen zur Verfügung:

`--installed, --i`	Auflisten nur der installierten Schemas. Der Standardwert ist **`false`**.
`--upgradeable, --u`	Auflisten nur derjenigen Schemas, für die Upgrades (Patches) zur Verfügung stehen. Der Standardwert ist **`false`**.
`--help, --h`	Anzeige der Hilfe zum Befehl.

Beispiele

- Um alle verfügbaren Schemas aufzulisten, führen Sie den folgenden Befehl aus: **`xmlschemamanager list`**
- Um nur installierte Schemas aufzulisten, führen Sie **`xmlschemamanager list -i`** aus.

- Um Schemas, die in ihrem Namen entweder "doc" oder "nitf" enthalten, aufzulisten, führen Sie **`xmlschemamanager list doc nitf`** aus.

4.1.8.5.6 reset

Mit diesem Befehl werden alle installierten Schemas und das Cache-Verzeichnis entfernt. Ihre Schemaumgebung wird vollständig zurückgesetzt. Nachdem Sie diesen Befehl ausgeführt haben, muss der Befehl `initialize` 158 ausgeführt werden, um das Cache-Verzeichnis neu zu erstellen. Führen Sie alternativ dazu den Befehl **`reset`** mit der Option **`-i`** aus. Da mit **`reset-i`** die Originalinstallation des Produkts wiederhergestellt wird, wird empfohlen, nach dem Zurücksetzen und Initialisieren auch den Befehl `update` 161 auszuführen. Führen Sie alternativ dazu den Befehl **`reset`** mit den Optionen **`-i`** und **`-u`** aus.

Syntax

```
<exec> reset [Optionen]
```

Optionen
Für den Befehl **`reset`** stehen die folgenden Optionen zur Verfügung:

`--init, --i`	Initialisierung des Schema-Managers nach dem Zurücksetzen. Der Standardwert ist **`false`**.
`--update, --u`	Aktualisiert die Liste der verfügbaren Schemas im Cache. Der Standardwert ist **`false`**.
`--silent, --s`	Nur Fehlermeldungen anzeigen. Der Standardwert ist **`false`**.
`--verbose, --v`	Anzeige detaillierter Informationen während der Ausführung. Der Standardwert ist **`false`**.
`--help, --h`	Anzeige der Hilfe zum Befehl.

Beispiele

- Um den Schema-Manager zurückzusetzen, führen Sie den folgenden Befehl aus: **`xmlschemamanager reset`**
- Um den Schema-Manager zurückzusetzen und ihn zu initialisieren, führen Sie **`xmlschemamanager reset -i`** aus.
- Um den Schema-Manager zurückzusetzen, ihn zu initialisieren und seine Schemaliste zu aktualisieren, führen Sie **`xmlschemamanager reset -i -u`** aus.

4.1.8.5.7 uninstall

Mit diesem Befehl deinstallieren Sie ein oder mehrere Schemas. Standardmäßig werden auch alle Schemas, die vom der aktuellen Schema referenziert werden, deinstalliert. Um nur das aktuelle Schema zu deinstallieren und die referenzierten Schemas beizubehalten, setzen Sie die Option `--k`.

 © 2019-2025 Altova GmbH

Syntax

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

Um mehrere Schemas zu deinstallieren, fügen Sie das Argument **Schema** mehrmals hinzu.

Als **Schema**-Argument kann eines der folgenden verwendet werden:

- Ein Schema-Identifier (im Format **<name>-<version>**, z.B: **cbcr-2.0**). Um die Schema-Identifier der installierten Schemas zu eruieren, führen Sie den Befehl `list -i` 159 aus. Sie können auch einen abgekürzten Schemanamen verwenden, sofern dieser eindeutig ist, z.B. **docbook**. Falls Sie einen abgekürzten Namen verwenden, werden alle Schemas, die die Abkürzung in ihrem Namen enthalten, deinstalliert.
- Der Pfad zu einer von der Altova-Website heruntergeladenen **.altova_xmlschemas**-Datei. Informationen zu diesen Dateien finden Sie in der *Einführung zu Schema-Manager: Funktionsweise* 147.

Optionen
Für den Befehl **uninstall** stehen die folgenden Optionen zur Verfügung:

`--keep-references, --k`	Definieren Sie diese Option, um referenzierte Schemas beizubehalten. Der Standardwert ist **false**.
`--silent, --s`	Nur Fehlermeldungen anzeigen. Der Standardwert ist **false**.
`--verbose, --v`	Anzeige detaillierter Informationen während der Ausführung. Der Standardwert ist **false**.
`--help, --h`	Anzeige der Hilfe zum Befehl.

Beispiel

Mit dem folgenden Befehl werden die Schemas CBCR 2.0 und EPUB 2.0 und deren Abhängigkeiten deinstalliert:

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

Mit dem folgenden Befehl wird das **eba-2.10**-Schema, nicht aber die davon referenzierten Schemas deinstalliert:

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

4.1.8.5.8 update

Mit diesem Befehl wird die Liste der über den Online-Speicher verfügbaren Schemas abgefragt und das lokale Cache-Verzeichnis wird aktualisiert. Normalerweise muss dieser Befehl nur ausgeführt werden, wenn Sie `reset` 160 und `initialize` 158 ausgeführt haben.

Syntax

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

Optionen
Für den Befehl **update** stehen die folgenden Optionen zur Verfügung:

`--silent, --s`	Nur Fehlermeldungen anzeigen. Der Standardwert ist **false**.
`--verbose, --v`	Anzeige detaillierter Informationen während der Ausführung. Der Standardwert ist **false**.
`--help, --h`	Anzeige der Hilfe zum Befehl.

Beispiel

Mit dem folgenden Befehl wird der lokale Cache mit der Liste der neuesten Schemas aktualisiert:

```
xmlschemamanager update
```

4.1.8.5.9 upgrade

Mit diesem Befehl werden alle installierten Schemas, für die ein Upgrade installiert werden kann, auf die neueste verfügbare *Patch*-Version aktualisiert. Um herauszufinden, welche Schemas aktualisiert werden können, starten Sie den Befehl `list-u` 159.

Anmerkung: Der Befehl **upgrade** entfernt ein veraltetes Schema, falls keine neuere Version zur Verfügung steht.

Syntax

```
<exec> upgrade [Optionen]
```

Optionen
Für den Befehl **upgrade** stehen die folgenden Optionen zur Verfügung:

`--silent, --s`	Nur Fehlermeldungen anzeigen. Der Standardwert ist **false**.
`--verbose, --v`	Anzeige detaillierter Informationen während der Ausführung. Der Standardwert ist **false**.
`--help, --h`	Anzeige der Hilfe zum Befehl.

 © 2019-2025 Altova GmbH

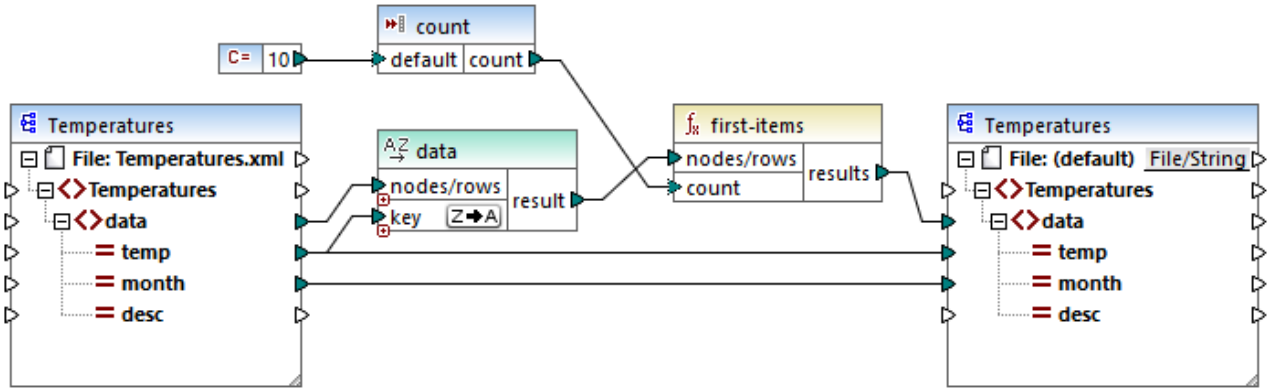
5 Transformationskomponenten

In diesem Abschnitt werden Transformationskomponenten zur Transformation von Daten oder zum temporären Speichern von Daten für die weitere Verarbeitung beschrieben. Im Folgenden finden Sie eine Liste von Transformationskomponenten:

- Einfache Input-Komponente 164
- Einfache Output-Komponente 171
- Variablen 175
- Sortieren von Komponenten 188
- Filter und Bedingungen 194
- Wertezuordnungen 200
- Funktionen 212

5.1 Einfache Input-Komponente

Wenn Sie ein Mapping erstellen müssen, das als Input Parameter erhält, können Sie das durch Hinzufügen einer speziellen Input-Komponente namens "einfache Input-Komponente" bewerkstelligen. Einfache Input-Komponenten haben immer einen einfachen Datentyp (z.B. String, Ganzzahl usw.) anstelle einer Struktur von Datenelementen und Sequenzen. So enthält etwa das Mapping unten die einfache Input-Komponente **count**. Sie hat die Aufgabe, die maximale Zeilenanzahl, die aus der XML-Quelldatei abgerufen werden soll (mit dem Wert **10** als Standardeinstellung), in Form eines Parameters bereitzustellen. Beachten Sie, dass die als Input für die Funktion first-items 301 bereitgestellten Nodes mit Hilfe einer Sortierkomponente sortiert werden, sodass im Mapping nur die höchsten *N* Temperaturen ausgegeben werden, wobei *N* der Wert des Parameters ist.



FindHighestTemperatures.mfd

Eine weitere häufige Einsatzmöglichkeit für einfache Input-Komponenten ist die Bereitstellung eines Dateinamens für das Mapping. Dies ist in Mappings, in denen Daten dynamisch aus Input-Dateien ausgelesen oder in Output-Dateien geschrieben werden sollen, hilfreich, siehe Dynamische Verarbeitung mehrerer Input- oder Output-Dateien 422. In der generierten XSLT-Datei entsprechen einfache Input-Komponenten Stylesheet-Parametern.

Sie können jede einzelne Input-Komponente (oder jeden Parameter) als optional oder obligatorisch erstellen (siehe Hinzufügen von einfachen Input-Komponenten 165). Bei Bedarf können Sie auch Standardwerte für die Input-Mapping-Parameter erstellen (siehe Erstellen eines Input-Standardwerts 167). Dadurch können Sie das Mapping ohne Probleme ausführen, selbst wenn Sie während der Ausführung des Mappings nicht explizit einen Parameterwert bereitstellen. Ein Beispiel dazu finden Sie unter Beispiel: Verwenden von Dateinamen als Mapping-Parameter 168.

Input-Parameter, die zum Mapping-Bereich hinzugefügt werden, sind nicht mit Input-Parametern in benutzerdefinierten Funktionen 219 zu verwechseln. Die beiden Parameterarten weisen die folgenden Ähnlichkeiten und Unterschiede auf:

Input-Parameter im Mapping-Bereich	Input-Parameter von benutzerdefinierten Funktionen
Werden über das Menü **Funktion | Input einfügen** hinzugefügt.	Werden über das Menü **Funktion | Input einfügen** hinzugefügt.

 © 2019-2025 Altova GmbH

Input-Parameter im Mapping-Bereich	Input-Parameter von benutzerdefinierten Funktionen
Können einfache Datentypen haben (String, Ganzzahl, usw.).	Können sowohl einfache als auch komplexe Datentypen haben.
Anwendbar auf das gesamte Mapping.	Können nur im Kontext der Funktion, in der sie definiert wurden, angewendet werden.

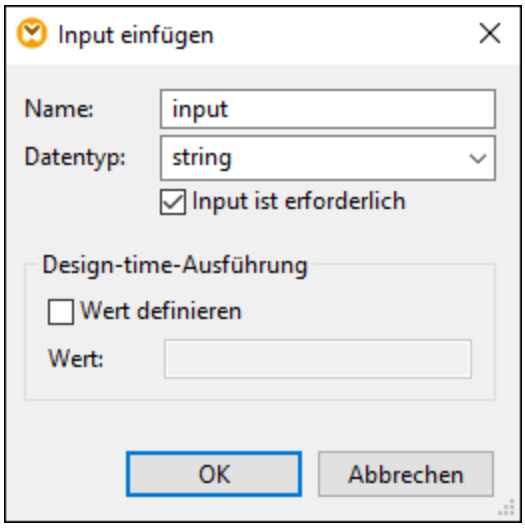
Wenn Sie (mit dem Menübefehl **Extras | Umgekehrtes Mapping erstellen**) ein umgekehrtes Mapping erstellen, wird eine einfache Input-Komponente zu einer einfachen Output-Komponente.

5.1.1 Hinzufügen von einfachen Input-Komponenten

So fügen Sie eine einfache Input-Komponente zum Mapping hinzu:

1. Stellen Sie sicher, dass im Mapping-Fenster das Hauptmapping (und keine benutzerdefinierte Funktion) angezeigt wird.
2. Wählen Sie eine der folgendem Methoden:

   - Klicken Sie im Menü **Funktion** auf **Input-Komponente einfügen**.
   - Klicken Sie im Menü **Einfügen** auf **Input-Komponente einfügen**.
   - Klicken Sie auf die Symbolleisten-Schaltfläche **Input-Komponente einfügen** .



3. Geben Sie einen Namen ein und wählen Sie den gewünschten Datentyp für diesen Input aus. Wenn der Input als zwingend erforderlicher Mapping-Parameter behandelt werden soll, aktivieren Sie das Kontrollkästchen **Input ist erforderlich**. Eine vollständige Liste der Einstellungen finden Sie unter Einstellungen für einfache Input-Komponenten 166.

Anmerkung: Der Parametername darf nur Buchstaben, Zahlen und Unterstrichzeichen enthalten. Andere Zeichen sind nicht zulässig. Dadurch funktioniert das Mapping in allen Codegenerierungssprachen.

4. Klicken Sie auf **OK.**

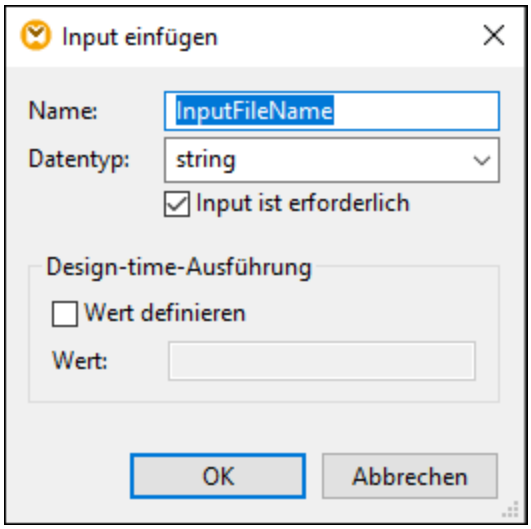
Sie können jede der hier definierten Einstellungen später ändern (siehe Einstellungen für einfache Input-Komponenten 166).

5.1.2 Einstellungen für einfache Input-Komponenten

Sie können die Einstellungen für eine einfache Input-Komponente beim Hinzufügen der Komponente zum Mapping-Bereich definieren. Sie können die Einstellungen später über das Dialogfeld "Input bearbeiten" jederzeit ändern.

Um das Dialogfeld "Input bearbeiten" zu öffnen, wählen Sie eine der folgenden Methoden:

- Wählen Sie die Komponente aus und klicken Sie im Menü **Komponente** auf **Eigenschaften**.
- Doppelklicken Sie auf die Komponente.
- Klicken Sie mit der rechten Maustaste auf die Komponente und wählen Sie den Befehl **Eigenschaften**.



Dialogfeld "Input bearbeiten"

Es stehen die folgenden Einstellungen zur Verfügung.

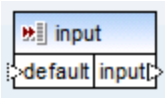
Name	Geben Sie einen beschreibenden Namen für den Input-Parameter, der dieser Komponente entspricht, ein. Zum Zeitpunkt der Mapping-Ausführung wird der in dieses Textfeld eingegebene Wert der Name des für das Mapping bereitgestellten Parameters; es sind daher keine Leer- oder Sonderzeichen zulässig.
Datentyp	Standardmäßig wird der Input-Parameter als String-Datentyp behandelt. Wenn der Parameter einen anderen Datentyp haben soll, wählen Sie den entsprechenden Wert aus der Liste aus. MapForce konvertiert den Input-Parameter bei der Ausführung des Mappings in den hier ausgewählten Datentyp.

 © 2019-2025 Altova GmbH

Input ist erforderlich	Wenn dieses Kontrollkästchen aktiviert ist, wird der Input-Parameter zu einem zwingend erforderlichen Parameter, d.h. das Mapping kann nur dann ausgeführt werden, wenn Sie einen Parameterwert angeben. Deaktivieren Sie dieses Kontrollkästchen, wenn Sie einen Standardwert für den Input-Parameter definieren möchten (siehe Erstellen eines Input-Standardwerts 167).
Wert definieren	Diese Einstellung wird nur angewendet, wenn Sie das Mapping während des Designs durch Klicken auf das Register **Vorschau** ausführen. Über diese Einstellung können Sie den als Mapping-Input zu verwendenden Wert direkt in der Komponente eingeben.
Wert	Diese Einstellung wird nur angewendet, wenn Sie das Mapping während des Designs durch Klicken auf das Register **Vorschau** ausführen. Um den gewünschten Wert für MapForce einzugeben, aktivieren Sie das Kontrollkästchen **Wert definieren** und geben Sie anschließend den gewünschten Wert ein. **Anmerkung:** Wenn Sie das Kontrollkästchen **Wert definieren** aktivieren und in das benachbarte Feld einen Wert eingeben, hat der eingegebene Wert Vorrang vor dem Standardwert, wenn Sie eine Vorschau auf das Mapping anzeigen (d.h. bei der Ausführung während des Designs). Der Design-Zeit-Wert hat jedoch im generierten XSLT-, XQuery- oder Programmcode, bei der Ausführung durch MapForce Server oder bei Bereitstellung auf FlowForce Server keine Auswirkung.

5.1.3 Erstellen eines Input-Standardwerts

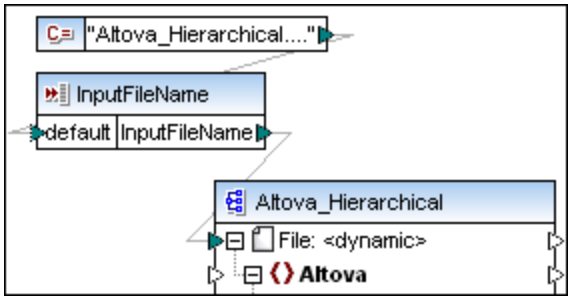
Beachten Sie nach Hinzufügen einer Input-Komponente zum Mapping-Bereich das Datenelement **default** links von der Komponente.



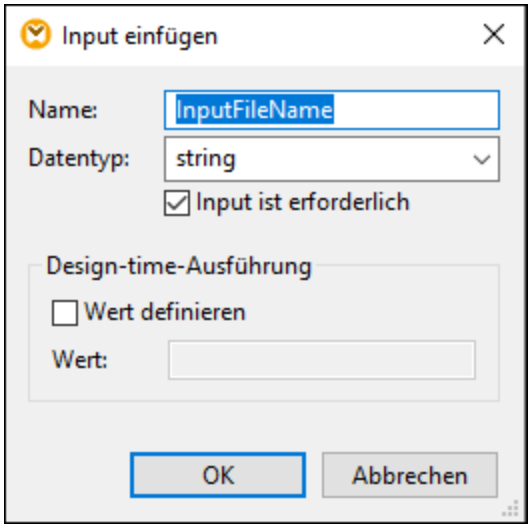
Einfache Input-Komponente

Über das Datenelement **default** können Sie wie folgt einen optionalen Standardwert zu dieser Input-Komponente hinzufügen:

1. Fügen Sie eine Konstantenkomponente hinzu (Menü **Einfügen | Konstante)** und verbinden Sie diese mit dem Datenelement **default** der Input-Komponente.



2. Doppelklicken Sie auf die Input-Komponente und deaktivieren Sie das Kontrollkästchen **Input ist erforderlich**. Wenn Sie einen Input-Standardwert erstellen, hat diese Einstellung keine Bedeutung und verursacht Mapping-Validierungswarnungen.



3. Klicken Sie auf **OK.**

Anmerkung: Wenn Sie das Kontrollkästchen **Wert definieren** aktivieren und in das benachbarte Feld einen Wert eingeben, hat der eingegebene Wert Vorrang vor dem Standardwert, wenn Sie eine Vorschau auf das Mapping anzeigen (d.h. bei der Ausführung während des Designs). Der Design-Zeit-Wert hat jedoch im generierten XSLT-, XQuery- oder Programmcode, bei der Ausführung durch MapForce Server oder bei Bereitstellung auf FlowForce Server keine Auswirkung.

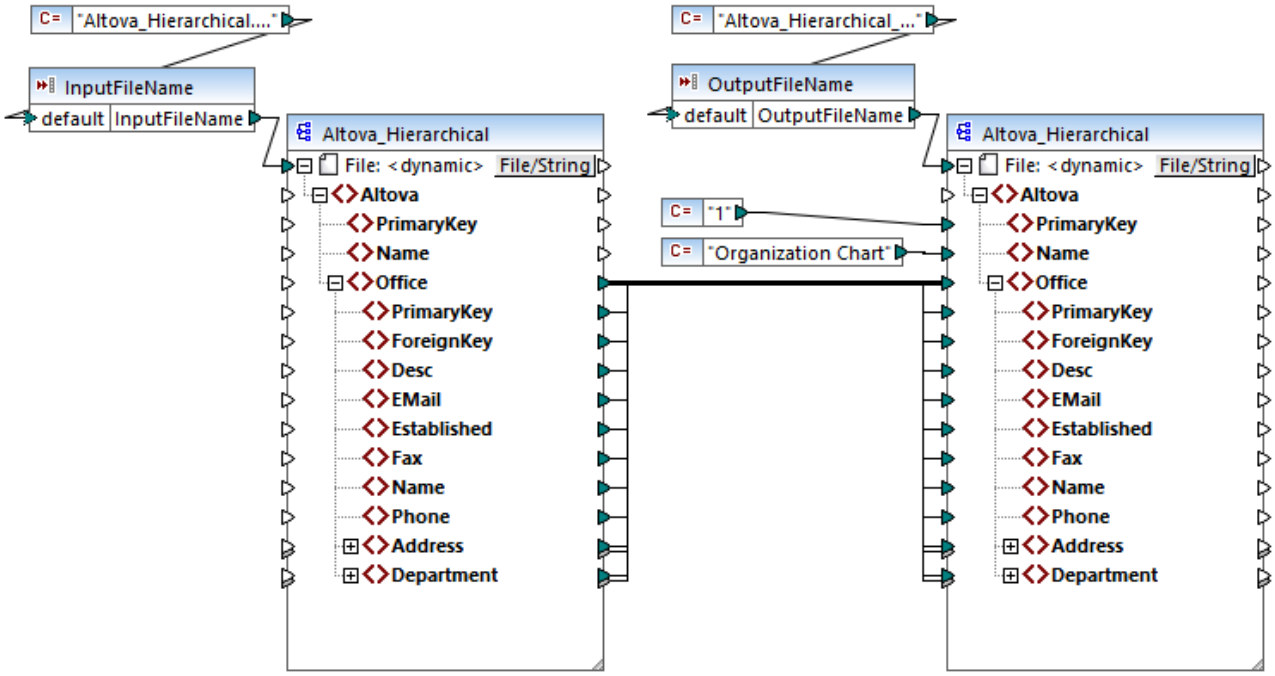
5.1.4 Beispiel: Verwenden von Dateinamen als Mapping-Parameter

In diesem Beispiel wird Schritt für Schritt beschrieben, wie Sie ein Mapping ausführen, das zur Laufzeit Input-Parameter erhält. Sie finden die in diesem Beispiel verwendete Mapping-Design-Datei unter dem Pfad: **<Dokumente>\Altova\MapForce2025\MapForceExamples\FileNamesAsParameters.mfd**.

In diesem Mapping werden Daten aus einer XML-Quelldatei ausgelesen und in eine XML-Zieldatei geschrieben. Die Daten werden beinahe unverändert in die Zieldatei geschrieben; nur die Attribute **PrimaryKey** und **Name** werden mit Konstantenwerten aus dem Mapping befüllt. Hauptaufgabe des Mappings ist, dem Aufrufenden eine Möglichkeit zu geben, die Namen der Input- und Output-Datei zur Mapping-Laufzeit in Form von Mapping-Parametern anzugeben.

Das Mapping hat zu diesem Zweck zwei Input-Komponenten: **InputFileName** und **OutputFileName**. Diese Komponenten stellen den Input-Dateinamen (bzw. den Output-Dateinamen) der XML-Quell- und der XML-Zieldatei bereit. Aus diesem Grund wurden Sie mit dem Datenelement **Datei:<dynamisch>** verbunden. Durch Klicken auf die Schaltfläche **Datei** (Datei) und Auswahl der Option **Über das Mapping bereitgestellte dynamische Dateinamen verwenden** können Sie eine Komponente in diesen Modus schalten.

 © 2019-2025 Altova GmbH



FileNamesAsParameters.mfd (MapForce Enterprise Edition)

Wenn Sie auf die Titelleiste einer der Komponenten (**InputFileName** oder **OutputFileName**) doppelklicken, sehen Sie deren Eigenschaften. So können Sie etwa den Datentyp des Input-Parameters definieren oder den Namen des Input-Parameters ändern, wie unter Einstellungen für einfache Input-Komponenten 166 beschrieben. In diesem Beispiel sind die Input- und Output-Parameter folgendermaßen konfiguriert:

- Der Parameter **InputFileName** hat den Typ "string" und hat einen Standardwert, der von einer im selben Mapping definierten Konstante stammt. Die Konstante hat den Typ "string" und den Wert "Altova_Hierarchical.xml". Bei Ausführung dieses Mappings wird daher versucht, Daten aus einer Datei namens "Altova_Hierarchical.xml" zu lesen, vorausgesetzt Sie geben keinen anderen Wert als Parameter an.
- Der Parameter **OutputFileName** hat den Typ "string" und hat einen Standardwert, der von einer im selben Mapping definierten Konstante stammt. Die Konstante hat den Typ "string" und den Wert "Altova_Hierarchical_output.xml". Das Mapping erstellt daher bei seiner Ausführung eine XML-Ausgabedatei namens "Altova_Hierarchical_output.xml", vorausgesetzt Sie geben keinen anderen Wert als Parameter an.

In den folgenden Abschnitten wird beschrieben, wie Sie das Mapping ausführen und in den folgenden Transformationssprachen Parameter bereitstellen:

- XSLT 2.0 169 mit Hilfe von RaptorXML Server

XSLT 2.0

Wenn Sie Code in XSLT 1.0, XSLT 2.0 oder XSLT 3.0 generieren, wird im gewählten Zielverzeichnis zusätzlich zur XSLT-Datei die Batch-Datei **DoTransform.bat** generiert. Mit Hilfe von **DoTransform.bat** können Sie das Mapping mit RaptorXML Server ausführen, siehe Automatisierung mit RaptorXML Server 428.

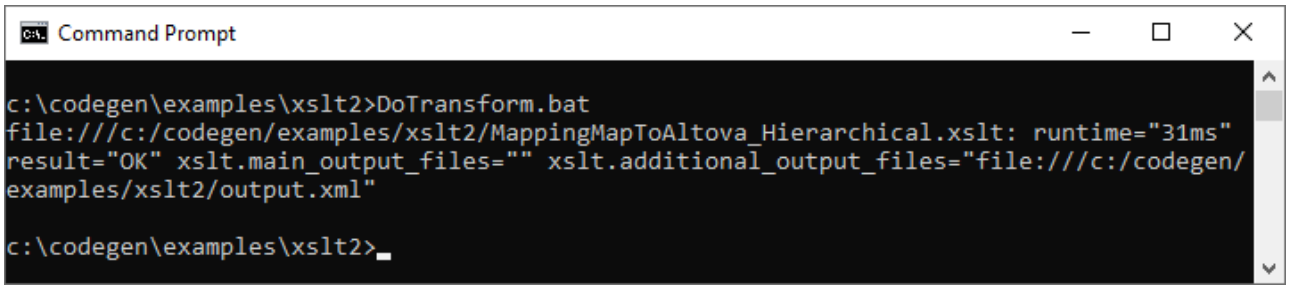
Um eine andere Input (oder Output)-Datei zu verwenden, bearbeiten Sie die Datei **DoTransform.bat**, damit Sie die erforderlichen Parameter enthält. Gehen Sie dazu folgendermaßen vor:

1. Generieren Sie zuerst den XSLT-Code (Um z.B. XSLT 2.0 zu generieren, klicken Sie im Menü **Datei** auf **Code generieren in | XSLT 2.0**).
2. Kopieren Sie die Datei **Altova_Hierarchical.xml** aus **<Dokumente>\Altova\MapForce2025\MapForceExamples** in das Verzeichnis, in dem Sie XSLT 2.0-Code generiert haben (in diesem Beispiel **c:\codegen\examples\xslt2**). Wie zuvor erwähnt, versucht das Mapping diese Datei zu lesen, wenn Sie keinen benutzerdefinierten Wert für den Parameter **InputFileName** bereitstellen.
3. Bearbeiten Sie **DoTransform.bat,** sodass diese den benutzerdefinierten Input-Parameter entweder vor oder nach `%*` enthält. Beachten Sie, dass der Parameterwert innerhalb von einfache Anführungszeichen gesetzt wird. Die verfügbaren Input-Parameter werden im Abschnitt **rem** (Remark) aufgelistet. Angenommen, Sie möchten eine Output-Datei mit dem Namen **output.xml** generieren. Ändern Sie dazu die Datei **DoTransform.bat** folgendermaßen:

```
@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%
```

Wenn Sie die Datei **DoTransform.bat** ausführen, stellt RaptorXML Server die Transformation mit Hilfe von **Altova_Hierarchical.xml** als Input fertig. Wenn Sie die obige Anleitung befolgt haben, erhält die generierte Ausgabedatei den Namen **output.xml**.



 © 2019-2025 Altova GmbH

5.2 Einfache Output-Komponente

Eine Output-Komponente (oder "einfacher Output") ist eine MapForce Komponente, über die ein String-Wert aus dem Mapping zurückgegeben wird. Output-Komponenten sind nur eine mögliche Art von Zielkomponenten29, sind aber nicht damit zu verwechseln. Verwenden Sie eine einfache Output-Komponente, wenn Sie einen String-Wert anhand des Mappings zurückgeben möchten. Einfache Output-Komponenten spielen im Mapping-Bereich die Rolle einer Zielkomponente mit einem String-Datentyp anstelle einer Struktur von Datenelementen und Sequenzen. Infolgedessen können Sie anstelle (oder zusätzlich zu) einer dateibasierten Zielkomponente eine einfache Output-Komponente erstellen. So können Sie eine einfache Output-Komponente z.B. verwenden, um die Ausgabe einer Funktion schnell zu testen und eine Vorschau des Ergebnisses anzuzeigen (siehe Beispiel: Testen der Funktionsausgabe173).

Einfache Output-Komponenten sollten nicht mit Output-Parametern in benutzerdefinierten Funktionen verwechselt werden (siehe Benutzerdefinierte Funktionen219). Die beiden Parameterarten weisen die folgenden Ähnlichkeiten und Unterschiede auf:

Output-Komponenten	Output-Parameter von benutzerdefinierten Funktionen
Werden über das Menü **Funktion | Output-Komponente einfügen** hinzugefügt.	Werden über das Menü **Funktion | Output-Komponente einfügen** hinzugefügt.
Haben "string" als Datentyp.	Können sowohl einfache als auch komplexe Datentypen haben.
Anwendbar auf das gesamte Mapping.	Können nur im Kontext der Funktion, in der sie definiert wurden, angewendet werden.

Bei Bedarf können Sie mehrere einfache Output-Komponenten zu einem Mapping hinzufügen. Sie können einfache Output-Komponenten auch in Kombination mit dateibasierten Zielkomponenten verwenden. Wenn Ihr Mapping mehrere Zielkomponenten enthält, können Sie eine Vorschau der von einer bestimmten Komponenten zurückgegebenen Daten anzeigen, indem Sie in der Titelleiste der Komponente auf die Schaltfläche **Vorschau** ( ) klicken und anschließend im Mapping-Fenster auf das Fenster **Ausgabe** klicken.

Sie können einfache Output-Komponenten folgendermaßen in MapForce-Transformationssprachen verwenden:

Sprache	Funktionsweise
XSLT 1.0, XSLT 2.0, XSLT 3.0	Wenn Sie XSLT-Dateien generieren, wird eine im Mapping definierte einfache Output-Komponente zur Ausgabe der XSLT-Transformation. Wenn Sie RaptorXML Server verwenden, können Sie RaptorXML Server anweisen, die Mapping-Ausgabe in die Datei zu schreiben, die als Wert an den Parameter `--output` übergeben wird. Um die Ausgabe in eine Datei zu schreiben, fügen Sie in der Datei **DoTransform.bat** den Parameter `--output` hinzu oder bearbeiten Sie ihn. Die folgende Datei **DoTransform.bat** wurde z.B. so bearbeitet, dass sie die Mapping-Ausgabe in die Datei **Output.txt** schreibt (siehe markierter Text).

Sprache	Funktionsweise
	`RaptorXML xslt --xslt-version=2 --input="MappingMapToResult1.xslt" --output="Output.txt" %* "MappingMapToResult1.xslt"` Wenn kein `--output` Parameter definiert ist, wird die Mapping-Ausgabe bei Ausführung des Mappings in den Output-Standard-Stream (stdout) geschrieben.

Wenn Sie (mit dem Menübefehl **Extras | Umgekehrtes Mapping erstellen**) ein umgekehrtes Mapping erstellen, wird eine einfache Output-Komponente zu einer einfachen Input-Komponente.

5.2.1 Hinzufügen einfacher Output-Komponenten

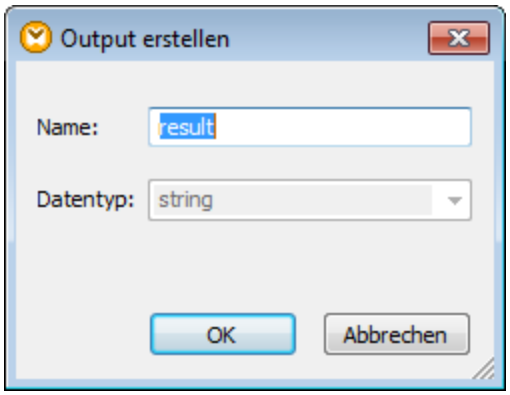
So fügen Sie eine Output-Komponente zum Mapping-Bereich hinzu:

1. Stellen Sie sicher, dass im Mapping-Fenster das Hauptmapping (und keine benutzerdefinierte Funktion) angezeigt wird.
2. Wählen Sie eine der folgenden Methoden:

   a. Klicken Sie im Menü **Funktion** auf **Output-Komponente einfügen**.

   b. Klicken Sie auf die Symbolleisten-Schaltfläche **Output-Komponente einfügen** .

3. Geben Sie einen Namen für die Komponente ein.
4. Klicken Sie auf **OK.**



Dialogfeld "Output erstellen"

Sie können den Komponentennamen später auf eine der folgenden Arten ändern:

- Wählen Sie die Komponente aus und klicken Sie im Menü **Komponente** auf **Eigenschaften**.
- Doppelklicken Sie auf den Komponententitel.
- Klicken Sie mit der rechten Maustaste auf den Komponententitel und wählen Sie **Eigenschaften**.

 © 2019-2025 Altova GmbH

5.2.2 Beispiel: Vorschau auf die Funktionsausgabe

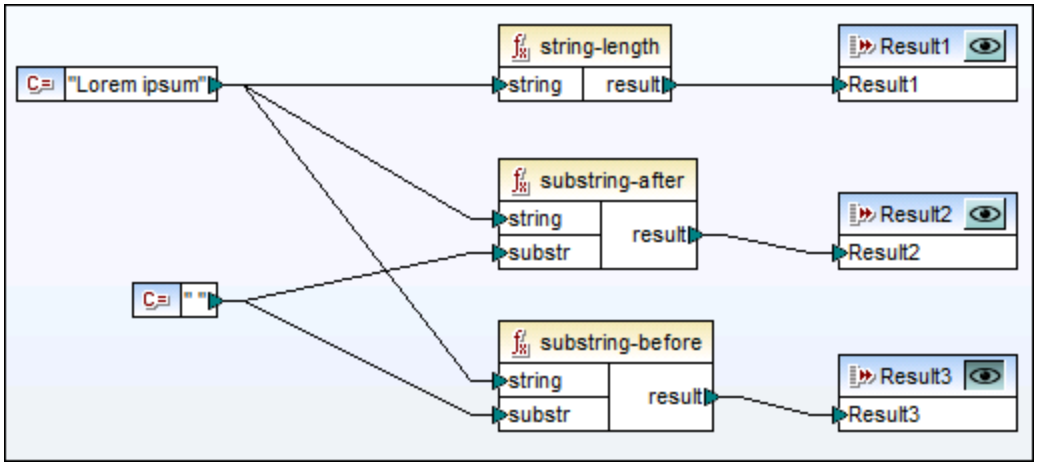
In diesem Beispiel wird gezeigt, wie Sie mit Hilfe einfacher Output-Komponenten eine Vorschau der von MapForce-Funktionen zurückgegebenen Ausgabe anzeigen können. Sie sollten bereits über ein grundlegendes Verständnis über Funktionen im Allgemeinen und MapForce-Funktionen im Besonderen verfügen. Wenn MapForce-Funktionen neu für Sie sind, lesen Sie bitte zuerst den Abschnitt Verwendung von Funktionen 212, bevor Sie fortfahren..

Wir wollen in diesem Beispiel eine Reihe von Funktionen zum Mapping-Bereich hinzufügen und lernen, wie man eine Vorschau ihrer Ausgabe mit Hilfe von einfachen Output-Komponenten anzeigt. In diesem Beispiel werden einige einfache Funktionen aus der core-Bibliothek verwendet. Hier sehen Sie eine Zusammenfassung ihrer Verwendung:

`string-length` 332	Gibt die Anzahl der Zeichen in dem als Argument angegebenen String zurück. Wenn Sie z.B. den Wert "Lorem ipsum" an diese Funktion übergeben, so ist das Ergebnis "11", da dies die Anzahl der Zeichen im Text "Lorem ipsum" ist.
`substring-after` 333	Gibt den Teil des String zurück, der hinter dem als Argument angegebenen Trennzeichen steht. Wenn Sie z.B. den Wert "Lorem ipsum" und das Trennzeichen (" ") an diese Funktion übergeben, ist das Ergebnis "ipsum".
`substring-before` 334	Gibt den Teil des String zurück, der vor dem als Argument angegebenen Trennzeichen steht. Wenn Sie z.B. den Wert "Lorem ipsum" und das Trennzeichen (" ") an diese Funktion übergeben, ist das Ergebnis "Lorem".

Um jede dieser Funktionen anhand eines benutzerdefinierten Textwerts (in diesem Beispiel "Lorem ipsum") zu testen, gehen Sie folgendermaßen vor:

1. Fügen Sie (mit dem Menübefehl **Einfügen | Konstante**) eine Konstante mit dem Wert "Lorem ipsum" zum Mapping-Bereich hinzu. Die Konstante bildet den Input-Parameter für jede der zu testenden Funktionen.
2. Fügen Sie die Funktionen **`string-length`**, **`substring-after`** und **`substring-before`** zum Mapping-Bereich hinzu, indem Sie sie aus der core-Bibliothek, Abschnitt **string functions,** in den Mapping-Bereich ziehen.
3. Fügen Sie eine Konstante mit einem Leerzeichen (" ") als Wert hinzu. Dieses Leerzeichen dient als Trennzeichenparameter für die Funktionen **`substring-after`** und **`substring-before`**.
4. Fügen Sie (mit dem Menübefehl **Funktion | Output-Komponente einfügen)** drei einfache Output-Komponenten hinzu. In diesem Beispiel haben wir die Komponenten *Result1*, *Result2* und *Result3* genannt, Sie können aber auch einen anderen Namen wählen.
5. Verbinden Sie die Komponenten wie unten gezeigt.



Testen der Funktionsausgabe mit Hilfe von einfachen Output-Komponenten

Wie im Beispiel oben gezeigt, wird der String "Lorem ipsum" als Input-Parameter für die Funktionen **`string-length`**, **`substring-after`** und **`substring-before`** verwendet. Zusätzlich dazu erhalten die Funktionen **`substring-after`** und **`substring-before`** einen Leerzeichenwert als zweiten Input-Parameter. Mit Hilfe der Komponenten *Result1*, *Result2* und *Result3* können Sie eine Vorschau des Ergebnisses jeder Funktion anzeigen.

So zeigen Sie eine Vorschau der Ausgabe einer Funktion an:

- Klicken Sie in der Titelleiste der Komponente auf die Schaltfläche **Vorschau** ( ) und klicken Sie anschließend im Mapping-Fenster auf das Fenster **Ausgabe**.

 © 2019-2025 Altova GmbH

5.3 Variablen

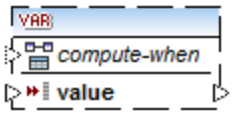
Bei Variablen handelt es sich um eine spezielle Art von Komponente, in der ein Mapping-Zwischenergebnis für die weitere Verarbeitung gespeichert wird. Variablen können einen simpleType (z.B. String, Ganzzahl, Boolsche Werte, usw.) oder complexType (eine Baumstruktur) haben. Beispiele für diese beiden Arten finden Sie in den Unterabschnitten weiter unten.

Einer der wichtigsten Aspekte von Variablen ist, dass es sich dabei um Sequenzen handelt und dass sie zum Erstellen von Sequenzen verwendet werden können. Mit dem Begriff *Sequenz* wird eine Liste von null oder mehr Datenelementen bezeichnet. Dadurch können mit Hilfe einer Variable mehrere Datenelemente im Laufe des Mappings verarbeitet werden. Nähere Informationen dazu finden Sie auch unter Mapping-Regeln und Strategien 77. Sie können einer Variablen auch ein Mal einen Wert zuweisen und diesen Wert im restlichen Mapping beibehalten. Nähere Informationen dazu finden Sie unter Ändern von Kontext und Geltungsbereich von Variablen 181.

Nähere Informationen zum Hinzufügen einer Variablen finden Sie unter Hinzufügen einer Variablen 177.

Einfache Variablen

Eine einfache Variable dient zur Darstellung eines atomaren Typs wie z.B. Strings, Zahlen und Booleschen Werten (*siehe Abbildung unten*).



Komplexe Variablen

Eine komplexe Variable hat eine Baumstruktur. In der Liste unten sehen Sie, auf welchen Strukturen eine komplexe Variable basieren kann.

MapForce Basic Edition:

- XML-Schema-Struktur

MapForce Professional Edition:

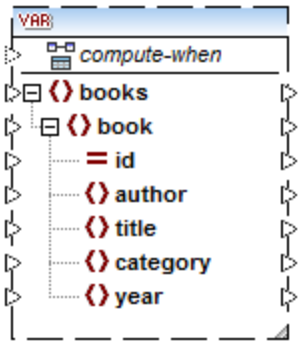
- XML-Schema-Struktur
- Datenbankstruktur

MapForce Enterprise Edition:

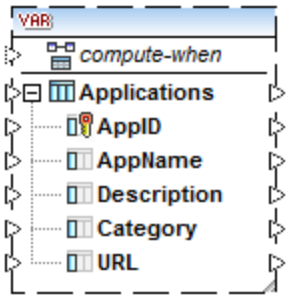
- XML-Schema-Struktur
- Datenbankstruktur
- EDI-Struktur
- FlexText Structure
- JSON-Schema-Struktur

Beispiel 1: Auf einem XML-Schema basierende Variable

Sie können eine Variable vom Typ complexType erstellen, indem Sie ein XML-Schema bereitstellen, das die Struktur der Variablen beschreibt (*siehe Abbildung unten*). Wenn das Schema globale Elemente enthält, können Sie auswählen, welches davon den Root-Node der Variablenstruktur bilden soll. Beachten Sie, dass mit einer Variablen keine XML-Instanzdatei verknüpft ist. Die Daten der Variablen werden zur Mapping-Laufzeit berechnet.



Beispiel 2: Auf einer Datenbank basierende Variable (MapForce Professional und Enterprise Edition)
Wenn Sie eine Datenbankstruktur für Ihre Variable auswählen (*siehe Abbildung unten*), können Sie eine bestimmte Datenbanktabelle als Root-Datenelement für die Variablenstruktur auswählen. Sie können in MapForce datenbankbasierte Variablen mit einer Struktur damit in Zusammenhang stehender Tabellen erstellen.



Die Struktur dieser Tabellen bildet eine speicherresidente Struktur, die keine Verbindung zur Datenbank zur Laufzeit hat. Die Variable beschreibt nur die Struktur, macht aber nichts mit den Daten. Daraus ergibt sich Folgendes:

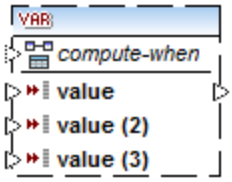
- Keine Validierung auf Basis des Datentyps
- Keine Möglichkeit der Verwendung des SQL WHERE-Filters
- Keine Tabellenaktionen in Parametern oder Variablen
- Keine automatische Behandlung von Sekundärschlüsseln
- Keine Primär-/Sekundärschlüsselüberprüfung

Compute-when
In beiden Beispielen oben hat jede Variable ein Datenelement namens `compute-when`. Die Verbindung mit diesem Datenelement ist optional. Dadurch können Sie festlegen, wie der Variablenwert im Mapping berechnet werden soll. Nähere Informationen dazu finden Sie unter Ändern von Kontext und Geltungsbereich von Variablen 181.

Variablen mit duplizierten Inputs

Falls nötig, können Datenelemente einer Variablenstruktur dupliziert werden, damit Daten aus mehr als einer Quellverbindung verbunden werden können. Dies ist ähnlich dem Duplizieren von Inputs 39 in Standardverbindungen. In der Abbildung unten sehen Sie eine einfache Variable mit duplizierten Inputs.

 © 2019-2025 Altova GmbH



Verkettete Mappings im Gegensatz zu Variablen

Variablen können mit Zwischenkomponenten eines verketteten Mappings 111 verglichen werden. Sie sind jedoch flexibler und praktischer, wenn in den einzelnen Phasen des Mappings keine Zwischendateien erzeugt werden müssen. In der folgenden Tabelle wurden die Unterschiede zwischen Variablen und verketteten Mappings gegenübergestellt.

Verkettete Mappings	Variablen
Verkettete Mappings bestehen aus zwei voneinander unabhängigen Schritten. So kann ein Mapping z.B. aus den drei Komponenten A, B, und C bestehen. Schritt 1: Mappen der Daten A auf die Daten B. Schritt 2: Mappen der Daten von B auf C.	Sie können steuern, wann und wie oft der Variablenwert bei der Ausführung des Mappings berechnet wird. Nähere Informationen dazu finden Sie unter Ändern von Kontext und Geltungsbereich von Variablen 181.
Wenn das Mapping ausgeführt wird, werden die Zwischenergebnisse extern in Dateien gespeichert.	Wenn das Mapping ausgeführt wird, werden die Zwischenergebnisse intern gespeichert. Es werden keine externen Dateien, die die Ergebnisse einer Variablen enthalten, erzeugt.
Das Zwischenergebnis kann über die Vorschau-Schaltfläche in einer Vorschau angezeigt werden.	Sie können keine Vorschau auf das Ergebnis einer Variablen anzeigen, da diese zur Mapping-Laufzeit berechnet wird.

Anmerkung: Variablen werden nicht unterstützt, wenn als Mapping-Transformationssprache XSLT 1.0 ausgewählt ist.

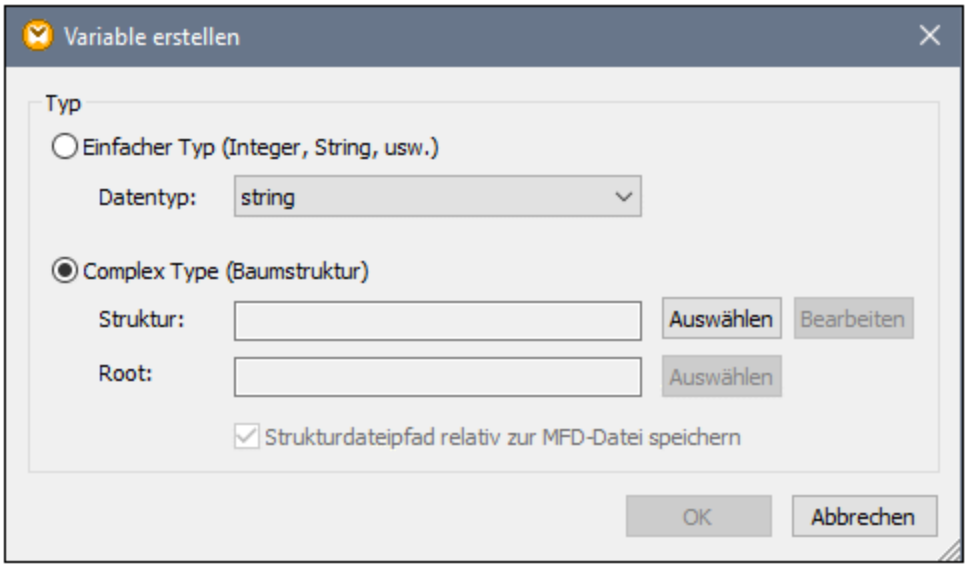
5.3.1 Hinzufügen einer Variablen

In diesem Kapitel wird erklärt, wie Sie eine Variable zu einem Mapping hinzufügen. Die erste Option ist, eine Variable über das Menü oder einen Symbolleistenbefehl hinzuzufügen. Bei der zweiten Option können Sie eine Variable über das Kontextmenü hinzufügen.

Option 1: über das Menü oder einen Symbolleistenbefehl

Mit Hilfe dieser Option können Sie eine Variable über das Menü oder einen Symbolleistenbefehl hinzuzufügen. Gehen Sie folgendermaßen vor:

1. Gehen Sie zum Menü **Datei** und klicken Sie auf **Variable**. Klicken Sie alternativ dazu auf die Symbolleisten-Schaltfläche (**Variable**).

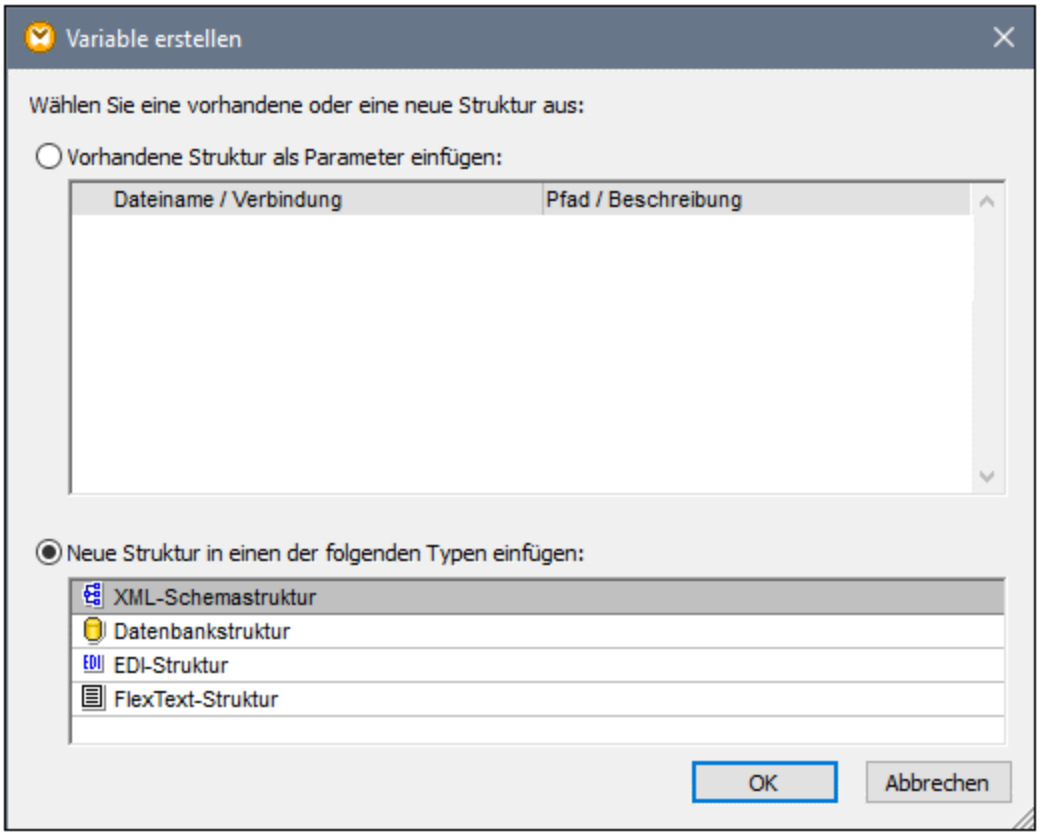


2. Wählen Sie den Typ der gewünschten Variable aus (einfacher Typ oder complex Type).

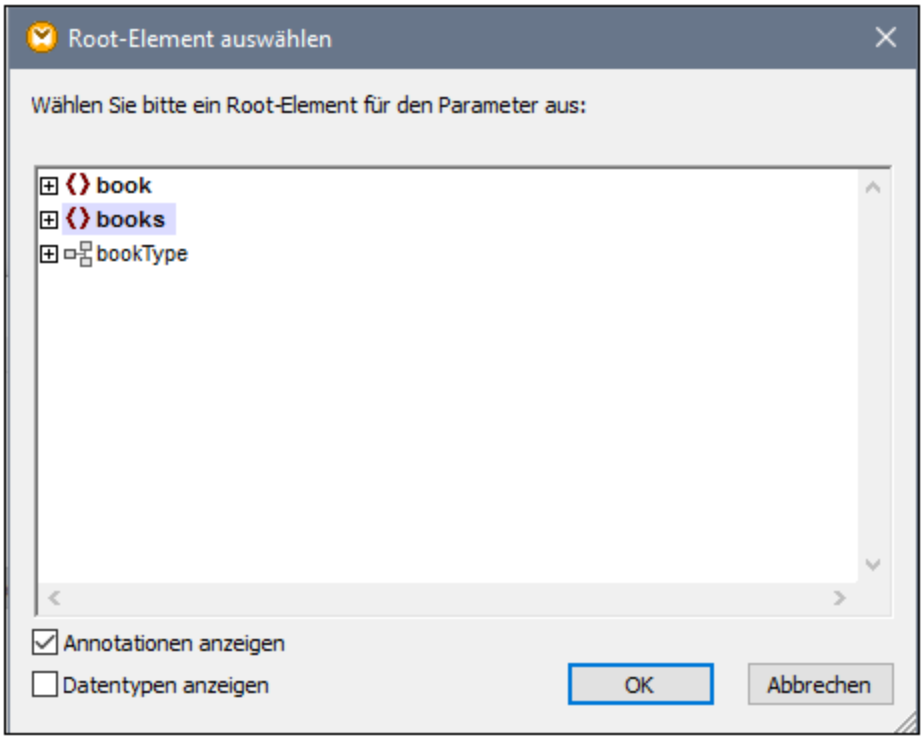
Bei Auswahl von **Complex Type** müssen einige zusätzliche Schritte durchgeführt werden:

3. Klicken Sie auf **Auswählen**, um die Quelldatei für die Struktur der Variablen 175 auszuwählen. Die Strukturen in der Abbildung unten beziehen sich nur auf die MapForce Enterprise Edition. Eine Liste der für andere MapForce Editionen relevanten Strukturen finden Sie im vorhergehenden Kapitel 175.

 © 2019-2025 Altova GmbH



4. Definieren Sie das Root-Datenelement für die Struktur der Variablen, wenn Sie danach gefragt werden. So können Sie z.B. in XML-Schemas jedes beliebige Element oder jeden beliebigen Typ aus der ausgewählten Quelldatei auswählen (*siehe Abbildung unten*).

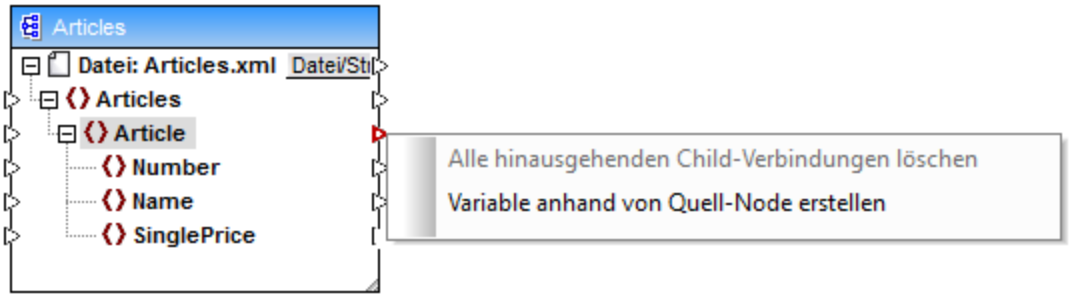


Option 2: über das Kontextmenü

Bei der zweiten Option können Sie eine Variable über das Kontextmenü erstellen. Unten finden Sie eine Liste der möglichen Optionen.

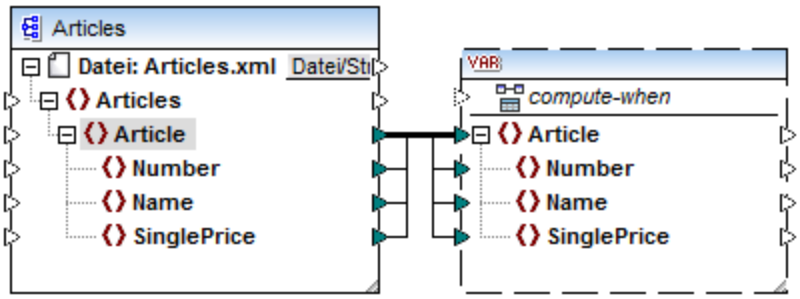
Variable anhand eines Quell-Node

Um anhand eines Quell-Node eine Variable zu erstellen, klicken Sie mit der rechten Maustaste auf den Output-Konnektor einer Komponente (in diesem Beispiel den Output-Konnektor des Elements `<Article>`) und wählen Sie den Befehl **Variable anhand von Quell-Node erstellen** (*siehe Abbildung unten*).



Daraufhin wird eine komplexe Variable mit dem Quellschema der Komponente `Articles` erstellt. Alle Datenelemente werden automatisch mittels einer "Alles kopieren"-Verbindung [54] verbunden (*siehe Abbildung unten*).

 © 2019-2025 Altova GmbH

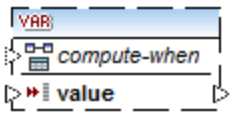


Variable anhand eines Ziel-Node
Um eine Variable anhand eines Ziel-Node zu erstellen, klicken Sie mit der rechten Maustaste auf den Input-Konnektor einer Zielkomponente und wählen Sie **Variable für Ziel-Node erstellen**. Daraufhin wird eine komplexe Variable mit demselben Schema, das auch in der Zielkomponente verwendet wird, erstellt. Alle Datenelemente werden automatisch mittels einer "Alles kopieren"-Verbindung verbunden.

Variable anhand eines Filters:
Um eine Variable mit Hilfe eines Filters zu erstellen, klicken Sie mit der rechten Maustaste auf den Output-Konnektor einer Filterkomponente (`on-true/on-false`) und wählen Sie den Befehl **Variable anhand von Quell-Node erstellen**. Daraufhin wird anhand des Quellschemas eine komplexe Komponente erstellt, wobei das mit dem Filter-Input verbundene Datenelement automatisch als Root-Element der Zwischenkomponente verwendet wird.

5.3.2 Geltungsbereich und Kontext von Variablen

Jede Variable hat ein `compute-when` Input-Datenelement (*siehe Abbildung unten*), mit dem Sie den Geltungsbereich der Variablen festlegen können, d.h. Sie können damit festlegen, wann und wie oft der Wert der Variablen bei der Ausführung des Mappings berechnet werden soll. Sie müssen diesen Input in vielen Fällen nicht verbinden, doch ist er manchmal wichtig, um den Standardkontext außer Kraft zu setzen oder die Mapping-Leistung zu optimieren.



In der Erläuterung von Geltungsbereich und Kontext von Variablen werden die folgenden Begriffe verwendet: *Substruktur* und *Variablenwert*. Als Substruktur wird eine Gruppe eines Datenelements/Node in einer Zielkomponente und aller seiner Nachfahren bezeichnet, z.B. ein Element `<Person>` mit seinen Child-Elementen `<FirstName>` und `<LastName>`.

Als Variablenwert werden die auf der Output-Seite der Variablenkomponente verfügbaren Daten bezeichnet.

- Bei einfachen Variablen handelt es sich um eine Sequenz atomarer Werte, die den in den Komponenteneigenschaften definierten Datentyp haben.

- Bei komplexen Variablen handelt es sich um eine Sequenz von Root Nodes (des in den Komponenteneigenschaften definierten Typs), einschließlich aller Nachfahren-Nodes der einzelnen Root Nodes.

Die Sequenz der atomaren Werte (oder Nodes) kann ein oder auch null Elemente enthalten. Dies ist abhängig davon, was mit der Input-Seite der Variablen und etwaigen übergeordneten Datenelementen in der Quell- und Zielkomponente verbunden ist.

"Compute-when" ist nicht verbunden (Standardeinstellung)

Wenn das `compute-when`-Input-Datenelement nicht (mit einem Output Node einer Quellkomponente) verbunden ist, wird der Variablenwert berechnet, *sobald er das erste Mal in einer Zielsubstruktur* (direkt über einen Konnektor von der Variablenkomponente zu einem Node in der Zielkomponente oder indirekt über Funktionen) verwendet wird. Derselbe Variablenwert wird auch für alle Child-Ziel-Nodes innerhalb der Substruktur verwendet.

Der tatsächliche Wert der Variablen hängt von etwaigen Verbindungen zwischen übergeordneten Datenelementen der Quell- und Zielkomponente ab. Dieses Standardverhalten ist dasselbe wie das von "complex" Outputs von regulären benutzerdefinierten Funktionen 222 und Webservice-Funktionsaufrufen. Wenn die Variable mit mehreren nicht miteinander in Zusammenhang stehenden Ziel-Nodes verbunden ist, wird der Wert der Variablen *für jeden davon separat berechnet*. Dabei können in jedem Fall unterschiedliche Ergebnisse erzeugt werden, da unterschiedliche Parent-Verbindungen sich auf den Kontext auswirken, in dem der Wert der Variablen ausgewertet wird.

"Compute-when" ist verbunden

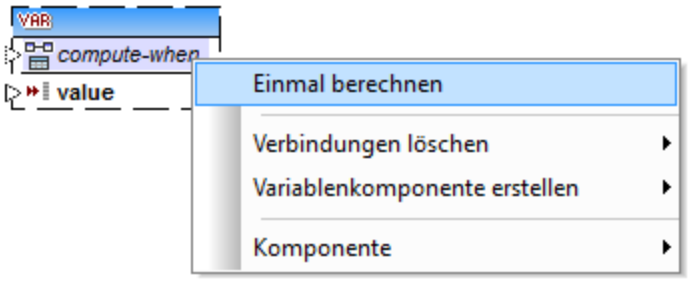
Wenn ein Output Node einer Quellkomponente mit `compute-when` verbunden ist, wird die Variable immer dann berechnet, *wenn das Quelldatenelement in der Zielsubstruktur zum ersten Mal verwendet wird*.

Die Variable verhält sich so, als wäre sie ein Child-Element des mit `compute-when` verbundenen Datenelements. Auf diese Art kann die Variable an ein bestimmtes Quelldatenelement gebunden werden, d.h. die Variable wird zur Laufzeit immer dann neu ausgewertet, wenn ein neues Datenelement aus der Sequenz in der Quellkomponente gelesen wird. Dies steht im Zusammenhang mit der allgemeinen Regel zu Verbindungen in MapForce: Erstelle für jedes Quelldatenelement ein Zieldatenelement. In diesem Fall berechnet MapForce aufgrund von `compute-when` den Variablenwert für jedes Quelldatenelement. Nähere Informationen dazu finden Sie unter Mapping-Regeln und Strategien 77.

Compute-once

Falls nötig, können Sie festlegen, dass der Variablenwert *einmal vor jedem Wert der Zielkomponenten berechnet wird*, sodass die Variable praktisch zu einer globalen Konstante für das restliche Mapping wird. Klicken Sie dazu mit der rechten Maustaste auf den Eintrag `compute-when` und wählen Sie im Kontextmenü den Befehl **Einmal berechnen**:

 © 2019-2025 Altova GmbH

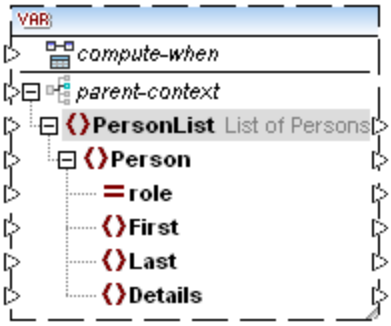


Wenn Sie den Geltungsbereich der Variablen in `compute-when=once` ändern, wird der Input-Konnektor aus dem `compute-when`-Datenelement entfernt, da eine solche Variable nur einmal ausgewertet wird. In einer benutzerdefinierten Funktion wird die Variable `compute-when=once` bei jedem Funktionsaufruf ausgewertet, bevor das tatsächliche Funktionsergebnis ausgewertet wird.

Parent-context

Das Argument `parent-context` ist ein optionales Argument in einigen MapForce Aggregatfunktionen der core-Bibliothek wie z.B. **`min`**, **`max`**, **`avg`**, **`count`**. Der parent-context bestimmt in einer Quellkomponente mit mehreren hierarchischen Sequenzen, an welcher Node-Gruppe die Funktion ausgeführt werden soll.

Ein `parent-context`-Datenelement muss z.B. dann hinzugefügt werden, wenn in Ihrem Mapping mehrere Filter verwendet werden, und Sie einen zusätzlichen Parent-Node benötigen, über den iteriert werden kann. Nähere Informationen dazu finden Sie unter Beispiel: Ändern des Parent-Kontexts 83. Um einen parent-context zu einer Variablen hinzuzufügen, klicken Sie mit der rechten Maustaste auf den Root Node, (in diesem Beispiel `PersonList`) und wählen Sie im Kontextmenü den Befehl **parent-context hinzufügen**. Daraufhin wird ein neuer Node `parent-context` zur bestehenden Hierarchie hinzugefügt.

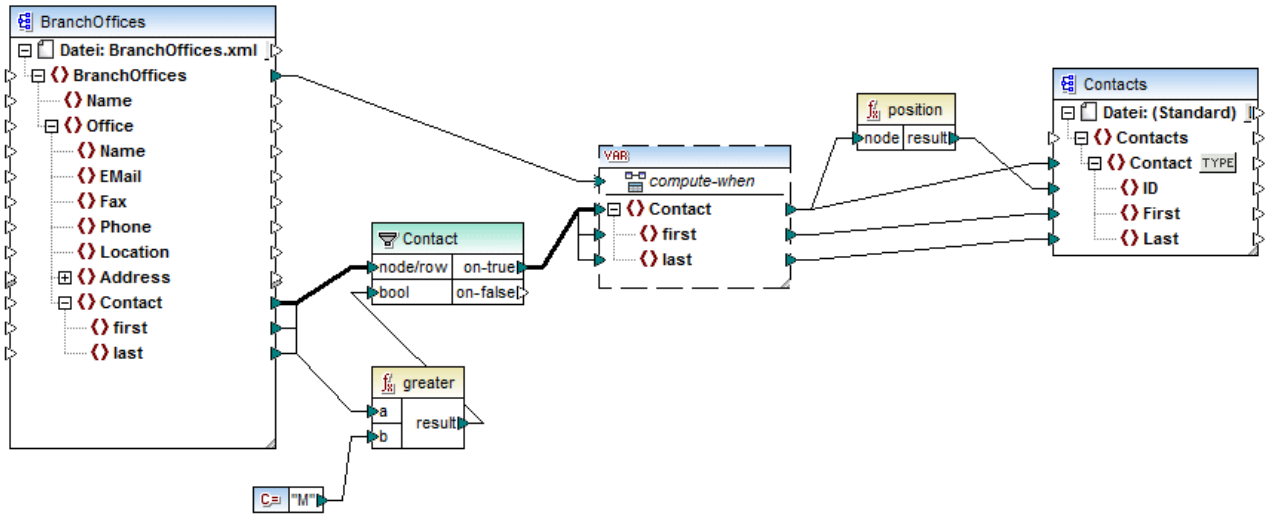


Durch den parent-context wird innerhalb der Komponente ein virtueller übergeordneter (Parent) Node zur Hierarchie hinzugefügt. Auf diese Art können Sie über einen zusätzlichen Node in derselben oder einer anderen Quellkomponente iterieren.

5.3.3 Beispiel: Filtern und Nummerieren von Nodes

Das in diesem Beispiel beschriebene Mapping finden Sie unter dem Namen **PositionInFilteredSequence.mfd** im Ordner **<Dokumente>\Altova\MapForce2025\MapForceExamples**.

Dieses Mapping liest eine XML-Datei mit Kontaktdaten mehrerer Personen, filtert die Daten und schreibt sie in eine XML-Zieldatei. Ziel des Mappings ist es, nur die Personen aus der XML-Quelldatei zu filtern, deren Nachname mit dem Buchstaben "M" oder einem nachfolgenden Buchstaben beginnt. Außerdem müssen die extrahierten Kontakte nummeriert werden. Die Nummerierung dient als eindeutiger Identifier für die einzelnen Kontakte in der XML-Zieldatei.



PositionInFilteredSequence.mfd

Um das obige Ziel zu erreichen, wurden die folgenden Komponententypen zum Mapping hinzugefügt:

- ein Filter (siehe Filter und Bedingungen 194)
- eine komplexe Variable (siehe Hinzufügen von Variablen 177)
- die Funktionen **`greater`** 280 und **`position`** 318 (siehe Hinzufügen einer Funktion zum Mapping 212)
- eine Konstante (Um eine Konstante hinzuzufügen, wählen Sie den Menübefehl **Einfügen | Konstante**).

Für die Variable wird dasselbe Schema wie für die Quellkomponente verwendet. Wenn Sie mit der rechten Maustaste auf die Variable klicken und im Kontextmenü den Befehl **Eigenschaften** wählen, sehen Sie, dass der Node **BranchOffices/Office/Contact** als Root-Node für diese Variablenstruktur ausgewählt ist.

Zuerst werden die Daten der Quellkomponente an den Filter übergeben. Der Filter übergibt nur die Datensätze an die Variable, die die Filterbedingungen erfüllen. In diesem Fall wurde der Filter so konfiguriert, dass nur die `Contact` Nodes abgerufen werden, deren Nachname gleich oder größer M ist. Zu diesem Zweck vergleicht die Funktion **`greater`** 280 jedes `last`-Datenelement mit dem Konstantenwert "M".

In der Variable wurde der `compute-when` Input mit dem Root-Datenelement der Quellkomponente (`BranchOffices`) verbunden. Dadurch wird die Variable zur Laufzeit jedes Mal, wenn ein neues Datenelement aus der Sequenz in der Quellkomponente gelesen wird, erneut ausgewertet. In diesem Mapping macht es jedoch keinen Unterschied, ob das `compute-when`-Datenelement verbunden ist oder nicht, da die Variable

 © 2019-2025 Altova GmbH

(indirekt über den Filter) mit dem Quelldatenelement `Contact` verbunden ist und so oft, wie Instanzen von `Contact`, die die Filterbedingungen erfüllen, vorhanden sind, berechnet wird.

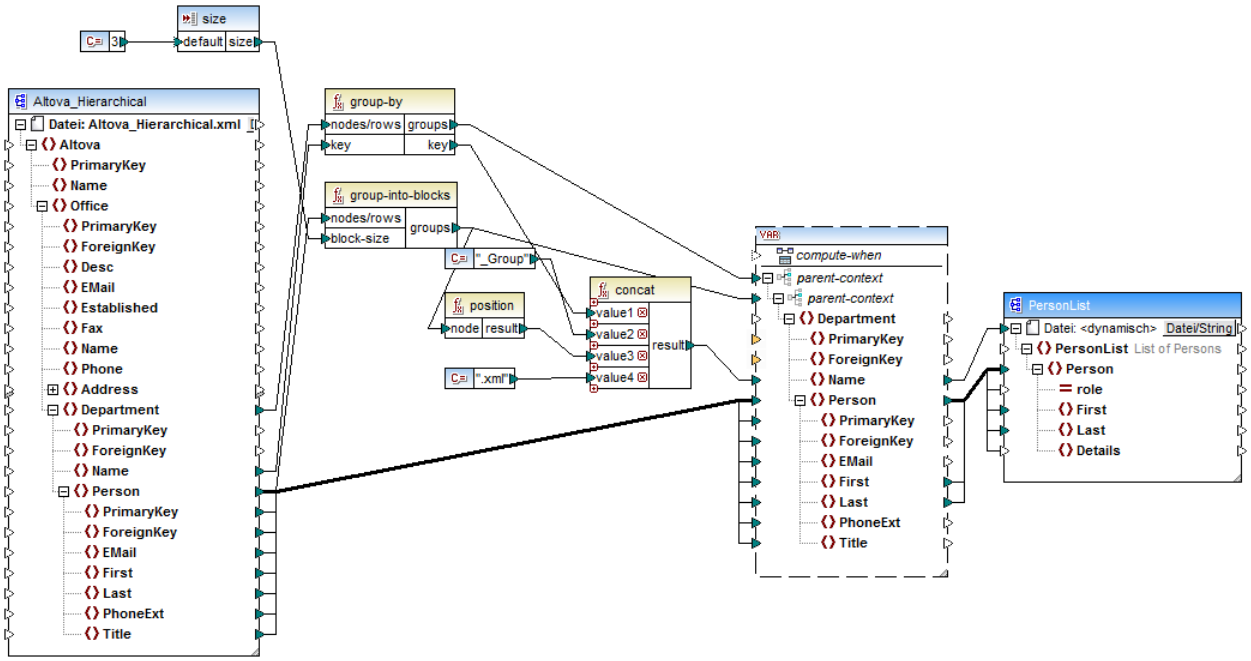
Die Funktion **`position`** 318 gibt für jede Iteration der Variablen die Nummer der aktuellen Sequenz zurück. Nur acht Kontakte erfüllen die Filterbedingungen; wenn Sie eine Vorschau des Mappings anzeigen und sich die Ausgabe ansehen, sehen Sie daher, dass die IDs 1 bis 8 in das `ID`-Element der Zielkomponente geschrieben wurden.

Falls Sie sich fragen, wozu die Variable überhaupt benötigt wurde: Der Grund dafür ist, dass alle Datensätze nummeriert werden müssen. Hätten wir das Filterergebnis direkt mit der Zielkomponente verbunden, hätte es keine Möglichkeit gegeben, die einzelnen Instanzen von `Contact` durchzunummerieren. Die Aufgabe der Variablen in diesem Mapping ist somit, die einzelnen Instanzen von `Contact` temporär im Mapping zu speichern, damit diese nummeriert werden können, bevor sie in die Zielkomponente geschrieben werden.

5.3.4 Beispiel: Unterteilen von Datensätzen in Gruppen und Untergruppen

Das in diesem Beispiel beschriebene Mapping finden Sie unter dem Namen **DividePersonsByDepartmentIntoGroups.mfd** im Ordner **<Dokumente>\Altova\MapForce2025\MapForceExamples**.

Das Mapping verarbeitet eine XML-Datei, die Mitarbeiterdatensätze einer fiktiven Firma enthält. Die Firma hat zwei Niederlassungen: "Nanonull, Inc." und "Nanonull Partners, Inc". Jede Niederlassung hat mehrere Abteilungen (departments) (z.B. "IT", "Marketing" usw.) und jede Abteilung hat einen oder mehrere Mitarbeiter. Ziel des Mappings ist es, unabhängig von der Niederlassung, aus jeder Abteilung Gruppen von maximal drei Mitarbeitern zu erstellen. Die Größe jeder Gruppe beträgt standardmäßig drei; dies sollte man jedoch bei Bedarf leicht ändern können. Jede Gruppe muss als separate XML-Datei mit dem Namen im Format "<Abteilungsname>_GruppeN" (z.B. **Marketing_Group1.xml**, **Marketing_Group2.xml**, usw.) gespeichert werden.



DividePersonsByDepartmentIntoGroups.mfd

Wie in der Abbildung oben gezeigt, wurden eine komplexe Variable sowie einige andere Komponententypen (hauptsächlich Funktionen) zum Mapping hinzugefügt. Die Variable hat dieselbe Struktur wie ein `Department`-Datenelement in der XML-Quelldatei. Wenn Sie mit der rechten Maustaste auf die Variable klicken, um ihre Eigenschaften anzuzeigen, sehen Sie, dass dafür dasselbe XML-Schema wie für die Quellkomponente verwendet wird und dass ihr Root Element `Department` ist. Wichtig ist vor allem, dass die Variable zwei ineinander verschachtelte `parent-context`-Datenelemente hat, mit denen sichergestellt wird, dass die Variable zuerst im Kontext jeder einzelnen Abteilung und anschließend im Kontext der einzelnen Gruppen in diesen Abteilungen berechnet wird (siehe auch Ändern von Kontext und Geltungsbereich von Variablen 181).

Das Mapping iteriert zuerst durch alle Abteilungen, um die Namen der einzelnen Abteilungen zu erhalten (diese Namen werden anschließend benötigt, um die Dateinamen für die einzelnen Gruppen zu erstellen). Zu diesem Zweck wird die **`group-by`** 305-Funktion mit dem Quelldatenelement `Department` verbunden und der Abteilungsname wird als Gruppierungsschlüssel bereitgestellt.

Als nächstes findet innerhalb des Kontexts der einzelnen Abteilungen eine zweite Gruppierung statt. Dabei ruft das Mapping die Funktion **`group-into-blocks`** 311 auf, um die gewünschten Mitarbeitergruppen zu erstellen. Die Größe der Gruppen wird durch eine einfache Input-Komponente mit dem Standardwert "3" angegeben. Der Standardwert stammt aus einer Konstante. Um die Gruppengröße in diesem Beispiel zu ändern, muss man nur die Konstantenwert nach Bedarf anpassen. Sie können allerdings auch die "size" Input-Komponente ändern, sodass die Größe jeder Gruppe dem Mapping einfach als Parameter bereitgestellt wird, wenn das Mapping durch generierten Code oder mit MapForce Server ausgeführt wird. Nähere Informationen dazu finden Sie unter Bereitstellen von Parametern für das Mapping 164.

Als nächstes wird der Wert der Variablen an die XML-Zielkomponente PersonList geliefert. Die Dateinamen der einzelnen erstellten Gruppen wurden durch Verkettung der folgenden Teile mit Hilfe der **`concat`** 329-Funktion berechnet:

 © 2019-2025 Altova GmbH

1. Der Name der jeweiligen Abteilung
2. Der String "_Group"
3. Die Nummer der Gruppe in der aktuellen Sequenz (z.B. "1", wenn es sich um die erste Gruppe in dieser Abteilung handelt)
4. Der String ".xml"

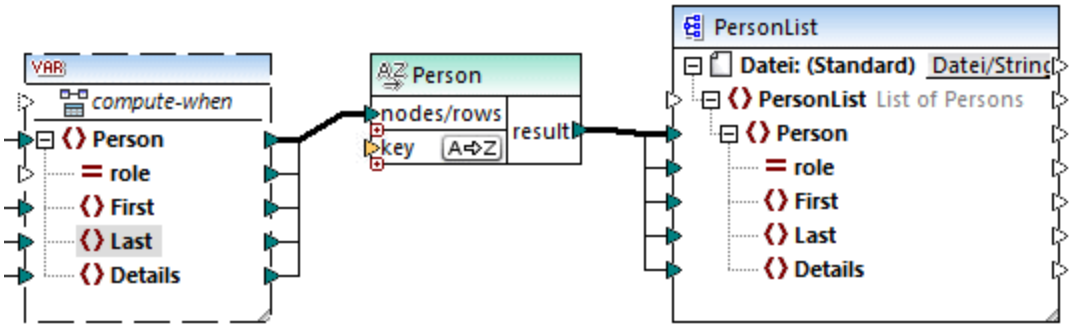
Das Ergebnis dieser Verkettung wird im Datenelement `Name` der Variablen gespeichert und anschließend als dynamischer Dateiname an die Zielkomponente weitergeleitet. Dadurch wird für jeden erhaltenen Wert ein neuer Dateiname erstellt. In diesem Beispiel werden anhand der Variablen insgesamt acht Gruppen erstellt, d.h. es werden bei Ausführung des Mappings, wie gewünscht, acht Ausgabedateien erstellt. Nähere Informationen zu diesem Methode finden Sie unter Dynamische Verarbeitung mehrerer Input- oder Output-Dateien 422.

5.4 Sortieren von Komponenten

Um Input-Daten auf Basis eines bestimmten Sortierschlüssels zu sortieren, verwenden Sie bitte die Sortierkomponente ("sort"). Die Sortierkomponente unterstützt die folgenden Zielsprachen: XSLT2, XQuery und den Built-in-Ausführungsprozessor.

Um eine Sortierkomponente zum Mapping hinzuzufügen, gehen Sie folgendermaßen vor:

- Klicken Sie mit der rechten Maustaste auf eine vorhandene Verbindung und wählen Sie im Kontextmenü den Befehl **Sortierung einfügen: Nodes/Zeilen**. Daraufhin wird die Sortierkomponente eingefügt und automatisch mit der Quell- und Zielkomponente verbunden. Im unten gezeigten Mapping wurde die Sortierkomponente z.B. zwischen einer Variablen und einer XML-Komponente eingefügt. Jetzt muss nur noch der Sortierschlüssel (das Feld, anhand dessen sortiert werden soll) manuell verbunden werden.



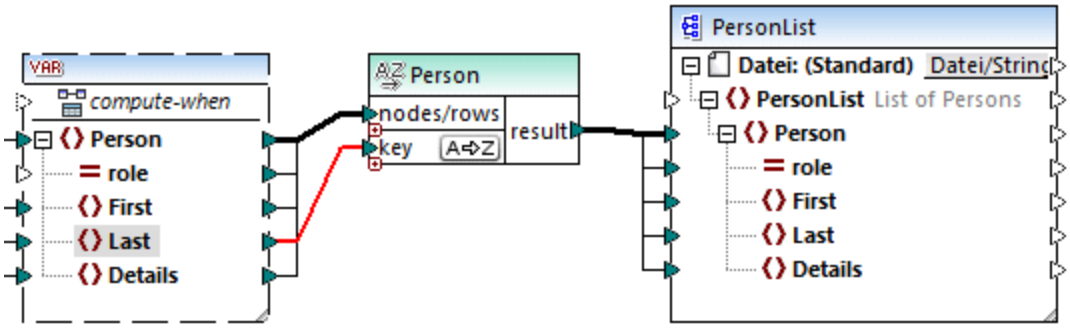
- Klicken Sie im Menü **Einfügen** auf **Sortieren** (oder alternativ dazu auf die Symbolleisten-Schaltfläche **Sortieren**). Daraufhin wird die Sortierkomponente in ihrer nicht verbundenen Form eingefügt.

Sobald eine Verbindung zur Quellkomponente hergestellt wurde, ändert sich der Name der Titelleiste in denjenigen des mit dem Datenelement `nodes/rows` verbundenen Datenelements.

So definieren Sie, nach welchem Datenelement sortiert werden soll:

- Verbinden Sie das Datenelement, nach dem sortiert werden soll, mit dem Parameter `key` der Sortierkomponente. Im unten gezeigten Mapping wurden die `Person`-Nodes/Zeilen z.B. nach dem Feld `Last` sortiert.

 © 2019-2025 Altova GmbH

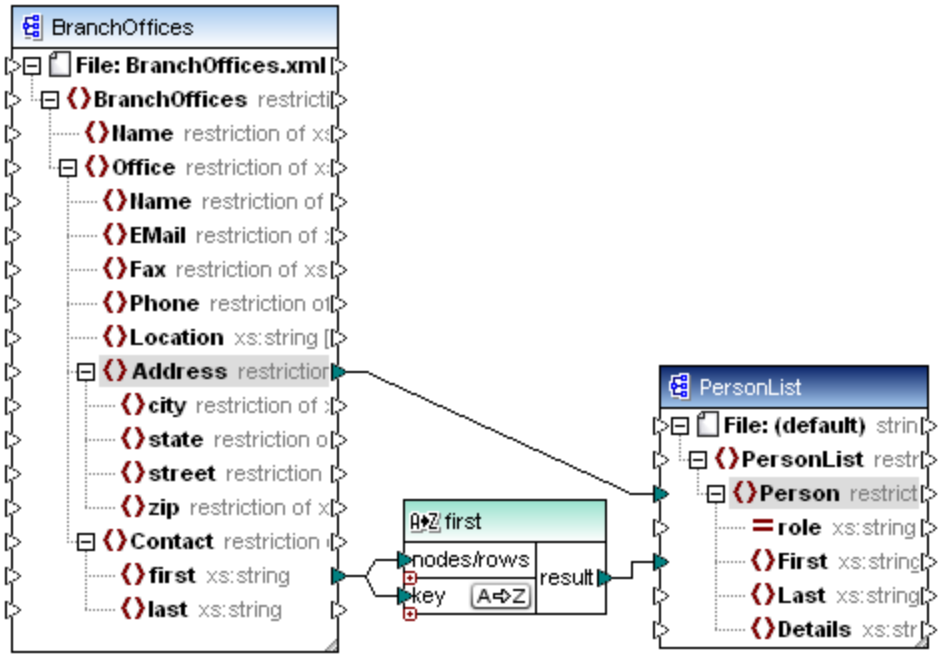


So ändern Sie die Sortierreihenfolge:

- Klicken Sie in der Sortierkomponente auf das Symbol A⇨Z. Es ändert sich daraufhin in Z➔A, um anzuzeigen, das die Sortierreihenfolge umgekehrt wurde.

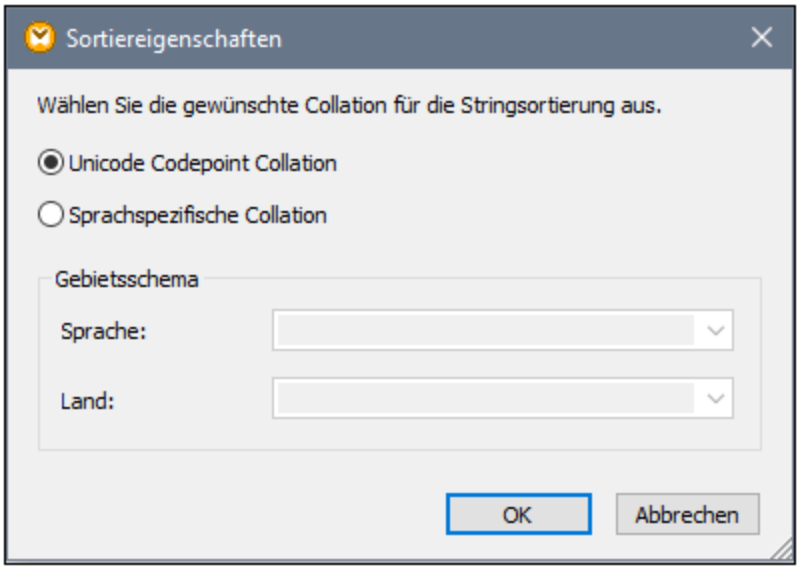
So sortieren Sie Input-Daten, die aus Einträgen vom Typ "simpleType" bestehen:

- Verbinden Sie das Datenelement mit den beiden Parametern `nodes/rows` und `key` der Sortierkomponente. Im unten gezeigten Mapping wird das Element vom simpleType `first` sortiert.



So sortieren Sie Strings mit Hilfe von sprachspezifischen Regeln:

- Doppelklicken Sie auf die Titelleiste der Sortierkomponente, um das Dialogfeld "Sortiereigenschaften" zu öffnen.

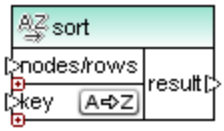


Unicode Codepoint Collation: Mit dieser Standardoption werden Strings auf Basis ihrer Codepoint-Werte verglichen/sortiert. Codepoint-Werte sind Ganzzahlen, die abstrakten Zeichen in dem vom Unicode Consortium abgesegneten universalen Zeichensatz zugewiesen sind. Mit Hilfe dieser Option kann eine Sortierung in den verschiedensten Sprachen und Scripts vorgenommen werden.

Sprachspezifische Collation: Mit dieser Option können Sie die Variante für eine bestimmte Sprache und ein bestimmtes Land festlegen, nach dem sortiert werden soll. Diese Option wird bei Verwendung des BUILT-IN-Ausführungsprozessors unterstützt. Im Fall von XSLT hängt die Unterstützung vom Prozessor ab, der für die Ausführung des Codes verwendet wird.

5.4.1 Sortieren nach mehreren Schlüsseln

Nachdem Sie eine Sortierkomponente zum Mapping hinzugefügt haben, wird standardmäßig ein einziger Sortierschlüssel namens `key` erstellt.



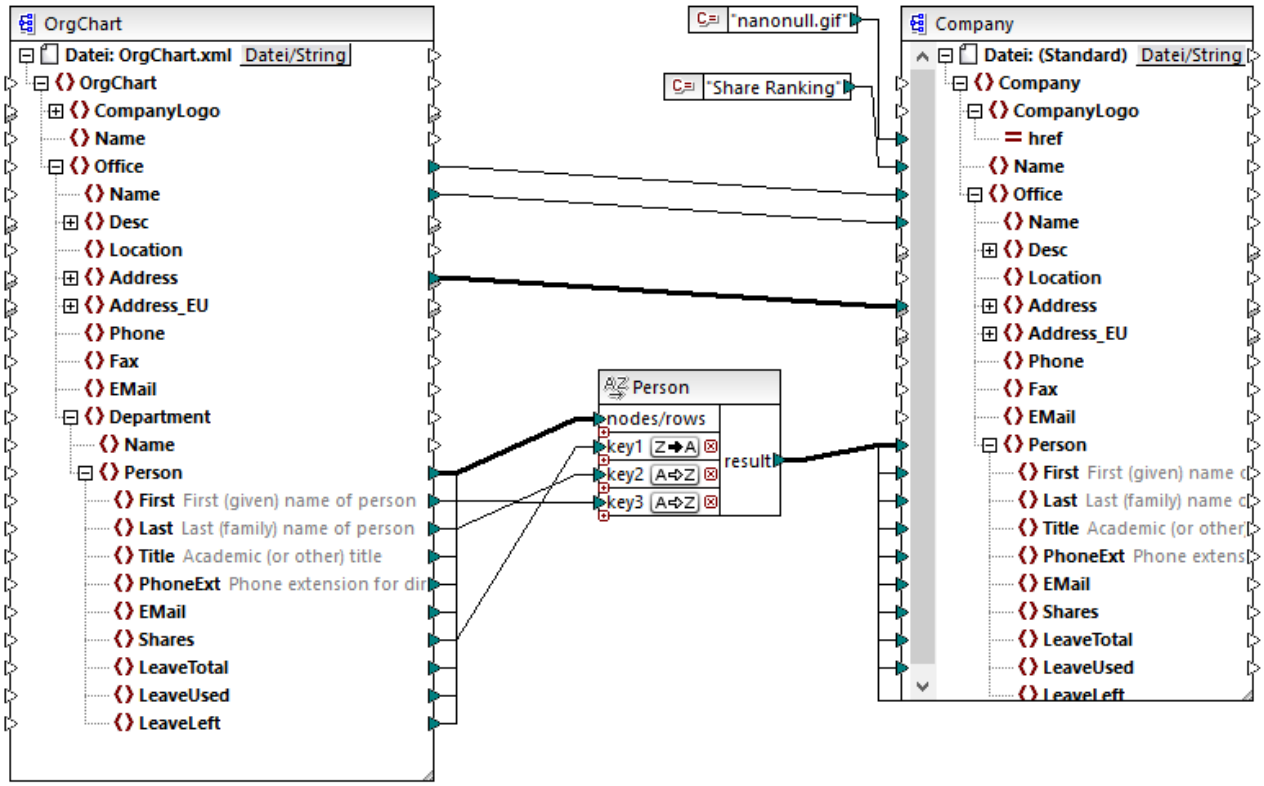
Standardsortierkomponente

Wenn Sie nach mehreren Schlüsseln sortieren möchten, passen Sie die Sortierkomponente folgendermaßen an:

- Klicken Sie auf das Symbol **Schlüssel hinzufügen** (⊡), um einen neuen Schlüssel (z.B. `key2` im Mapping unten) hinzuzufügen.
- Klicken Sie auf das Symbol **Schlüssel löschen** (⊠), um einen Schlüssel zu löschen.
- Ziehen Sie eine Verbindung auf das Symbol ⊡, um einen Schlüssel hinzuzufügen und gleichzeitig zu verbinden.

 © 2019-2025 Altova GmbH

Unter dem folgenden Pfad finden Sie ein Mapping zum Veranschaulichung der Sortierung nach mehreren Schlüsseln: **<Dokumente>\Altova\MapForce2025\MapForceExamples\SortByMultipleKeys.mfd**.



SortByMultipleKeys.mfd

Im oben gezeigten Mapping wurden die `Person`-Datensätze nach drei Sortierschlüsseln sortiert:

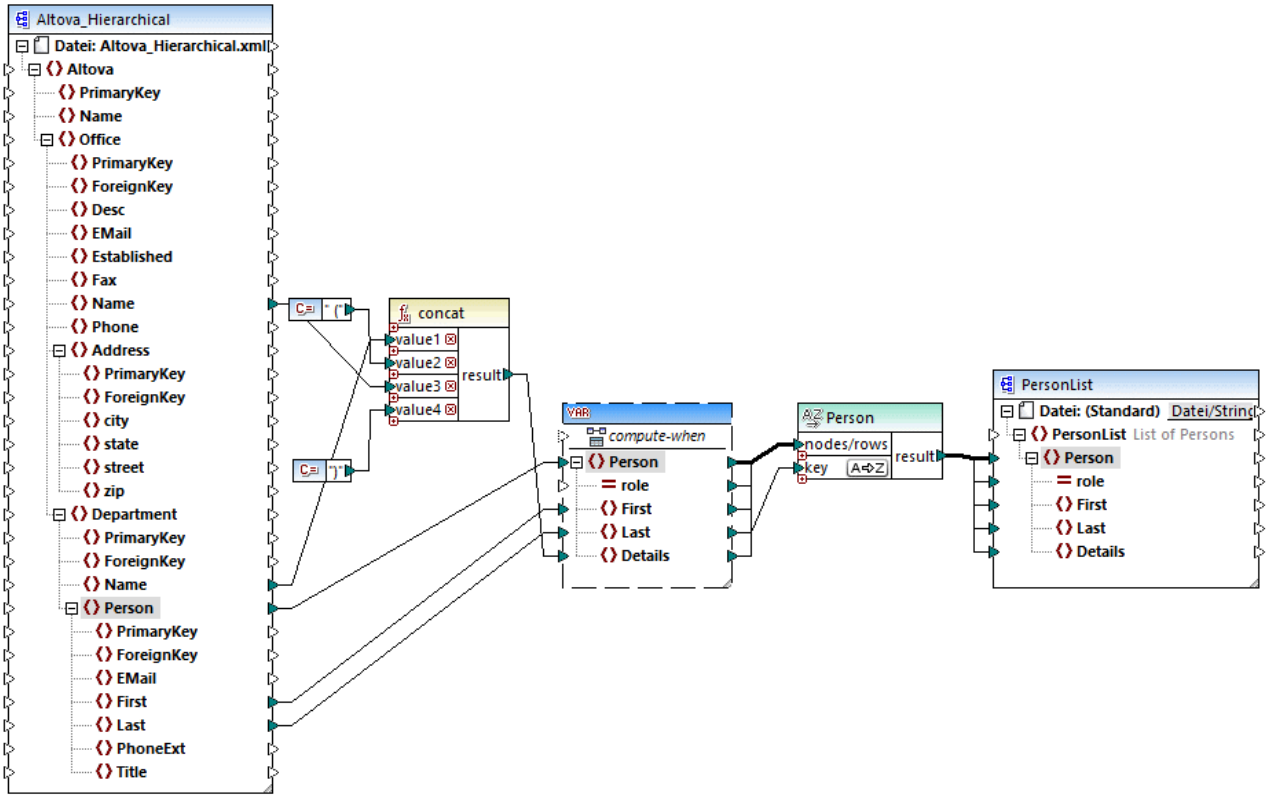
1. `Shares` (Anzahl der Aktien im Besitz einer Person)
2. `Last` (Nachname)
3. `First` (Vorname)

Beachten Sie, dass die Position des Sortierschlüssels in der Sortierkomponente die Sortierpriorität festlegt. So werden die Datensätze etwa im Beispiel oben zuerst nach der Anzahl der Aktien (shares) sortiert. Dies ist der Sortierschlüssel mit der höchsten Priorität. Wenn die Anzahl der Aktien dieselbe ist, werden die Personen nach ihrem Nachnamen sortiert. Wenn schließlich mehrere Personen dieselbe Anzahl von Aktien und denselben Nachnamen haben, wird der Vorname der Person berücksichtigt.

Die Sortierreihenfolge kann für jeden Schlüssel unterschiedlich sein. Im oben gezeigten Mapping hat der Schlüssel `Shares` eine absteigende Sortierreihenfolge (Z-A), während die anderen beiden Schlüssel eine aufsteigende Sortierreihenfolge haben (A-Z).

5.4.2 Sortieren mit Variablen

In einigen Fälle müssen eventuell Zwischenvariablen zum Mapping hinzugefügt werden, um das gewünschte Ergebnis zu erzielen. In diesem Beispiel wird gezeigt, wie Sie Datensätze aus einer XML-Dati extrahieren und mit Hilfe von Zwischenvariablen sortieren. Das Mapping-Beispiel dazu finden Sie unter dem folgenden Pfad: **<Dokumente>\Altova\MapForce2025\MapForceExamples\Altova_Hierarchical_Sort.mfd**.



Altova_Hierarchical_Sort.mfd

In diesem Mapping werden Daten aus einer XML-Quelldatei namens **Altova_Hierarchical.xml** gelesen und in eine XML-Zieldatei geschrieben. Wie Sie oben sehen, enthält die XML-Quellkomponente Informationen über ein fiktives Unternehmen. Die Firma besteht aus Niederlassungen (Office). Die Niederlassungen sind in Abteilungen (Department) unterteilt, die wiederum aus Personen (Person) bestehen.

Die XML-Zielkomponente `PersonList` enthält eine Liste von `Person`-Datensätzen. Das Datenelement `Details` dient zur Aufnahme von Informationen über die Niederlassung und Abteilung, zu der die Person gehört.

Ziel ist es, alle Personen aus der XML-Quelldatei zu extrahieren und alphabetisch nach dem Nachnamen (last name) zu sortieren. Außerdem müssen der Name der Niederlassung und der Abteilung, zu der jede Person gehört, in das Datenelement `Details` geschrieben werden.

Um dies zu erreichen, werden in diesem Beispiel die folgenden Komponententypen verwendet:

 © 2019-2025 Altova GmbH

1. die Funktion **`concat`**. Diese Funktion gibt in diesem Mapping einen String im Format `Office(Department)` zurück. Sie erhält als Input den Namen der Niederlassung und den Abteilungsnamen und zwei Konstanten, die die öffnende und schließende Klammer bereitstellen. Siehe auch Hinzufügen einer Funktion zum Mapping 212.
2. eine Zwischenvariable. Die Rolle der Variablen ist es, alle für eine Person relevanten Daten in denselben Mapping-Kontext zu bringen. Aufgrund der Variablen werden im Kontext jeder Person jeweils die Abteilung und die Niederlassung einer Person nachgesehen. Anders ausgedrückt, merkt sich die Variable den Namen der Niederlassung und Abteilung, zu der eine Person gehört. Ohne die Variable wäre der Kontext falsch, was im Mapping zu unerwünschten Ergebnissen führen würde (eine genauere Anleitung zum Ausführen eines Mappings finden Sie unter Mapping-Regeln und -Strategien 77). Beachten Sie, dass die Variable die Struktur der XML-Zieldatei repliziert (sie verwendet dasselbe XML-Schema). Auf diese Art kann das Sortierergebnis mittels einer "Alles kopieren"-Verbindung mit der Zielkomponente verbunden werden. Siehe auch Verwendung von Variablen 175 und "Alles kopieren"-Verbindungen 54.
3. eine Sortierkomponente, die die eigentliche Sortierung durchführt. Beachten Sie, dass der Input "key" der `sort`-Komponente mit dem Datenelement `Last` der Variablen verbunden ist, wodurch alle Personendatensätze nach dem Nachnamen sortiert werden.

5.5 Filter und Bedingungen

Wenn Sie Daten filtern oder basierend auf einer Bedingung einen Wert ermitteln müssen, können Sie dazu die folgenden Komponententypen verwenden:

- Filter: Nodes/Zeilen ( )
- If-Else-Bedingung ( )

Sie können diese Komponenten entweder über das Menü **Einfügen** oder über die Symbolleiste **Komponente einfügen** zum Mapping hinzufügen. Beachten Sie, dass jede der obigen Komponenten ein bestimmtes Verhalten aufweist und bestimmte Voraussetzungen hat. Die Unterschiede sind in den folgenden Abschnitten erklärt.

Filtern von Nodes oder Zeilen

Um Daten, darunter XML-Nodes, zu filtern, verwenden Sie eine **Filter: Nodes/Zeilen**-Komponente. Mit Hilfe der **Filter: Nodes/Zeilen**-Komponente können Sie eine Untergruppe von Nodes anhand einer true- oder false-Bedingung aus einer größeren Datenmenge abrufen. Ihre Struktur im Mapping-Bereich sieht folgendermaßen aus:

In der oben gezeigten Struktur definiert die mit **bool** verbundene Bedingung, ob der verbundene Node/die verbundene Zeile in den **on-true** oder in den **on-false** Output geleitet wird. Wenn die Bedingung true lautet, wird **der Node/die Zeile** an den **on-true**-Output übergeben. Umgekehrt, wenn die Bedingung false ist, wird der **Node/die Zeile** an den **on-false**-Output übergeben.

Wenn Sie in Ihrem Mapping nur Datenelemente, die die Filterbedingung erfüllen, verwenden möchten, können Sie den **on-false**-Output unverbunden lassen. Wenn die Datenelemente, die die Filterbedingung *nicht erfüllen*, verarbeitet werden müssen, verbinden Sie den **on-false**-Output mit einer Zielkomponente für solche Datenelemente.

Eine schrittweise Anleitung anhand eines Mapping-Beispiels finden Sie unter Beispiel: Filtern von Nodes 196.

Rückgabe eines Werts auf Basis einer Bedingung

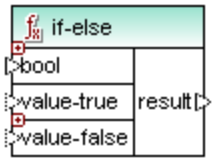
Wenn Sie einen einzelnen Wert (und nicht einen Node oder eine Zeile) auf Basis einer Bedingung abrufen möchten, verwenden Sie eine **If-Else-Bedingung**. Beachten Sie, dass If-Else-Bedingungen sich nicht zum Filtern von Nodes oder Zeilen eignen. Im Gegensatz zu **Filter: Nodes/Zeilen**-Komponenten gibt eine **If-Else-Bedingung** einen Wert vom Typ simpleType (z.B. einen String oder eine Ganzzahl) zurück. **If-Else-Bedingungen** eignen sich daher nur für Szenarien, in denen ein einzelner Wert anhand einer Bedingung verarbeitet werden soll. Angenommen, Sie haben eine Liste von Durchschnittstemperaturen pro Monat im folgenden Format:

 © 2019-2025 Altova GmbH

```
<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>
```

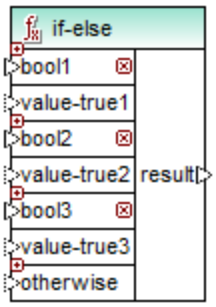
Mit Hilfe einer **If-Else-Bedingung** können Sie für jedes Datenelement in der Liste, dessen Temperaturwert höher als 20 Grad Celsius beträgt, den Wert "hoch" und für jedes Datenelement, dessen Temperaturwert niedriger als 5 Grad Celsius beträgt, "niedrig" zurückgeben.

Im Mapping sieht die Struktur der **If-Else-Bedingung** folgendermaßen aus:



Wenn die mit **bool** verbundene Bedingung true ist, so wird der mit **value-true** verbundene Wert als **result** ausgegeben. Wenn die Bedingung false ist, so wird der mit **value-false** verbundene Wert als **result** ausgegeben. Der Datentyp von **result** ist im Vorhinein nicht bekannt; er hängt vom Datentyp des mit **value-true** oder **value-false** verbundenen Werts ab. Wichtig ist, dass es sich immer um einen simpleType (String, Ganzzahl, usw.) handeln muss. Die Verbindung von Input-Werten von complexTypes (wie Nodes oder Zeilen) wird von **If-Else-Bedingungen** nicht unterstützt.

If-Else-Bedingungen sind erweiterbar, d.h. Sie können durch Klick auf die Schaltfläche **Hinzufügen** (⊞) mehrere Bedingungen zur Komponente hinzufügen. Um eine zuvor hinzugefügte Bedingung zu löschen, klicken Sie auf die Schaltfläche **Löschen** (⊠). Auf diese Art können Sie mehrere Bedingungen überprüfen und für jede Bedingung einen anderen Wert zurückgeben, falls die Bedingung true ist.



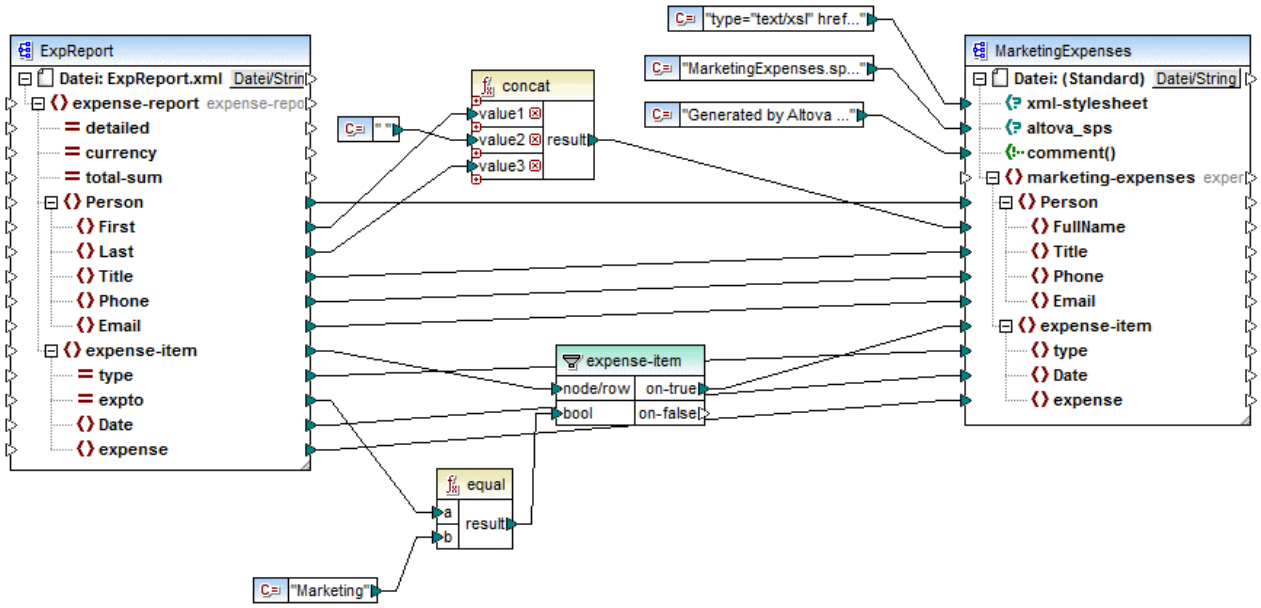
Erweiterte **If-Else-Bedingungen** werden von oben nach unten ausgewertet (die erste Bedingung wird als erste überprüft, anschließend die zweite, usw.). Wenn ein Wert zurückgegeben werden soll, wenn keine der Bedingungen true ist, verbinden Sie sie mit **otherwise**.

Eine schrittweise Anleitung anhand eines Mapping-Beispiels finden Sie unter Beispiel: Rückgabe eines Werts auf Basis einer Bedingung 198.

5.5.1 Beispiel: Filtern von Nodes

In diesem Beispiel wird gezeigt, wie Sie Nodes auf Basis einer true/false-Bedingung filtern können. Zu diesem Zweck wird eine **Filter: Nodes/Zeilen** ( )-Bedingung eingesetzt.

Das in diesem Beispiel beschriebene Mapping befindet sich unter dem folgenden Pfad: **<Dokumente>\Altova\MapForce2025\MapForceExamples\MarketingExpenses.mfd**.



Wie oben gezeigt, liest das Mapping Daten aus einer XML-Quelldatei, die einen Spesenbericht enthält aus ("ExpReport") und schreibt Daten in eine XML-Zieldatei ("MarketingExpenses"). Zwischen der Ziel- und der Quellkomponente befinden sich einige weitere Komponenten. Die wichtigste davon ist der **expense-item**-Filter ( ), der das Thema dieses Kapitels ist.

Ziel des Mappings ist es, nur die Ausgabenposten herauszufiltern, die der Marketing-Abteilung zugeschrieben werden. Zu diesem Zweck wurde eine Filter-Komponente zum Mapping hinzugefügt. (Um einen Filter hinzuzufügen, klicken Sie auf das Menü **Einfügen** und wählen Sie den Befehl **Filter: Nodes/Zeilen**.)

Um festzustellen, ob ein Ausgabenposten zu Marketing gehört, wird der Wert des Attributs "expto" in der Quellkomponente überprüft. Dieses Attribut hat den Wert "Marketing", wenn es sich um eine Marketing-Ausgabe handelt. Im Codefragment unten gehören das erste und dritte Datenelement zu Marketing, das zweite zu Development und das vierte zu 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>
...
```

XML-Input vor Ausführung des Mappings

Im Mapping-Bereich wird der **node/row**-Input des Filters mit dem **expense-item**-Node in der Quellkomponente verbunden. Damit wird gewährleistet, dass die Filterkomponente die Liste der zu verarbeitenden Nodes erhält.

Als Bedingung, anhand welcher die Filterung erfolgen soll, haben wir die **equal**-Funktion aus der MapForce core-Bibliothek hinzugefügt (nähere Informationen dazu finden Sie unter Hinzufügen einer Funktion zum Mapping 212). Die **equal**-Funktion vergleicht den Wert des Attributs `expto` mit einer Konstante, die den Wert "Marketing" hat. (Um eine Konstante hinzuzufügen, klicken Sie auf das Menü **Einfügen** und anschließend auf **Konstante**.)

Da wir nur diejenigen Datenelemente, die die Bedingung erfüllen, filtern müssen, haben wir nur den **on-true**-Input des Filters mit der Zielkomponente verbunden.

Wenn Sie durch Klicken auf das Fenster **Ausgabe** eine Vorschau auf das Mapping-Ergebnis anzeigen, wertet MapForce für jeden expense-item-Node die mit dem **bool**-Input des Filters verbundene Bedingung aus. Wenn die Bedingung true ist, wird der expense-item-Node an die Zielkomponente übergeben; andernfalls wird er ignoriert. In der Ausgabe werden folglich nur die Ausgabenposten, die den Kriterien entsprechen, angezeigt:

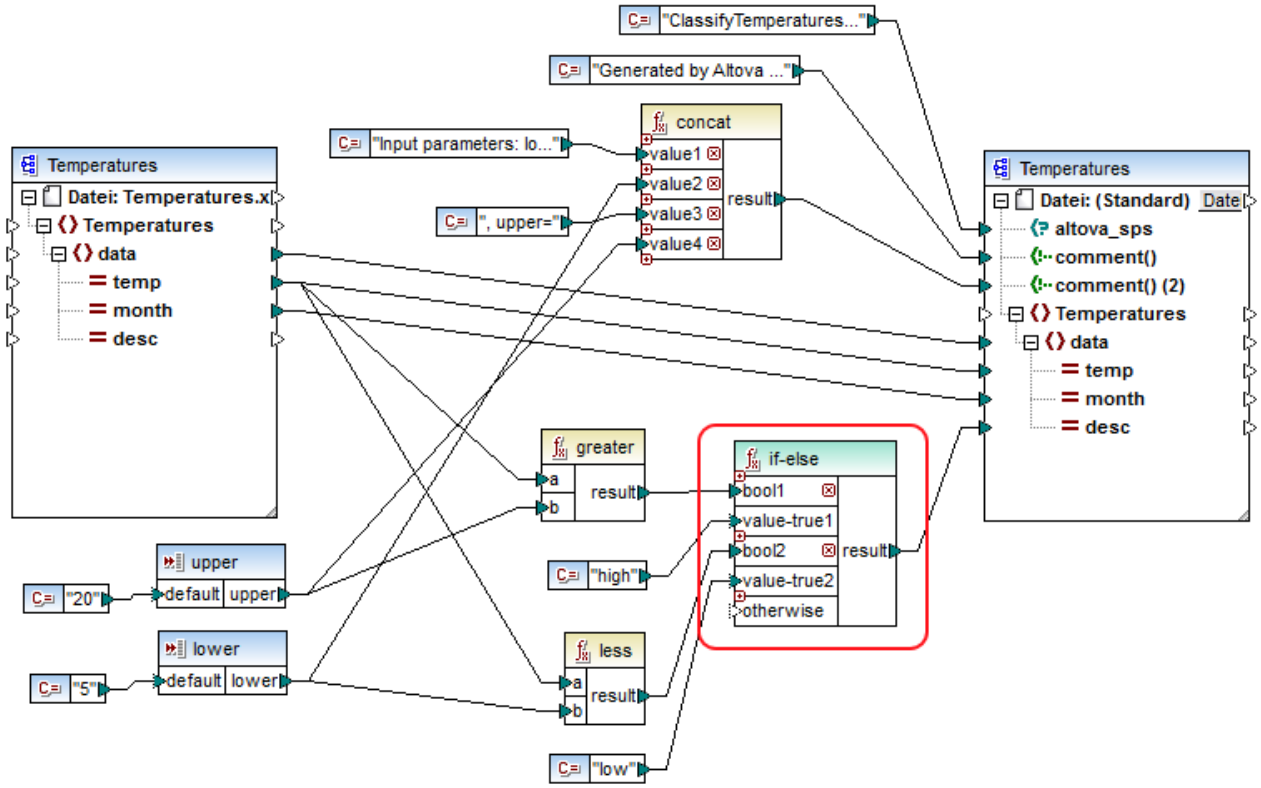
```
...
   <expense-item>
      <type>Meal</type>
      <Date>2003-01-01</Date>
      <expense>122.11</expense>
   </expense-item>
   <expense-item>
      <type>Lodging</type>
      <Date>2003-01-02</Date>
      <expense>299.45</expense>
   </expense-item>
...
```

XML-Ausgabe nach Ausführung des Mappings

5.5.2 Beispiel: Rückgabe eines Werts auf Basis einer Bedingung

In diesem Beispiel wird gezeigt, wie Sie einen einfachen Wert auf Basis einer true/false-Bedingung aus einer Komponente abrufen. Zu diesem Zweck wird eine **If-Else-Bedingung** ( ) verwendet. **If-Else-Bedingungen** dürfen nicht mit Filterkomponenten verwechselt werden. **If-Else-Bedingungen** eignen sich nur, um einfache Werte (Strings, Ganzzahlen, usw.) auf Basis einer Bedingung zu verarbeiten. Wenn Sie komplexe Werte wie Nodes filtern müssen, verwenden Sie stattdessen einen Filter (siehe Beispiel: Filtern von Nodes 196).

Das in diesem Beispiel beschriebene Mapping befindet sich unter dem folgenden Pfad: **%EXFOLDER% >ClassifyTemperatures.mfd**.



In diesem Mapping werden aus einer XML-Quelldatei, die Temperaturdaten ("Temperatures") enthält, Daten ausgelesen und in eine XML-Zieldatei geschrieben, die demselben Schema entspricht. Zwischen der Ziel- und der Quellkomponente befinden sich einige weitere Komponenten. Eine davon ist die hier beschriebene **if-else**-Bedingung (rot markiert).

Ziel des Mappings ist es, zu jedem Temperaturdatensatz in der Zielkomponente eine kurze Beschreibung hinzuzufügen. Wenn die Temperatur mehr als 20 Grad Celsius ist, so soll die Beschreibung "high" lauten. Wenn die Temperatur weniger als 5 Grad Celsius ist, soll die Beschreibung "low" lauten. In allen anderen Fällen soll keine Beschreibung hinzugefügt werden.

Zu diesem Zweck benötigen wir eine bedingte Verarbeitung. Um dies zu erreichen, wurde eine If-Else-Bedingung zum Mapping hinzugefügt. (Um eine If-Else-Bedingung hinzuzufügen, klicken Sie auf das Menü

 © 2019-2025 Altova GmbH

Einfügen und wählen Sie den Befehl **If-Else-Bedingung**.). Die If-Else-Bedingung in diesem Mapping wurde (mittels der Schaltfläche ⊡) erweitert, um zwei Bedingungen Platz zu bieten: **bool1** und **bool2**.

Die Bedingungen selbst werden von den aus der MapForce Core-Bibliothek hinzugefügten Funktionen **greater** und **less** bereitgestellt (nähere Informationen dazu finden Sie unter Hinzufügen einer Funktion zum Mapping 212). Diese Funktionen werten die von den beiden Input-Komponenten "upper" und "lower" bereitgestellten Werte aus. (Um eine Input-Komponente hinzuzufügen, klicken Sie auf das Menü **Einfügen** und anschließend auf **Input-Komponente einfügen**. Nähere Informationen zu Input-Komponenten finden Sie unter Bereitstellen von Parametern für das Mapping 164.)

Die Funktionen **greater** und **less** geben entweder true oder false zurück. Vom Ergebnis der Funktion ist es abhängig, was in die Zielinstanz geschrieben wird. Wenn also der Wert des Attributs "temp" in der Quellkomponente größer als 20 ist, so wird der Konstantenwert "high" an die **if-else**-Bedingung übergeben. Wenn der Wert des Attributs "temp" in der Quellkomponente kleiner als 5 ist, so wird der Konstantenwert "low" an die **if-else**-Bedingung übergeben. Der **otherwise**-Input wird nicht verbunden. Wenn daher keine der obigen Bedingungen zutrifft, wird nichts an den **result**-Output-Konnektor übergeben.

Der **result** Output-Konnektor übergibt schließlich diesen Wert (einmal für jeden Temperaturdatensatz) an das Attribut "desc" in der Zielkomponente.

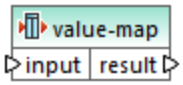
Wenn Sie fertig sind, klicken Sie auf das Fenster **Ausgabe**, um eine Vorschau auf das Mapping-Ergebnis zu sehen. Beachten Sie, dass die erzeugte XML-Ausgabe nun in allen Nodes, in denen "temperature" größer als 20 oder kleiner als 5 ist, das Attribut "desc" enthält.

```
...
   <data temp="-3.6" month="2006-01" desc="low"/>
   <data temp="-0.7" month="2006-02" desc="low"/>
   <data temp="7.5" month="2006-03"/>
   <data temp="12.4" month="2006-04"/>
   <data temp="16.2" month="2006-05"/>
   <data temp="19" month="2006-06"/>
   <data temp="22.7" month="2006-07" desc="high"/>
   <data temp="23.2" month="2006-08" desc="high"/>
...
```

XML-Ausgabe nach Ausführung des Mappings

5.6 Wertezuordnungen

Mit Hilfe der Wertezuordnungskomponente (*Abbildung unten*) können Sie einen Wert mit Hilfe einer vordefinierten Lookup-Tabelle durch einen anderen Wert ersetzen. Eine solche Komponente verarbeitet immer nur einen Wert auf einmal, daher hat sie im Mapping einen **Input** und ein **Ergebnis** (result).



Eine Wertezuordnung eignet sich, um einzelne Datenelemente in zwei Datensätzen zu mappen, um Datenelemente zu ersetzen. So könnten Sie z.B. in Form von Zahlen ausgedrückte Wochentage (1, 2, 3, 4, 5, 6 und 7) auf die Namen der einzelnen Wochentage ("Montag", "Dienstag", usw.) mappen. Oder Sie können die Namen der Monate ("Januar", "Februar", "März", usw.) auf die Zahlendarstellung der einzelnen Monate (1, 2, 3, usw.) mappen. Zur Mapping-Laufzeit werden die entsprechenden Werte entsprechend Ihrer benutzerdefinierten Lookup-Tabelle ersetzt. Die Werte in den beiden Datensätzen können einen unterschiedlichen Typ haben, doch müssen für jeden Datensatz Werte desselben Datentyps gespeichert sein.

Wertezuordnungskomponenten eignen sich für einfache Lookup-Verfahren, bei denen jeder Wert im ersten Datensatz einem einzigen Wert im zweiten Datensatz entspricht. Wenn ein Wert nicht in der Lookup-Tabelle gefunden wird, können Sie ihn entweder durch einen benutzerdefinierten Wert oder durch einen leeren Wert ersetzen oder ihn unverändert an die zweite Komponente übergeben. Wenn Sie Werte anhand komplexerer Kriterien nachschlagen oder filtern müssen, verwenden Sie stattdessen eine der Filterkomponenten 194.

Wenn Sie Code generieren oder eine MapForce Server-Ausführungsdatei anhand des Mappings generieren, wird die Lookup-Tabelle in den generierten Code bzw. die generierte Datei eingebettet. Daher empfiehlt es sich nur dann direkt im Mapping eine Lookup-Tabelle zu definieren, wenn sich Ihre Daten nicht ständig ändern und die Tabelle nicht sehr groß ist (nicht größer als einige hundert Einträge). Wenn sich die Lookup-Daten regelmäßig ändern, ist es oft schwierig sowohl das Mapping als auch den generierten Code regelmäßig auf aktuellem Stand zu halten. Es ist einfacher, die Lookup-Tabelle in Form einer Text-, XML-, Datenbank- oder eventuell Excel-Datei zu warten.

Bei sehr großen Lookup-Tabellen wird die Mapping-Ausführung durch die Lookup-Tabelle verlangsamt. In diesem Fall wird empfohlen, stattdessen eine Datenbankkomponente mit SQL-Where zu verwenden. Datenbankkomponenten stehen in der MapForce Professional und Enterprise Edition zur Verfügung. Besonders gut dafür eigenen sich aufgrund Ihrer Portabilität SQLite-Datenbanken. Serverseitig können Sie die Performance von Lookup-Tabellen durch Ausführung eines Mappings mit der MapForce Server oder MapForce Server Advanced Edition verbessern.

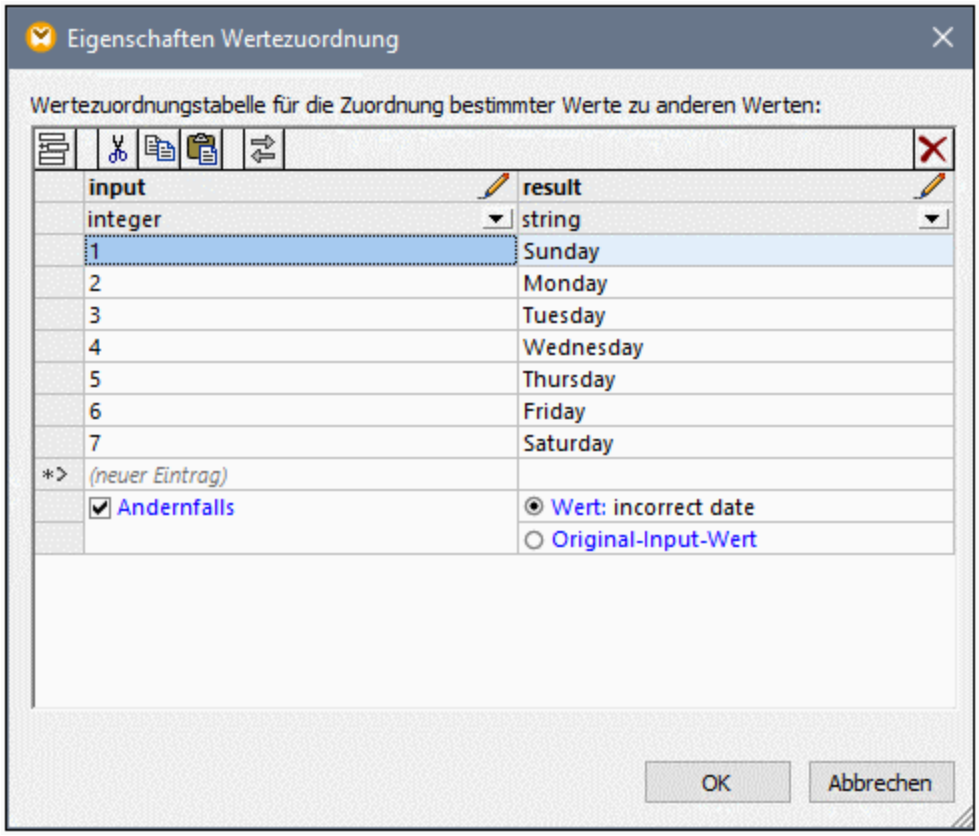
Erstellen von Wertezuordnungen

Um eine Wertezuordnungskomponente zum Mapping hinzuzufügen, wählen Sie eine der folgenden Methoden:

- Klicken Sie auf die Symbolleisten-Schaltfläche **Wertezuordnung einfügen** .
- Klicken Sie im Menü **Einfügen** auf **Wertezuordnung**.
- Klicken Sie mit der rechten Maustaste auf eine Verbindung und wählen Sie im Kontextmenü den Befehl **Wertezuordnung einfügen**.

 © 2019-2025 Altova GmbH

Daraufhin wird eine neue Wertezuordnungskomponente zum Mapping hinzugefügt. Sie können nun Datenelementpaare zur Lookup-Tabelle hinzufügen. Doppelklicken Sie dazu auf die Titelleiste der Komponente oder rechtsklicken Sie auf die Komponente und wählen Sie im Kontextmenü den Befehl **Eigenschaften**



MapForce ersetzt die einzelnen Werte, die zum **Input** der Wertezuordnung gelangen zur Mapping-Laufzeit. Wenn die *linke Spalte* der Lookup-Tabelle einen übereinstimmenden Wert enthält, wird der Original-Input-Wert durch den Wert aus der *rechten Spalte* ersetzt. Andernfalls haben Sie die Möglichkeit, folgende Alternativen zu konfigurieren:

- einen Ersetzungswert. Im obigen Beispiel ist der Ersetzungswert der Text "incorrect date" (falsches Datum). Sie können auch festlegen, dass der Ersetzungswert leer ist, indem Sie keinen Text eingeben.
- den Original-Input-Wert: In diesem Fall wird der Original-Input-Wert unverändert an das Mapping übergeben, wenn die Lookup-Tabelle keine Entsprechung enthält.

Wenn Sie keine "Andernfalls"-Bedingung konfigurieren, wird jedesmal, wenn keine Entsprechung gefunden wird, ein **leerer Node** zurückgegeben. In diesem Fall wird nichts an die Zielkomponente übergeben und die Ausgabe enthält fehlende Felder. Um dies zu vermeiden, sollten Sie entweder die Andernfalls-Bedingung konfigurieren oder die Funktion `substitute-missing` 325 verwenden.

Es gibt einen Unterschied zwischen der Definition eines leeren Ersetzungswerts und einer überhaupt fehlenden Definition der Andernfalls-Bedingung. Im ersten Fall wird das Feld in der Ausgabe generiert, hat aber einen

leeren Wert. Im zweiten Fall wird das Feld (oder XML-Element), das den Wert enthält, gar nicht erstellt. Nähere Informationen dazu finden Sie unter: Beispiel: Ersetzen von Stellenbezeichnungen 208.

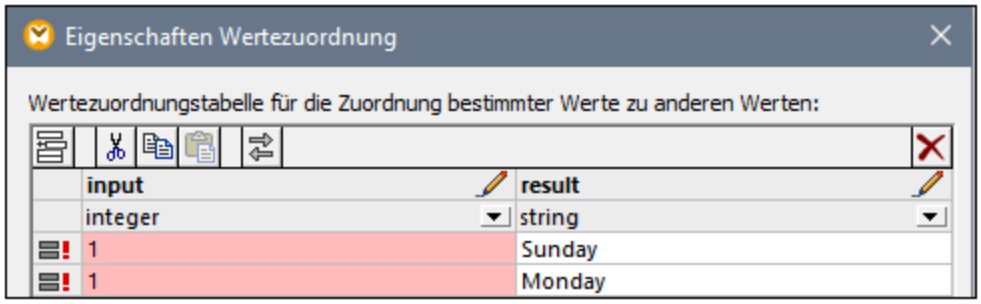
Befüllen einer Wertezuordnung

Sie können so viele Wertpaare wie nötig in einer Lookup-Tabelle definieren. Sie können die Werte manuell eingeben oder Tabellendaten aus Text-, CSV-, oder Excel-Tabellen hineinkopieren. In den meisten Fällen funktioniert es auch, wenn Sie Tabellen über einen gebräuchlichen Browser aus einer HTML-Seite kopieren und einfügen. Wenn Sie Daten aus Textdateien kopieren, müssen die Felder durch Tabulatorzeichen voneinander getrennt sein. Außerdem erkennt MapForce in den meisten Fällen durch Kommas oder Semikola getrennten Text.

Beachten Sie bei der Erstellung von Lookup-Tabellen die folgenden Punkte:

1. Alle Datenelemente in der linken Spalte müssen eindeutig sein. Andernfalls ließe sich nicht ermitteln, für welches Datenelement genau eine Entsprechung benötigt wird.
2. Datenelemente, die zur selben Spalte gehören, müssen denselben Datentyp haben. Sie können den Datentyp aus der Dropdown-Liste am oberen Rand der jeweiligen Spalte der Lookup-Tabelle auswählen. Um Boolesche Typen zu konvertieren, geben Sie den Text "true" oder "false" wortwörtlich ein. Ein Beispiel dazu finden Sie unter Beispiel: Ersetzen von Wochentagen 205.

Wenn MapForce in der Lookup-Tabelle ungültige Daten findet, wird eine Fehlermeldung angezeigt und die ungültigen Zeilen werden rosa markiert, z.B:



Um Daten aus einer externen Quelle in die Wertezuordnungskomponente zu importieren, gehen Sie folgendermaßen vor:

1. Wählen Sie im Ausgangsprogramm (z.B. Excel) die entsprechenden Zellen aus. Dabei kann es sich entweder um eine einzelne Spalte oder um zwei benachbarte Spalten handeln.
2. Kopieren Sie die Daten mit dem Befehl **Kopieren** des externen Programms in die Zwischenablage.
3. Klicken Sie in der Wertezuordnungskomponente auf die Zeile, vor der die Daten eingefügt werden sollen.
4. Klicken Sie in der Wertezuordnungskomponente auf die Schaltfläche **Tabelle aus der Zwischenablage einfügen** oder drücken Sie alternativ dazu **Strg+V** oder **Umschalt+Einfg**.

Anmerkung: Die Schaltfläche **Tabelle aus der Zwischenablage einfügen** ist nur dann aktiv, wenn Sie vorher Daten aus einer Ausgangsdatei kopiert haben (d.h. wenn sich in der Zwischenablage Daten befinden).

Wenn Ihre Zwischenablage mehrere Spalten enthält, werden nur Daten aus den beiden ersten Spalten in die Lookup-Tabelle eingefügt, weitere Spalten werden ignoriert. Wenn Sie Daten aus einer einzigen Spalte über vorhandene Werte einfügen, erscheint ein Kontextmenü, in dem Sie gefragt werden, ob die Daten aus der Zwischenablage als neue Zeilen eingefügt werden sollen oder ob die vorhandenen Zeilen überschrieben werden

 © 2019-2025 Altova GmbH

sollen. Stellen Sie daher sicher, dass die Zwischenablage nur eine Spalte und nicht mehrere enthält, wenn Sie vorhandene Werte in der Lookup-Tabelle überschreiben möchten, anstatt neue Zeilen einzufügen.

Um Zeilen manuell vor einer vorhandenen Zeile einzufügen, klicken Sie auf die betreffende Zeile und anschließend auf die Schaltfläche **Einfügen** .

Um eine vorhandene Zeile an eine andere Position zu verschieben, ziehen Sie die Zeile (noch oben oder nach unten) an die neue Position, während Sie die linke Maustaste gedrückt halten.

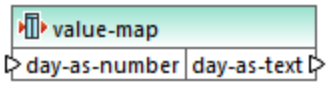
Um Zeilen zu kopieren oder auszuschneiden, um sie an einer anderen Position einzufügen, wählen Sie die Zeile zuerst aus und klicken Sie anschließend auf die Schaltfläche **Kopieren** (bzw. **Ausschneiden**). Sie können auch mehrere nicht unbedingt aufeinander folgende Zeilen kopieren oder ausschneiden. Halten Sie dazu beim Auswählen der Zeilen die **Strg**-Taste gedrückt. Beachten Sie, dass ausgeschnittener oder kopierter Text immer Werte aus beiden Spalten enthält. Sie können nicht Werte aus nur einer Spalte ausschneiden oder kopieren.

Um eine Zeile zu entfernen, klicken Sie darauf und anschließend auf die Schaltfläche **Löschen** .

Um die linke und die rechte Spalte zu vertauschen, klicken Sie auf die Schaltfläche **Spalten tauschen** .

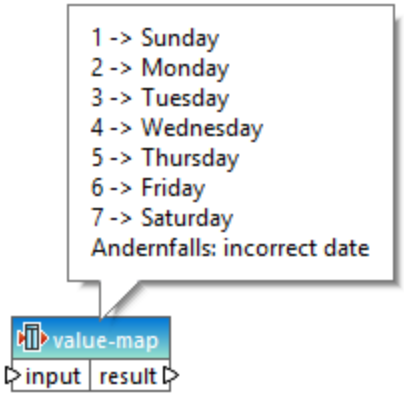
Umbenennen von Wertezuordnungsparametern

Standardmäßig hat der Input-Parameter einer Wertezuordnungskomponente den Namen "input" und der Output-Parameter den Namen "result". Um das Mapping übersichtlicher zu machen, haben Sie die Möglichkeit, jeden dieser Parameter durch Klicken auf die Schaltfläche **Bearbeiten** neben dem jeweiligen Namen umzubenennen. Im Folgenden sehen Sie ein Beispiel für eine Wertezuordnung mit benutzerdefinierten Parameternamen:



Vorschau auf eine Wertezuordnung

Nachdem Sie mit der Erstellung einer Wertezuordnung fertig sind, können Sie direkt über das Mapping schnell eine Vorschau darauf anzeigen, indem Sie die Maus über die Titelleiste der Komponente platzieren:

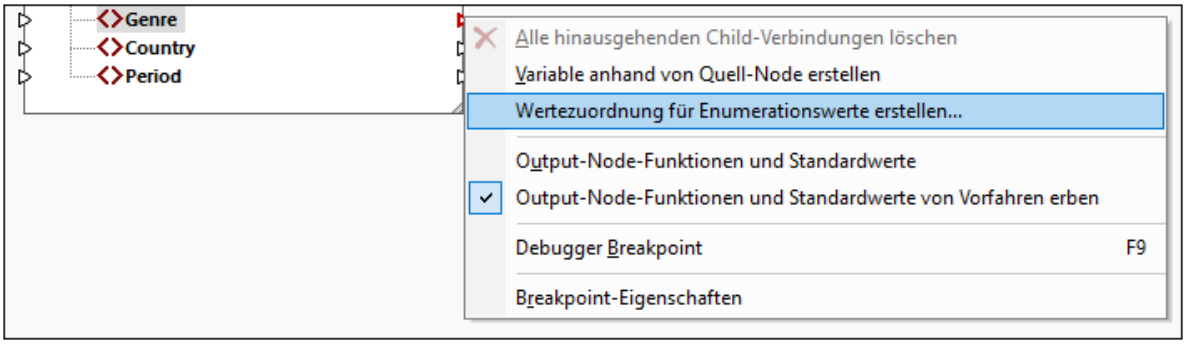


Erstellen einer Wertezuordnung anhand eines Enumerationstyps

Sie können in MapForce eine Wertezuordnung anhand von Nodes mit Enumerationswerten vornehmen. Diese Funktionalität wird derzeit für XML-Komponenten unterstützt, deren Nodes Enumeration Facets (*alle Editionen*) haben, und für EDI-Komponenten, deren Nodes EDI-Codelisten (*Enterprise Edition*) haben. Sie können eine solche Wertezuordnung je nach Bedarf anhand des Input- oder Output-Konnektors eines Node erstellen.

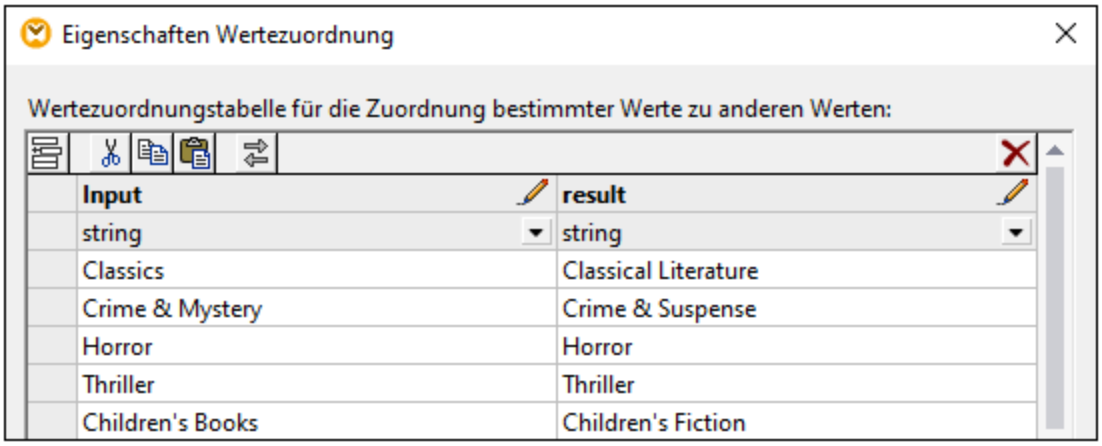
Um anhand eines Enumerationstyps eine Wertezuordnung zu erstellen, gehen Sie folgendermaßen vor:

1. Klicken Sie je nach Bedarf mit der rechten Maustaste auf den Input- oder Output-Konnektor des Node für die Enumerationswerte, anhand derer Sie eine Wertezuordnung erstellen möchten: In unserem Beispiel (*Abbildung unten*) haben wir den Output-Konnektor des Node `Genre` ausgewählt.



2. Wählen Sie im Kontextmenü den Befehl **Wertezuordnung für Enumerationswerte erstellen** aus.
3. Daraufhin wird das Dialogfeld **Eigenschaften Wertezuordnung** angezeigt. Beide Spalten `input` und `result` der Wertezuordnung enthalten dieselben vorausgefüllten Enumerationswerte. Sie können die Werte nun nach Bedarf überprüfen und bearbeiten. In der Abbildung unten sehen Sie die Liste der `Genre`-Werte aus der XML-Quelldatei (`input`) und die Liste der geänderten Werte, die gemappt werden sollen (`result`).

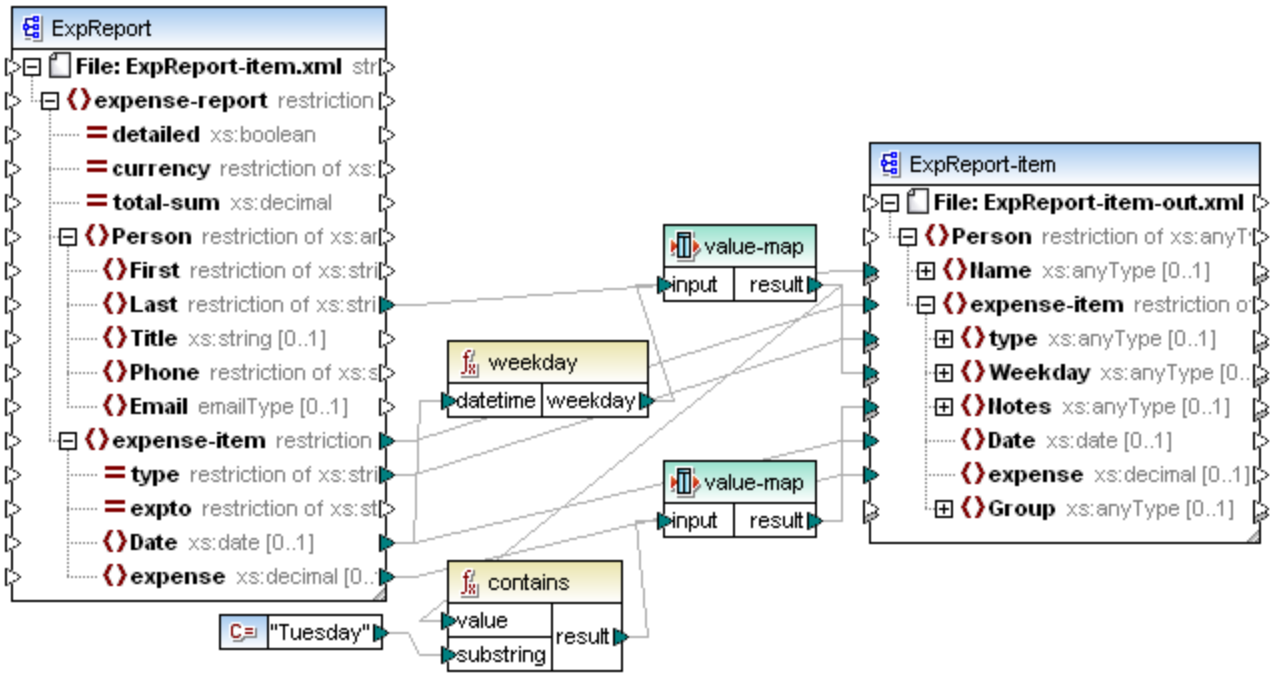
 © 2019-2025 Altova GmbH



4. Klicken Sie nach Überprüfung der Enumerationswerte auf **OK**. Dadurch wird die Wertezuordnungskomponente zum Mapping-Bereich hinzugefügt. Die Wertezuordnungskomponente wird automatisch mit dem Node verbunden, dessen Enumerationswerte zum Erstellen der Wertezuordnung verwendet wurden.
5. Verbinden Sie den anderen Parameter der Wertezuordnung mit dem entsprechenden Node und fahren Sie mit dem Design Ihres Mappings fort, wie gewohnt.

5.6.1 Beispiel: Ersetzen von Wochentagen

In diesem Beispiel sehen Sie eine Wertezuordnung, mit der Ganzzahlwerte durch Wochentagsnamen (1 = Sunday, 2 = Monday, usw.) ersetzt werden. Das Mapping zu diesem Beispiel befindet sich unter dem folgenden Pfad: **<Dokumente>\Altova\MapForce2025\MapForceExamples\Expense-valmap.mfd**.

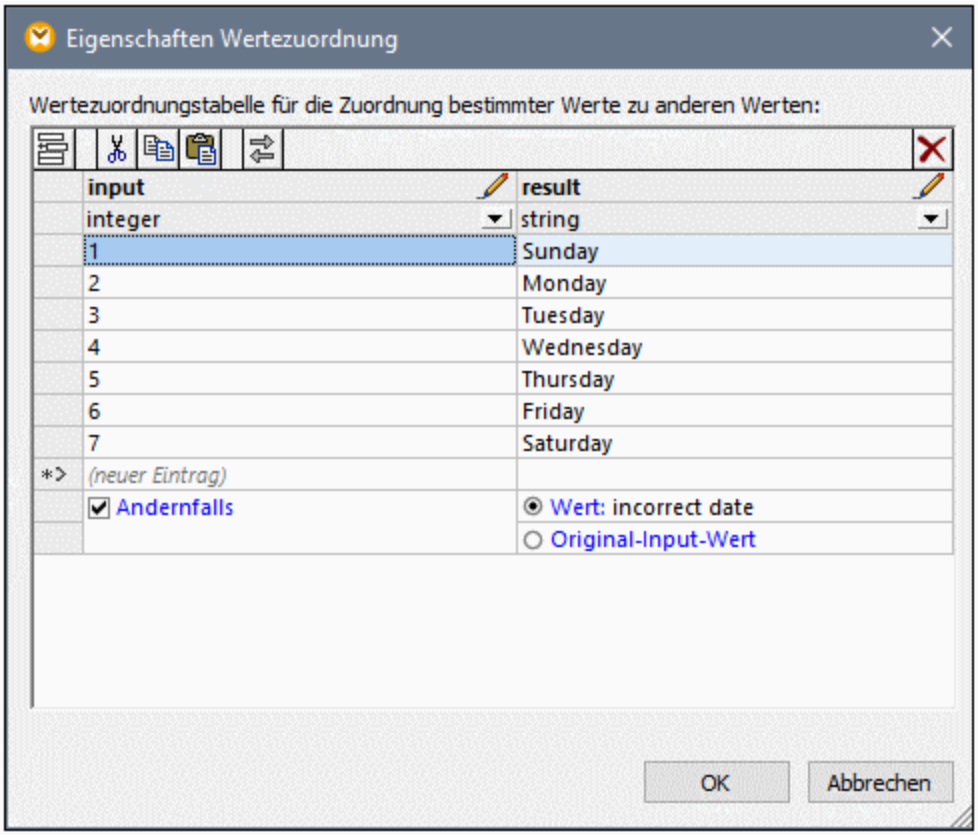


Expense-valmap.mfd

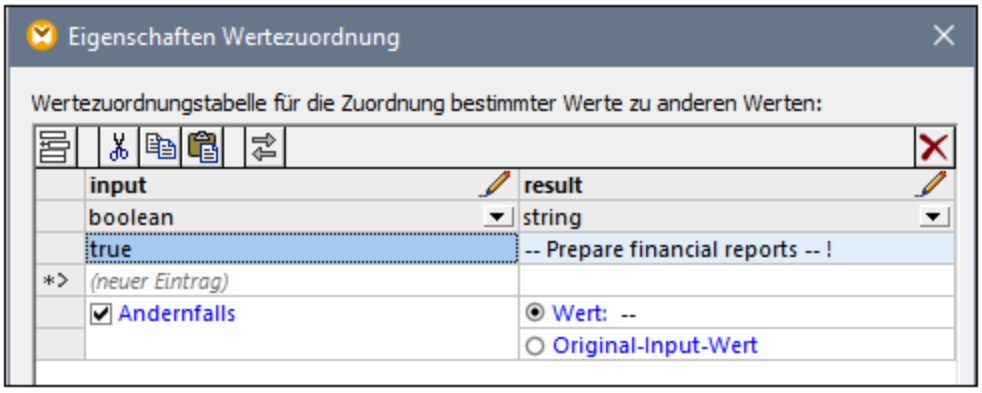
Der Wochentag wird aus dem Datenelement "**Date**" in der Datenquelle extrahiert, der numerische Wert wird in Text konvertiert und dieser Text wird in das Datenelement "**Weekday**" der Zielkomponente eingefügt. Dabei geschieht Folgendes:

- Die Funktion `weekday` extrahiert die Zahl für den Wochentag aus dem Datenelement **Date** in der Quelldatei. Das Ergebnis dieser Funktion sind Ganzzahlen von 1 bis 7.
- Die erste Wertezuordnungskomponente transformiert die Ganzzahlen in Wochentage (1 = Sunday, 2 = Monday, usw.). Wenn ein ungültiger Ganzzahlwert außerhalb des Bereichs von 1-7 gefunden wird, wird der Text "incorrect date" (falsches Datum) zurückgegeben.

 © 2019-2025 Altova GmbH



- Wenn der Wochentag den Wert "Tuesday" enthält, so wird der Text "Prepare Financial Reports" in das Datenelement "**Notes**" in der Zielkomponente geschrieben. Dies erfolgt mit Hilfe der Funktion `contains`, die einen Booleschen Wert (**true** oder **false)** an eine zweite Wertezuordnungskomponente übergibt. Die zweite Wertezuordnung hat die folgende Konfiguration:



Die oben gezeigte Wertezuordnung ist folgendermaßen zu interpretieren:

- Immer, wenn der Boolesche Wert **true** ist, muss der Wert in den Text "-- Prepare financial reports -- ! ". konvertiert werden. In allen anderen Fällen wird der Text "--" zurückgegeben.

Beachten Sie, dass der Datentyp der ersten Spalte als "boolean" definiert ist. Damit wird sichergestellt, dass der Boolesche Input-Wert **true** als Boolescher Wert erkannt wird.

5.6.2 Beispiel: Ersetzen von Stellenbezeichnungen

In diesem Beispiel wird gezeigt, wie Sie Werte von bestimmten Elementen in einer XML-Datei mit Hilfe von Wertezuordnungskomponenten (d.h. mit Hilfe einer vordefinierten Lookup-Tabelle) ersetzen.

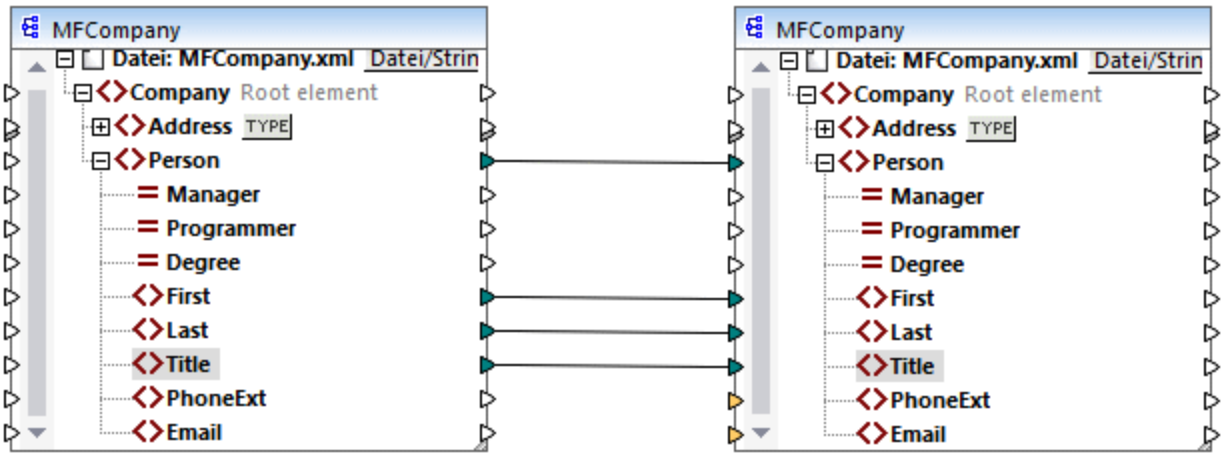
Sie finden die XML-Datei zu diesem Beispiel unter dem folgenden Pfad: **<Dokumente>\Altova\MapForce2025\MapForceExamples\MFCompany.xml**. Unter anderem sind darin Informationen über die Angestellten einer Firma und deren Stellenbezeichnungen gespeichert, z.B.:

```
<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>
```

Angenommen, Sie müssen einige der Stellenbezeichnungen in der obigen XML-Datei ersetzen. So soll etwa der Titel "Software Engineer" durch "Code Magician" und der Titel "Support Engineer" durch "Support Magician" ersetzt werden. Alle anderen Stellenbezeichnungen sollen unverändert bleiben.

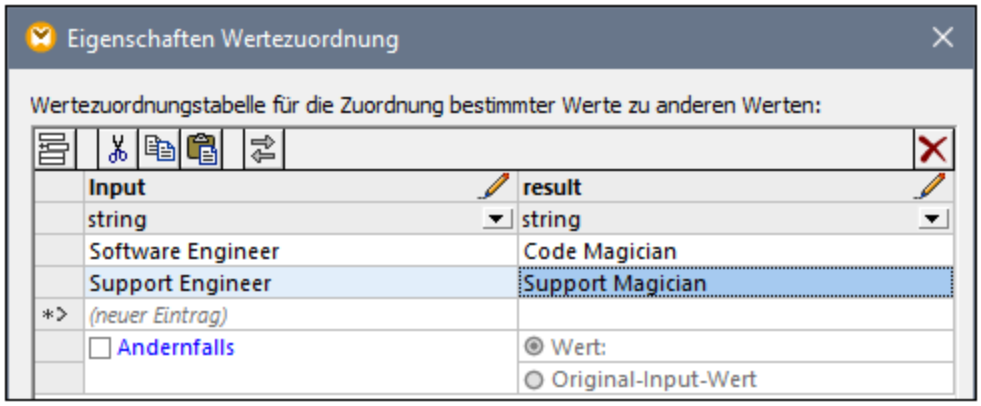
Fügen Sie für diese Aufgabe die XML-Datei durch Klicken auf die Symbolleisten-Schaltfläche **XML-Schema/Datei einfügen** oder Auswahl des Menübefehls **Einfügen | XML-Schema/Datei** zum Mapping hinzu. Kopieren Sie als nächstes die XML-Komponente im Mapping, fügen Sie sie ein und erstellen Sie die Verbindungen wie unten gezeigt. Eventuell müssen Sie zuerst die Symbolleisten-Schaltfläche **Automatische Verbindung von Sub-Einträge aktivieren/deaktivieren** deaktivieren, damit nicht benötigte Verbindungen nicht automatisch erstellt werden.

 © 2019-2025 Altova GmbH



Im bisher erstellten Mapping werden die **Person**-Elemente einfach in die XML-Zieldatei kopiert, ohne dass an den Elementen **First**, **Last** und **Title** irgendwelche Änderungen vorgenommen werden.

Um die benötigten Stellenbezeichnungen zu ersetzen, fügen Sie eine Wertezuordnungskomponente hinzu. Klicken Sie mit der rechten Maustaste auf die Verbindung zwischen den beiden **Title**-Elementen und wählen Sie im Kontextmenü den Befehl **Wertezuordnung einfügen.** Konfigurieren Sie die Wertezuordnungseigenschaften, wie unten gezeigt:

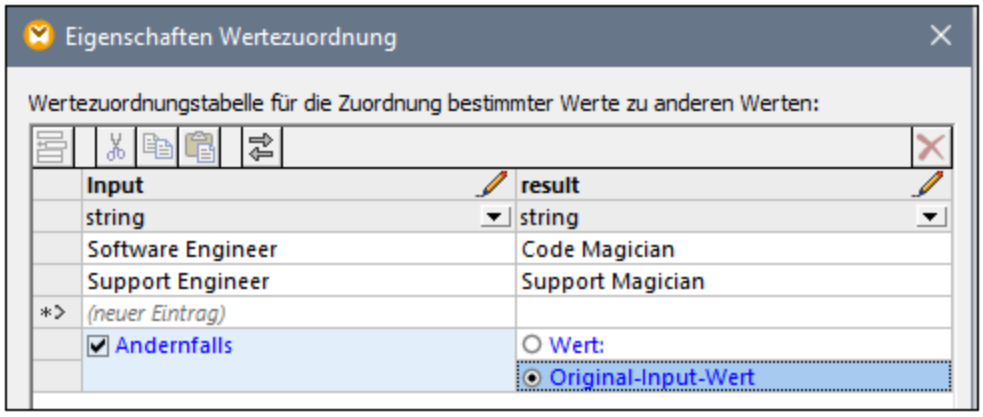


Gemäß der obigen Konfiguration wird jede Instanz von "Software Engineer" durch "Code Magician" und jede Instanz von "Support Engineer" durch "Support Magician" ersetzt. Beachten Sie, dass die **Andernfalls**-Bedingung noch nicht definiert wurde, daher gibt die Wertezuordnung überall dort, wo die Stellenbezeichnug eine andere als "Software Engineer" und "Support Engineer" ist, einen *leeren Node* zurück. Wenn Sie daher auf das Fenster Ausgabe klicken und eine Vorschau auf das Mapping anzeigen, fehlt bei einigen der Person-Elemente das Element **Title**, z.B.:

```
<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>
```

Wie zuvor erwähnt, verursachen fehlende Nodes fehlende Einträge in der generierten Ausgabe, daher wurde im obigen XML-Fragment nur bei Michelle Butler die Stellenbezeichnung (title) ersetzt, da ihre Stellenbezeichnung in der Lookup-Tabelle vorhanden war. Die bisher erstellte Konfiguration muss daher folgendermaßen ergänzt werden:

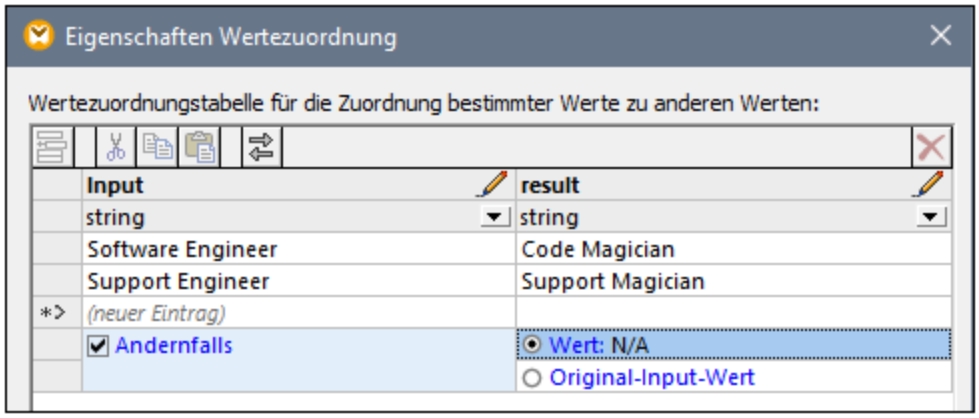


Mit der obigen Konfiguration geschieht zur Mapping-Laufzeit Folgendes:

- Jede einzelne Instanz von "Software Engineer" wird durch "Code Magician" ersetzt.
- Jede einzelne Instanz von "Support Engineer" wird durch "Support Magician" ersetzt.
- Wenn die ursprüngliche Stellenbezeichnung in der Lookup-Tabelle nicht gefunden wird, gibt die Wertezuordnung die Stellenbezeichnung unverändert zurück..

Zu Demonstrationszwecken können wir alle anderen Stellenbezeichnungen mit Ausnahme von "Software Engineer" und "Support Engineer", auch in einen benutzerdefinierten Wert, z.B. in "N/A" ändern. Definieren Sie dazu die Eigenschaften, wie unten gezeigt:

 © 2019-2025 Altova GmbH



Wenn Sie jetzt eine Vorschau auf das Mapping anzeigen, sind alle Stellenbezeichnungen in der Ausgabe vorhanden, doch weisen diejenigen ohne Entsprechung in der Lookup-Tabelle den Wert "N/A" auf, z.B.:

```
<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>
```

Damit ist das Wertezuordnungsbeispiel abgeschlossen. Durch Anwendung der oben gezeigten Logik können Sie das gewünschte Ergebnis nun auch in anderen Mappings erzielen.

5.7 Funktionen

Sie können Daten in MapForce mit Hilfe der folgenden Funktionskategorien transformieren:

- **Vordefinierte MapForce-Funktionen:** Diese Funktionen wurden in MapForce vordefiniert und Sie können damit in Ihren Mappings die verschiedensten Verarbeitungsaufgaben im Zusammenhang mit Strings, Datumswerten und anderen Datentypen durchführen. Des Weiteren können damit Gruppierungen, Aggregationen, automatische Nummerierung und verschiedene andere Aufgaben durchgeführt werden. Informationen zu den einzelnen vordefinierten Funktionen finden Sie unter Referenz Funktionsbibliothek 250.

- **Node-Funktionen und Standardwerte (*Professional und Enterprise Edition*):** Hierbei handelt es sich um speziellere Funktionen, mit denen Sie eine benutzerdefinierte Verarbeitungslogik erstellen und an einem oder mehreren untergeordneten Nodes einer Mapping-Komponente anwenden können. Sie können Daten damit entweder *vor* dem Erreichen eines Node einer Mapping-Struktur oder *unmittelbar nach* dem Verlassen des Node verarbeiten.

- **Benutzerdefinierte Funktionen (UDFs = user defined functions):** Es handelt sich hierbei um MapForce-Funktionen, die Sie selbst unter Verwendung verschiedener Komponentenarten und der bereits in MapForce verfügbaren vordefinierten Funktionen erstellen können, siehe Benutzerdefinierte Funktionen 219.

- **Spezielle benutzerdefinierte Funktionen:** Es handelt sich hierbei um Funktionen, die Sie aus externen Quellen wie XSLT-Bibliotheken importieren und an MapForce anpassen können. Damit diese Funktionen in MapForce wiederverwendet werden können, muss der Rückgabetyp dieser angepassten Funktionen ein simple type (wie String oder Ganzzahl) sein und auch die Parameter dieser Funktionen müssen den Typ simple type haben. Nähere Informationen dazu finden Sie unter Importieren benutzerdefinierter XSLT-Funktionen 239

- **Webservice-Aufrufe** (***Enterprise Edition***)**:** Sie können in MapForce einen zuvor definierten Webservice (einen WSDL-basierten Service oder eine generische HTTP API) aufrufen.

5.7.1 Grundlegendes zu Funktionen

Die folgenden Unterabschnitte enthalten eine Übersicht über grundlegende Aktionen im Zusammenhang mit Funktionen. Welche Funktionen im Fenster **Bibliotheken** angezeigt werden, hängt von der ausgewählten Transformationssprache ab. Nähere Informationen dazu finden Sie unter Transformationssprachen 16.

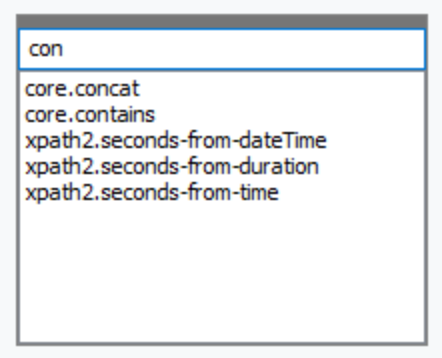
Hinzufügen einer Funktion

MapForce enthält eine große Zahl vordefinierter Funktionen, die Sie zum Mapping hinzufügen können. Nähere Informationen zu den einzelnen vordefinierten Funktionen finden Sie unter Referenz Funktionsbibliothek 250. Wählen Sie eine der folgenden Methoden, um eine Funktion zu einem Mapping hinzuzufügen:

- Klicken Sie im Fenster **Bibliotheken** auf die benötigte Funktion und ziehen Sie sie in den Mapping-Bereich. Um Funktionen nach Namen zu filtern, beginnen Sie mit der Eingabe des Funktionsnamens in das Textfeld im unteren Bereich des Fensters.
- Doppelklicken Sie in den leeren Bereich des Mappings und beginnen Sie mit der Eingabe des Funktionsnamens (*siehe Abbildung unten*). Um einen Tooltip mit näheren Informationen zu einer

 © 2019-2025 Altova GmbH

Funktion zu sehen, wählen Sie die Funktion in der Liste aus. Um eine Funktion zu Ihrem Mapping hinzuzufügen, doppelklicken Sie auf die entsprechende Funktion in der Auswahlliste.



Hinzufügen einer benutzerdefinierten Funktion
Sie können benutzerdefinierte Funktionen (UDFs) auf dieselbe Art, wie oben beschrieben, zum Mapping hinzufügen, vorausgesetzt (i) die UDF wurde bereits im selben Mapping erstellt oder (ii) Sie haben ein Mapping importiert, das UDFs als lokale oder globale Bibliothek enthält.

Hinzufügen einer Konstante

Mit Hilfe von Konstanten können Sie benutzerdefinierten Text und Zahlen zu einem Mapping hinzufügen. Wählen Sie eine der folgenden Optionen, um eine Konstante zu einem Mapping hinzuzufügen:

- Klicken Sie mit der rechten Maustaste in den leeren Mapping-Bereich und wählen Sie im Kontextmenü **Konstante einfügen**. Geben Sie den Wert ein und wählen Sie einen Datentyp aus: *String*, *Zahl* oder *alle anderen*.
- Klicken Sie auf dem Menübefehl **Einfügen | Konstante**. Geben Sie den Wert ein und wählen Sie einen Datentyp aus: *String*, *Zahl* oder *alle anderen*.
- Klicken Sie auf die Symbolleisten-Schaltfläche **Konstante**. Geben Sie den Wert ein und wählen Sie einen Datentyp aus: *String*, *Zahl* oder *alle anderen*.
- Doppelklicken Sie in den leeren Bereich des Mappings. Geben Sie ein doppeltes Anführungszeichen, gefolgt vom Wert der Konstante ein. Das schließende Anführungszeichen ist optional. Um eine numerische Konstante einzufügen, geben Sie einfach die Zahl ein.

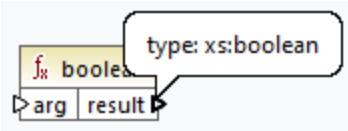
Suchen nach einer Funktion

Um im Fenster **Bibliotheken** nach einer Funktion zu suchen, geben Sie die ersten Zeichen des Funktionsnamens in das Textfeld im unteren Bereich des Fensters ein. Standardmäßig sucht MapForce nach Funktionsnamen und Beschreibungstext. Wenn Sie die Funktionsbeschreibung bei der Suche exkludieren möchten, klicken Sie auf den Nach unten-Pfeil und deaktivieren Sie die Option *In Funktionsbeschreibungen suchen*. Um die Suche abzubrechen, drücken Sie die **Esc**-Taste oder klicken Sie auf ✖.

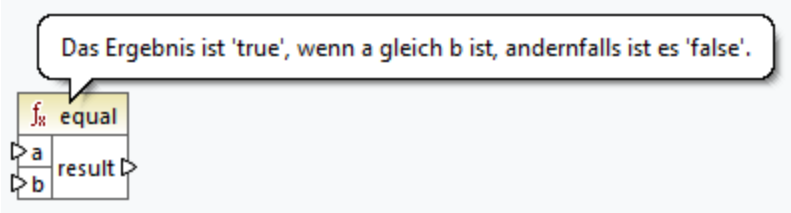
Um nach allen Instanzen einer Funktion im gerade aktiven Mapping zu suchen, klicken Sie im Fenster **Bibliotheken** mit der rechten Maustaste auf den Funktionsnamen und wählen Sie im Kontextmenü den Befehl **Alle Aufrufe suchen**. Die Suchergebnisse werden im Fenster **Meldungen** angezeigt.

Anzeigen von Typ und Beschreibung einer Funktion

Um den Datentyp eines Funktions-Input- oder -Output-Arguments zu sehen, platzieren Sie den Mauszeiger über den Argumentbereich einer Funktion (*siehe Abbildung unten*). Stellen Sie sicher, dass die Symbolleisten-Schaltfläche (**Tipps anzeigen**) aktiviert ist.

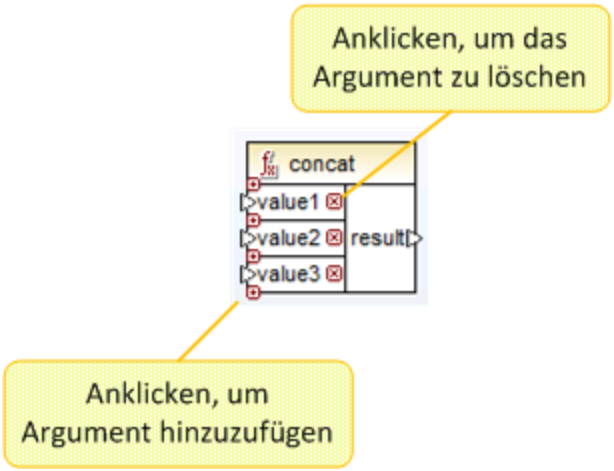


Um die Beschreibung einer Funktion anzuzeigen, platzieren Sie den Mauszeiger über die Titelleiste der Funktion (*siehe Abbildung unten*). Stellen Sie sicher, dass die Symbolleisten-Schaltfläche (**Tipps anzeigen**) aktiviert ist.



Hinzufügen/Löschen von Funktionsargumenten

Einige vordefinierte MapForce-Funktionen können erweitert werden, d.h. Sie können so viele Parameter, wie Sie für Ihre Mapping-Zwecke benötigen, hinzufügen. Ein gutes Beispiel dafür ist die Funktion **concat** 329. Um (bei Funktionen, die dies unterstützen) Funktionsargumente hinzuzufügen oder zu löschen, klicken Sie neben dem entsprechenden Parameter auf **Parameter hinzufügen** (⊞) bzw. **Parameter löschen** (⊠) (*siehe unten*). Wenn Sie eine Verbindung auf das Symbol ⊞ ziehen, wird ein weiterer Parameter hinzugefügt und verbunden.



 © 2019-2025 Altova GmbH

5.7.2 Verwalten von Funktionsbibliotheken

Sie können in MapForce die folgenden Arten von Bibliotheken in ein Mapping importieren und darin verwenden:

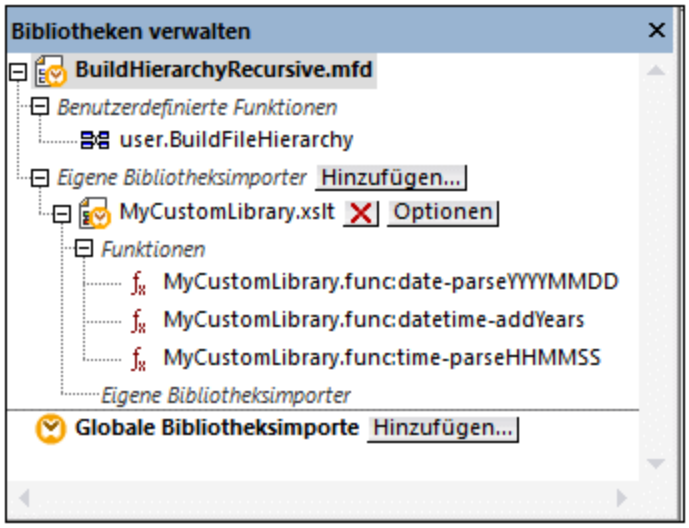
- Alle Mapping-Design-Dateien (*.mfd), die benutzerdefinierte Funktionen (UDFs) enthalten. Dies bezieht sich speziell auf Mapping-Dateien, die mit MapForce anhand von vordefinierten MapForce-Funktionen und Komponentenbausteinen erstellte benutzerdefinierte Funktionen (UDFs) enthalten. Nähere Informationen dazu finden Sie unter Benutzerdefinierte Funktionen 220.
- Benutzerdefinierte XSLT-Dateien, die Funktionen enthalten. Dies bezieht sich auf außerhalb von MapForce geschriebene XSLT-Funktionen, die sich wie unter Importieren benutzerdefinierter XSLT-Funktionen 239 beschrieben, für den Import in MapForce eignen.

Fenster "Bibliotheken verwalten"

Alle von einer Mapping-Datei verwendeten Bibliotheken können im Fenster "Bibliotheken verwalten" angezeigt und verwaltet werden. Dazu gehören auch benutzerdefinierte Funktionen (UDFs) und benutzerdefinierte Bibliotheken.

Standardmäßig wird das Fenster **Bibliotheken verwalten** nicht angezeigt. Um es anzuzeigen, wählen Sie eine der folgenden Methoden:

- Klicken Sie im Menü **Ansicht** auf **Bibliotheken verwalten**.
- Klicken Sie im unteren Bereich des Fensters **Bibliotheken** auf die Schaltfläche **Bibliotheken hinzufügen/entfernen**.



Sie können auswählen, ob benutzerdefinierte Funktionen (UDFs) und Bibliotheken nur für das gerade aktive Mapping-Dokument oder für alle geöffneten Mapping-Dokumente angezeigt werden sollen. Um die importierten Funktionen und Bibliotheken für alle gerade offenen Mapping-Dokumente anzuzeigen, klicken Sie mit der rechten Maustaste in das Fenster und wählen Sie im Kontextmenü den Befehl **Offene Dokumente anzeigen**.

Um anstelle des Namens den Pfad des geöffneten Mapping-Dokuments anzuzeigen, klicken Sie mit der rechten Maustaste in das Fenster und wählen Sie im Kontextmenü den Befehl **Dateipfade anzeigen**.

Die im Fenster "Bibliotheken verwalten" angezeigten Daten sind folgendermaßen hierarchisch gegliedert:

- Alle gerade offenen Mapping-Dokumente werden auf oberster Ebene angezeigt. Jeder Eintrag hat zwei Verzweigungen: **Benutzerdefinierte Funktionen** und **Eigene Bibliotheksimporte**.
  - Unter **Benutzerdefinierte Funktionen** werden alle in diesem Dokument enthaltenen UDFs angezeigt.
  - Unter **Eigene Bibliotheksdateien** werden *lokal* in das aktuelle Mapping-Dokument importierte Bibliotheken angezeigt. Mit dem Begriff "Bibliotheken" sind andere Mapping-Dokumente (.mfd-Dateien, die benutzerdefinierte Funktionen enthalten) oder externe in XSLT 1.0, XSLT 2.0, XQuery 1.0*, Java*, C#* geschriebene benutzerdefinierte Bibliotheken oder zuvor erwähnte .mff-Dateien gemeint. Beachten Sie, dass die Struktur **Eigene Bibliotheksimporte** mehrere Ebenen tief sein könnte, da in ein Mapping-Dokument wiederum weitere Mapping-Dokumente als Bibliothek importiert sein können.
- Der Eintrag **Globale Bibliotheksimporte** umfasst alle *global* auf Applikationsebene importierten benutzerdefinierten Bibliotheken. Auch im Fall von .mfd-Dateien könnte die Struktur aus den oben genannten Gründen mehrere Ebenen tief sein.

** Diese Sprachen werden nur in der MapForce Professional oder Enterprise Edition unterstützt.*

Anmerkung: Die XSLT-, XQuery-, C#- und Java-Bibliotheken können eigene Abhängigkeiten aufweisen. Solche Abhängigkeiten werden im Fenster "Bibliotheken" nicht angezeigt.

Kontextmenübefehle

Durch Rechtsklick auf ein Objekt und Auswahl einer der folgenden Kontextmenüoptionen können Sie verschiedene Operationen an Objekten im Fenster "Bibliotheken" ausführen.

Befehl	Beschreibung	Anwendbar auf
Öffnen	Öffnet das Mapping.	Mappings
Hinzufügen	Öffnet ein Dialogfeld, in dem Sie zu einer Bibliothek mit benutzerdefinierten Funktionen navigieren können.	Eigene Bibliotheksimporte
Sucht die Funktion im Fenster "Bibliotheken"	Ändert den Fokus in das Fenster "Bibliotheken" und wählt die Funktion aus.	Funktionen
Ausschneiden, Kopieren, Löschen	Diese Windows-Standardbefehle können nur auf benutzerdefinierte MapForce-Funktionen angewendet werden. Funktionen aus externen XSLT-Dateien oder anderen Bibliotheksarten können nicht kopiert und eingefügt werden.	Benutzerdefinierte Funktionen
Einfügen	Damit kann eine zuvor in die Zwischenablage kopierte benutzerdefinierte Funktion in die aktuelle Bibliothek eingefügt werden.	Bibliotheken (UDF)
Optionen	Öffnet ein Dialogfeld, in dem Sie Optionen für die aktuelle Bibliothek definieren oder ändern können.	Bibliotheken

 © 2019-2025 Altova GmbH

Befehl	Beschreibung	Anwendbar auf
Alle offenen Dokumente anzeigen	Wenn diese Option aktiviert ist, werden im Fenster "Bibliotheken verwalten" alle derzeit geöffneten Mappings angezeigt. Dies ist normalerweise dann nützlich, wenn Sie Funktionen zwischen Mappings kopieren und einfügen möchten. Andernfalls wird nur das Mapping angezeigt, auf dem sich gerade der Fokus befindet.	Immer
Dateipfade anzeigen	Wenn diese Option aktiviert ist, werden die Objekte im Fenster "Bibliotheken verwalten" mit ihrem vollständigen Dateipfad angezeigt. Andernfalls wird nur der Objektname angezeigt.	Immer

5.7.2.1 Lokale und globale Bibliotheken

Sie können Bibliotheken *lokal* oder *global* importieren. Globale Importe erfolgen auf Applikationsebene. Wenn eine Bibliothek global importiert wurde, können Sie ihre Funktionen von jedem Mapping aus verwenden.

Lokale Importe erfolgen auf Dateiebene. Angenommen, Sie entschließen sich bei der Arbeit am Mapping **A.mfd** dazu, alle benutzerdefinierten Funktionen aus dem Mapping **B.mfd** zu importieren. In diesem Fall gilt das Mapping **B.mfd** als lokale in das Mapping **A.mfd** importierte Bibliothek und Sie können Funktionen aus **B.mfd** auch in **A.mfd** verwenden. Auch wenn Sie Funktionen aus einer XSLT-Datei in **A.mfd** importieren, ist dies ein lokaler Import.

Alle lokalen und globalen Importe können im Fenster "Bibliotheken verwalten" angezeigt und verwaltet werden. Um eine Bibliothek zu importieren, wählen Sie eine der folgenden Methoden:

1. Klicken Sie im unteren Bereich des Fensters Bibliotheken 20 auf die Schaltfläche **Bibliotheken hinzufügen/entfernen**. Daraufhin wird das Fenster **Bibliotheken verwalten** geöffnet (*siehe Abbildung unten*).

2. Um Funktionen als *lokale* Bibliothek (nur im Geltungsbereich des aktuellen Dokuments) zu importieren, klicken Sie auf die Schaltfläche **Hinzufügen** unterhalb des aktuellen Mapping-Namens. Um Funktionen als *globale* Bibliothek (auf Programmebene zu importieren, klicken Sie auf die Schaltfläche **Hinzufügen** neben **Globale Bibliotheksimporte**. Wenn Sie eine Bibliothek *lokal* importieren, können Sie den Pfad zur Bibliotheksdatei als relativ zur Mapping-Datei definieren. Bei gobal importierten Bibliotheken ist der Pfad zur importierten Bibliothek immer absolut.

Miteinander in Konflikt stehende Funktionsnamen

Manchmal kommt es vor, dass derselbe Funktionsname auf jeder der folgenden Ebenen definiert ist.

- im Hauptmapping
- in einer lokal importierten Bibliothek
- in einer global importierten Bibliothek

Um Unklarheiten zu vermeiden, versucht MapForce in einem solchen Fall, die Funktion in genau der oben angeführten Reihenfolge aufzurufen. D.h. die direkt im Mapping definierte Funktion hat Vorrang, wenn derselbe Funktionsname in einer lokal importierten Bibliothek vorkommt. Außerdem hat die lokal importierte Funktion Vorrang vor der global importierten Funktion (vorausgesetzt beide Funktionen haben denselben Namen).

Wenn mehrere Funktionen desselben Namens vorhanden sind, wird nur die Funktion aufgerufen, die gemäß der obigen Regel Vorrang hat; andere nicht eindeutige Funktionsnamen werden blockiert. Diese blockierten Funktionen werden im Fenster "Bibliotheken" ausgegraut angezeigt und können im Mapping nicht verwendet werden.

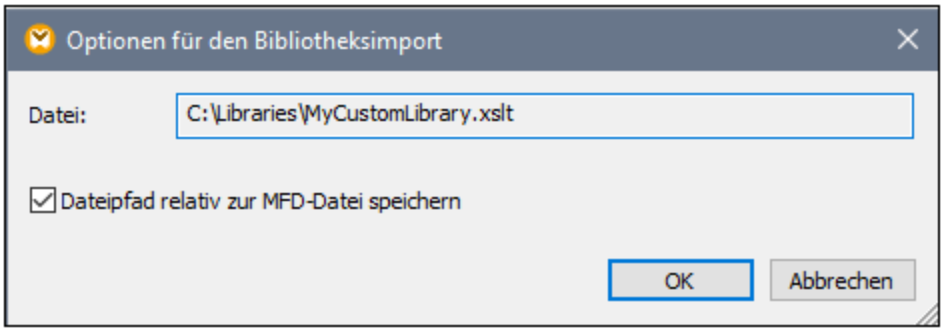
5.7.2.2 Relative Bibliothekspfade

Sie können den Pfad jeder importierten Bibliotheksdatei als relativ zur Mapping-Design-Datei (.mfd) definieren, vorausgesetzt, die Bibliothek wurde lokal (und nicht global) importiert, wie unter Lokale und globale Bibliotheken 217 beschrieben.

> Relative Bibliothekspfade können nur für *lokal* auf Dokumentebene importierte Bibliotheken definiert werden. Wenn ein Mapping *global* auf Programmebene importiert wurde, ist sein Pfad immer absolut.

So definieren Sie einen Bibliothekspfad als relativ zur Mapping-Design-Datei:

1. Klicken Sie im unteren Bereich des Fensters "Bibliotheken" auf die Schaltfläche **Bibliotheken hinzufügen/entfernen**. Daraufhin wird das Fenster "Bibliotheken verwalten" 22 geöffnet.
2. Klicken Sie neben der gewünschten Bibliothek auf **Optionen**. (Klicken Sie alternativ dazu mit der rechten Maustaste auf die Bibliothek und wählen Sie im Kontextmenü den Befehl **Optionen** aus).



3. Aktivieren Sie das Kontrollkästchen **Alle Dateipfade relativ zur MFD-Datei speichern**.

 © 2019-2025 Altova GmbH

Anmerkung: Wenn das Kontrollkästchen ausgegraut ist, vergewissern Sie sich, dass die Bibliothek tatsächlich lokal und nicht global importiert wurde.

Wenn das Kontrollkästchen aktiviert ist, aktualisiert MapForce den Pfad zu von der Komponente referenzierten Bibliotheksdateien, wenn Sie die Mapping-Datei mit dem Menübefehl **Speichern unter** in einem anderen Verzeichnis speichern. Wenn sich Bibliotheksdateien im selben Verzeichnis wie die Mapping-Datei befinden, funktioniert die Pfadreferenz weiterhin, wenn Sie das gesamte Verzeichnis in einen anderen Ordner auf dem Rechner verschieben, siehe auch Verwenden relativer Pfade in einer Komponente 40.

Beachten Sie, dass im Kontrollkästchen **Alle Dateipfade relativ zur MFD-Datei speichern** definiert ist, dass Pfade *relativ zur Mapping-Datei* sein sollen. Dies hat keine Auswirkung auf Pfade im generierten Code. Informationen zur Behandlung von Pfadreferenzen im generierten Code finden Sie unter Pfade in verschiedenen Ausführungsumgebungen 43.

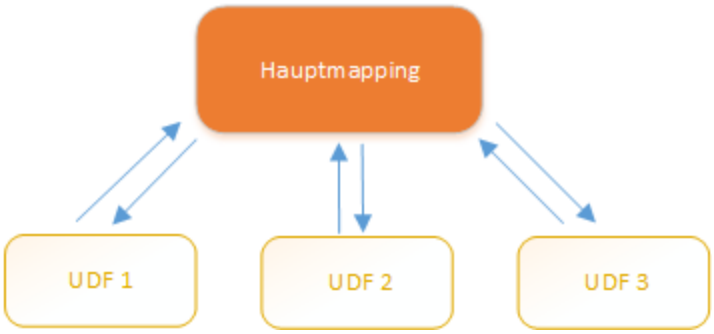
5.7.3 Benutzerdefinierte Funktionen

Benutzerdefinierte Funktionen (UDFs = User Defined Functions) sind eigene Funktionen, die der Benutzer einmal definiert und die anschließend im selben oder in mehreren Mappings wiederverwendet werden können. Benutzerdefinierte Funktionen bilden selbst eine Art von Mini-Mapping: Normalerweise bestehen sie aus einem oder mehreren Input-Parametern, einigen Datenverarbeitungszwischenkomponenten und einem Output zur Rückgabe der Daten an das aufrufende Mapping bzw. eine andere benutzerdefinierte Funktion, die diese aufruft.

Vorteile von benutzerdefinierten Funktionen

Benutzerdefinierte Funktionen haben folgende Vorteile:

- Sie können mehrmals in einem Mapping oder auch in mehreren Mappings verwendet werden.
- Benutzerdefinierte Funktionen machen Ihr Mapping übersichtlicher. So können darin etwa Teile eines Mappings in kleinere Bausteine verpackt werden, sodass die Implementierungsdetails ausgeblendet werden. Ein Beispiel dafür sehen Sie im nachstehenden Diagramm.

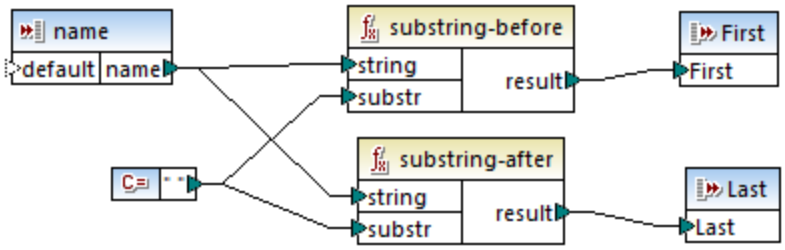


- Benutzerdefinierte Funktionen sind flexibel und dienen dazu, Strings, Zahlen, Datumsangaben und andere Daten auf eine bestimmte über vordefinierte MapForce-Funktionen hinausgehende Weise zu verarbeiten. So können Sie etwa Text auf eine bestimmte Art verketten oder aufteilen, komplexe Berechnungen durchführen, Datums- und Uhrzeitangaben bearbeiten usw.
- Eine weitere häufige Anwendungsmöglichkeit von benutzerdefinierten Funktionen ist, den Inhalt eines bestimmten Felds in einer XML-Datei, einer Datenbank oder einem anderen von Ihrer MapForce Edition unterstützten Datenformat abzurufen und diese Daten auf geeignete Weise darzustellen. Nähere Informationen dazu finden Sie unter Look-up-Implementierung 235.
- Benutzerdefinierte Funktionen können rekursiv aufgerufen werden (d.h. die benutzerdefinierte Funktion ruft sich selbst auf). Die benutzerdefinierte Funktion muss dazu als reguläre Funktion (nicht inline

gesetzte Funktion) 222 definiert sein. Mit Hilfe rekursiver benutzerdefinierter Funktionen 230 können Sie verschiedene komplexe Mapping-Aufgaben lösen, wie z.B. das Iterieren über Datenstrukturen mit einer Tiefe von *N* Children, wobei *N* im Vorhinein nicht bekannt ist.

Beispiel

Im Folgenden sehen Sie ein Beispiel für eine einfache benutzerdefinierte Funktion, die einen String in zwei separate Strings aufteilt. Diese benutzerdefinierte Funktion ist Teil eines größeren Mappings mit dem Namen **`MapForceExamples\ContactsFromPO.mfd`**. Sie erhält als Parameter einen Namen (z.B. `Helen Smith`), wendet die vordefinierten Funktionen **`substring-before`** und **`substring-after`** darauf an und gibt anschließend als Ergebnis zwei Werte zurück: `Helen` und `Smith`.



In diesem Abschnitt

In diesem Abschnitt wird erläutert, wie Sie mit benutzerdefinierten Funktionen arbeiten. Er ist in die folgenden Kapitel gegliedert:

- Benutzerdefinierte Funktionen: Grundlagen 220
- Parameter von benutzerdefinierten Funktionen 225
- Rekursive benutzerdefinierte Funktionen 230
- Look-up-Implementierung 235

5.7.3.1 Benutzerdefinierte Funktionen: Grundlagen

In diesem Kapitel wird beschrieben, wie Sie benutzerdefinierte Funktionen (kurz UDFs = User-Defined Functions) erstellen, importieren, bearbeiten, kopieren und einfügen und löschen.

Erstellen einer benutzerdefinierten Funktion

In diesem Unterabschnitt erfahren Sie, wie Sie eine benutzerdefinierte Funktion von Grund auf neu oder anhand bestehender Komponenten erstellen. Sie benötigen dazu mindestens eine Ausgabekomponente, mit der einige Daten verbunden sind. Als Input-Parameter kann eine Funktion null, einen oder mehrere Inputs haben. Die Input- und Output-Parameter können den Typ simpleType (z.B. String) oder complexType (eine Struktur) haben. Nähere Informationen zu einfachen und komplexen Parametern finden Sie unter Parameter von benutzerdefinierten Funktionen 225.

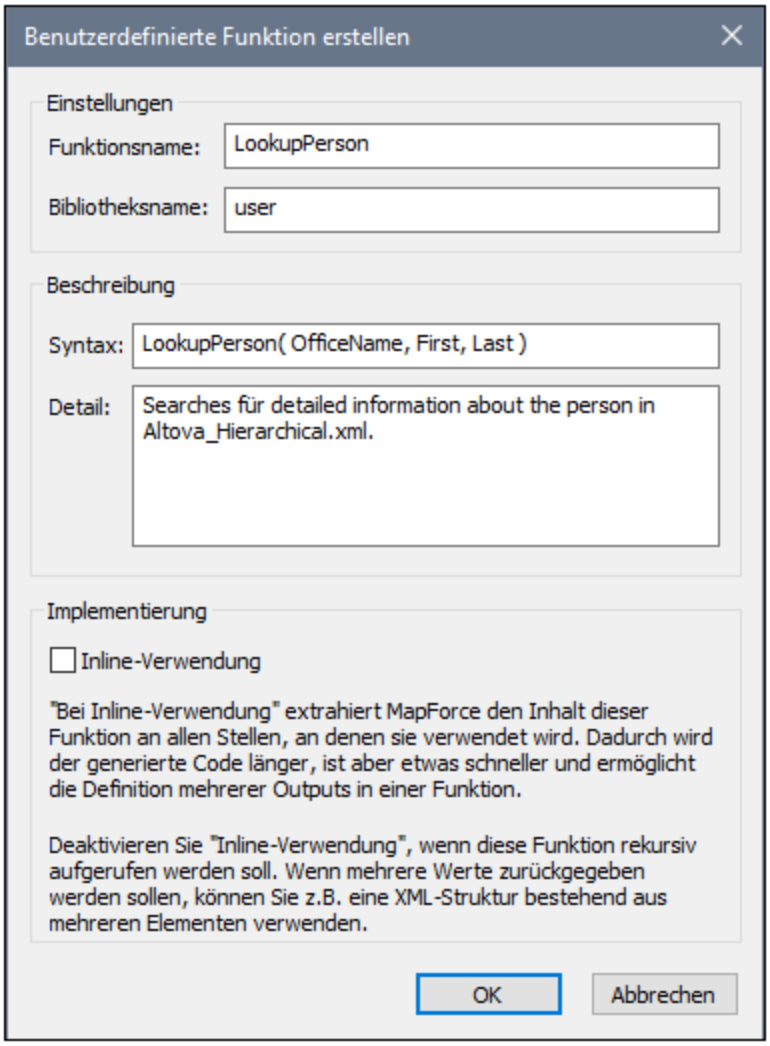
Neuerstellung einer benutzerdefinierten Funktion

Um eine benutzerdefinierte Funktion von Grund auf neu zu erstellen, gehen Sie folgendermaßen vor:

 © 2019-2025 Altova GmbH

1. Wählen Sie **Funktion | Benutzerdefinierte Funktion erstellen**. Klicken Sie alternativ dazu auf die Symbolleisten-Schaltfläche .

2. Geben Sie die entsprechenden Informationen in das Dialogfeld **Benutzerdefinierte Funktion erstellen** ein (*siehe Abbildung unten*).



Es stehen die folgenden Optionen zur Verfügung:

- *Funktionsname:* Obligatorisches Feld. Im Namen der benutzerdefinierten Funktion können die folgenden Zeichen verwendet werden: Alphanumerische Zeichen (a-z, A-Z, 0-9), ein Unterstrich (_), ein Bindestrich (-) und ein Doppelpunkt (:).
- *Bibliotheksname:* Obligatorisches Feld. Dies ist der Name einer Funktionsbibliothek (im Fenster "Bibliotheken" 20), in dem Ihre Funktion gespeichert wird. Wenn Sie keine Bibliothek angeben, wird die Funktion in eine Standardbibliothek namens `user` platziert.
- *Syntax:* Optionales Feld. Geben Sie hier Text ein, um die Syntax der Funktion genau zu beschreiben (z.B. die erwarteten Parameter). Dieser Text wird im Fenster **Bibliotheken** neben der Funktion angezeigt. Er hat keine Auswirkung auf die Implementierung der Funktion.
- *Detail:* Optionales Feld. Diese Beschreibung wird angezeigt, wenn Sie im Fenster **Bibliotheken** oder an anderen Stellen den Cursor über die Funktion platzieren.

- *Inline-Verwendung:* Aktivieren Sie dieses Kontrollkästchen, wenn die Funktion als Inline-Funktion erstellt werden soll. Nähere Informationen dazu finden Sie weiter unten unter *Reguläre benutzerdefinierte Funktionen und Inline-Funktionen*.

3. Klicken Sie auf **OK.** Die Funktion wird daraufhin sofort im Fenster **Bibliotheken** unter dem oben angegebenen Bibliotheksnamen angezeigt. Das Mapping-Fenster wird nun neu gezeichnet, damit Sie eine neue Funktion erstellen können (es handelt sich hierbei um ein eigenständiges Mapping, das so genannte *Funktionsmapping*). Das Funktionsmapping enthält standardmäßig eine Ausgabekomponente.

4. Fügen Sie alle erforderlichen Komponenten zum Funktionsmapping hinzu. Gehen Sie dabei auf die gleiche Weise vor, wie in einem Standard-Mapping.

Um die benutzerdefinierte Funktion in einem Mapping zu verwenden, ziehen Sie diese aus dem Fenster **Bibliotheken** in den Hauptmapping-Bereich. Siehe auch *Aufrufen und Importieren von benutzerdefinerten Funktionen* wieder unten.

Benutzerdefinierte Funktionen anhand bestehender Komponenten
Um eine benutzerdefinierte Funktion anhand bestehender Komponenten zu erstellen, gehen Sie folgendermaßen vor:

1. Wählen Sie die gewünschten Komponenten im Mapping aus, indem Sie mit der Maus ein Rechteck aufziehen. Sie können die Komponenten auch durch Anklicken bei gedrückter **Strg**-Taste auswählen.
2. Wählen Sie den Menübefehl **Funktion | Benutzerdefinierte Funktion von Auswahl erstellen**. Klicken Sie alternativ dazu auf die Symbolleisten-Schaltfläche .
3. Gehen Sie vor, wie in Schritt 2-4 unter *Neuerstellung einer benutzerdefinierten Funktion* beschrieben.

Reguläre benutzerdefinierte Funktionen und Inline-Funktionen
Es gibt zwei Arten von benutzerdefinierten Funktionen: *Reguläre Funktionen* und *Inline-Funktionen*. Sie können beim Erstellen der Funktion den Typ Ihrer benutzerdefinierten Funktion definieren. Inline-Funktionen und reguläre Funktionen verhalten sich in Bezug auf die Codegenerierung, Rekursivität und die Möglichkeit, mehrere Output-Parameter zu haben, unterschiedlich. Die Tabelle unten enthält eine Übersicht über die wichtigsten Unterschiede zwischen regulären benutzerdefinierten Funktionen und inline gesetzten benutzerdefinierten Funktionen.

Inline-Funktionen (strichlierter Rand)	**Reguläre Funktionen (durchgezogener Rand)**
Bei Inline-Funktionen wird der Code der benutzerdefinierten Funktion an allen Stellen, an denen die Funktion aufgerufen wird, eingefügt. Wenn die benutzerdefinierte Funktion mehrmals aufgerufen wird, würde der generierte Programmcode beträchtlich länger.	Der Code für die benutzerdefinierte Funktion wird einmal aufgerufen und die Inputs dazu werden als Parameterwerte übergeben. Wenn die benutzerdefinierte Funktion mehrmals aufgerufen wird, wird sie jedes Mal mit den entsprechenden Parameterwerten ausgewertet.
Inline-Funktionen können mehrere Outputs haben und daher als Resultat mehrere Werte haben.	Reguläre Funktionen können nur einen Output haben. Um mehrere Werte zurückzugeben, können Sie den Output als complexType (z.B. als XML-Struktur) deklarieren. Auf diese Art könnten mehrere Werte an die aufrufende Komponente zurückgegeben werden.

 © 2019-2025 Altova GmbH

Inline-Funktionen (strichlierter Rand)	Reguläre Funktionen (durchgezogener Rand)
Inline-Funktionen können nicht rekursiv aufgerufen werden.	Reguläre Funktionen können rekursiv aufgerufen werden.
Bei Inline-Funktionen kann kein Prioritätskontext 87 für einen Parameter definiert werden.	Bei regulären Funktionen kann ein Prioritätskontext für einen Parameter definiert werden.
Inline-Funktionen wirken sich auf den Kontext im Mapping, das die benutzerdefinierte Funktion aufruft, aus und umgekehrt wirkt sich der Kontext auf die benutzerdefinierte Funktion aus.	Reguläre benutzerdefinierte Funktionen haben ihren eigenen lokalen Kontext.

Anmerkung: Das Umschalten einer benutzerdefinierten Funktion zwischen "regulär" und "inline" kann sich auf den Mapping-Kontext 79 auswirken, wodurch im Mapping ein anderes Ergebnis erzeugt werden kann.

Aufrufen und Importieren von benutzerdefinierten Funktionen

Nachdem Sie eine benutzerdefinierte Funktion erstellt haben, können Sie diese entweder vom selben Mapping, in dem Sie diese erstellt haben, oder von einem beliebigen anderen Mapping aus aufrufen.

Aufrufen von benutzerdefinierten Funktionen über dasselbe Mapping
Um eine benutzerdefinierte Funktion vom selben Mapping aus aufzurufen, gehen Sie folgendermaßen vor:

1. Suchen Sie die gewünschte Funktion im Fenster **Bibliotheken** unter der Bibliothek, die Sie bei der Erstellung der Funktion definiert haben. Geben Sie dazu die ersten Buchstaben des Namens in der Fenster **Bibliotheken** ein.
2. Ziehen Sie die Funktion aus dem Fenster **Bibliotheken** in das Mapping. Sie können nun alle erforderlichen Parameter damit verbinden.

Importieren einer benutzerdefinierten Funktion aus einem anderen Mapping
Um eine benutzerdefinierte Funktion aus einem anderen Mapping zu importieren, gehen Sie folgendermaßen vor:

1. Klicken Sie im unteren Bereich des Fensters Bibliotheken 20 auf die Schaltfläche **Bibliotheken hinzufügen/entfernen**. Daraufhin wird das Fenster **Bibliotheken verwalten** geöffnet (*siehe Abbildung unten*).

2. Um Funktionen als *lokale* Bibliothek (nur im Geltungsbereich des aktuellen Dokuments) zu

importieren, klicken Sie auf die Schaltfläche **Hinzufügen** unterhalb des aktuellen Mapping-Namens. Um Funktionen als *globale* Bibliothek (auf Programmebene zu importieren, klicken Sie auf die Schaltfläche **Hinzufügen** neben **Globale Bibliotheksimporte**. Wenn Sie eine Bibliothek *lokal* importieren, können Sie den Pfad zur Bibliotheksdatei als relativ zur Mapping-Datei definieren. Bei gobal importierten Bibliotheken ist der Pfad zur importierten Bibliothek immer absolut.

3. Navigieren Sie zur **`.mfd`**-Datei, die die benutzerdefinierte Funktion enthält und klicken Sie auf **Öffnen**. Daraufhin erscheint ein Meldungsfeld, in dem Sie darüber informiert werden, dass eine neue Bibliothek hinzugefügt wurde und über das Fenster **Bibliotheken** aufgerufen werden kann.

Sie können nun jede der importierten Funktionen im aktuellen Mapping verwenden, indem Sie die Funktion aus dem **Bibliotheksfenster** in das Mapping ziehen. Nähere Informationen zum Anzeigen und Organisieren von Funktionsbibliotheken finden Sie unter Verwalten von Funktionsbibliotheken 215.

Mapping mit Anmeldeinformationen (Enterprise Edition)
Wenn die importierte **`.mfd`**-Datei Anmeldeinformationen enthält, werden diese im Anmeldeinformationen-Manager als importiert (gelb hinterlegt) angezeigt. Importierte Anmeldeinformationen werden standardmäßig nicht mit dem Mapping gespeichert. Sie können jedoch optional eine lokale Kopie anlegen und diese im Mapping speichern.

Bearbeiten von benutzerdefinierten Funktionen

Um eine benutzerdefinierte Funktion zu bearbeiten, gehen Sie folgendermaßen vor:

1. Öffnen Sie das Mapping, das die benutzerdefinierte Funktion enthält.
2. Doppelklicken Sie im Mapping auf die Titelleiste der benutzerdefinierten Funktion, um den Inhalt der Funktion zu sehen. Sie können Komponenten darin je nach Bedarf hinzufügen, bearbeiten oder entfernen.
3. Um die Eigenschaften der Funktion (wie Name oder Beschreibung) zu ändern, klicken Sie mit der rechten Maustaste in einen leeren Bereich des Mappings und wählen Sie im Kontextmenü den Befehl **Funktionseinstellungen**. Klicken Sie alternativ dazu auf die Symbolleisten-Schaltfläche .

Sie können eine Funktion auch durch Doppelklick auf ihren Namen im Fenster **Bibliotheken** bearbeiten, doch können nur Funktionen im derzeit aktiven Dokument auf diese Art geöffnet werden. Wenn Sie auf eine benutzerdefinierte Funktion doppelklicken, die in einem anderen Mapping erstellt wurde, so wird dieses Mapping in einem neuen Fenster geöffnet. Wenn Sie eine benutzerdefinierte Funktion, die in mehrere Mappings importiert wurde, bearbeiten oder löschen, so wirkt sich diese Änderung auf alle diese Mappings aus.

Um zum Hauptmapping zurückzugelangen, klicken Sie in der linken oberen Ecke des Mapping-Fensters auf die Schaltfläche .

Mit Hilfe der Symbolleisten-Schaltflächen und können Sie in MapForce durch verschiedene Mappings und benutzerdefinierte Funktionen navigieren. Die Tastaturkürzel für diese Schaltflächen sind **Alt+nach links** bzw. **Alt+nach rechts**.

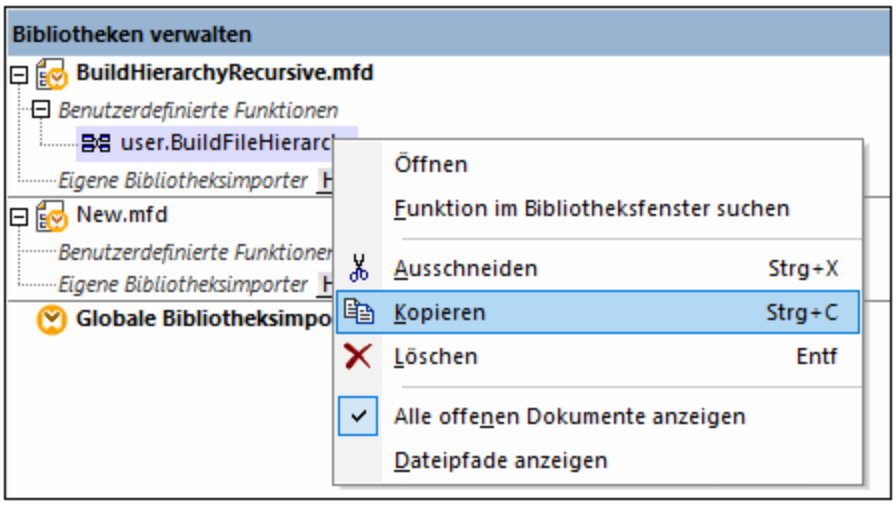
Kopieren und Einfügen von benutzerdefinierten Funktionen

Um eine benutzerdefinierte Funktion zu kopieren und in ein anderes Mapping einzufügen, gehen Sie folgendermaßen vor:

1. Öffnen Sie das Fenster "Bibliotheken verwalten" 215.

 © 2019-2025 Altova GmbH

2. Klicken Sie mit der rechten Maustaste auf einen leeren Bereich im Fenster **Bibliotheken** und wählen Sie die Option **Alle offenen Dokumente anzeigen**.
3. Öffnen Sie das Quell- und das Zielmapping. Beide Mappings müssen unbedingt gespeichert worden sein, damit die Pfade korrekt aufgelöst werden. Siehe auch Kopieren-Einfügen und relative Pfade 41.
4. Klicken Sie mit der rechten Maustaste im Fenster **Bibliotheken verwalten** auf die gewünschte benutzerdefinierte Funktion aus dem Quellmapping und wählen Sie im Kontextmenü den Befehl **Kopieren** (*siehe Abbildung unten*) oder drücken Sie **Strg+C**. Lassen Sie des Fenster **Bibliotheken verwalten** geöffnet.



5. Wechseln Sie in das Zielmapping (das Fenster **Bibliotheken verwalten** ändert sich entsprechend), klicken Sie mit der rechten Maustaste auf *Benutzerdefinierte Funktionen* und wählen Sie im Kontextmenü den Befehl **Einfügen**.

Löschen von benutzerdefinierten Funktionen

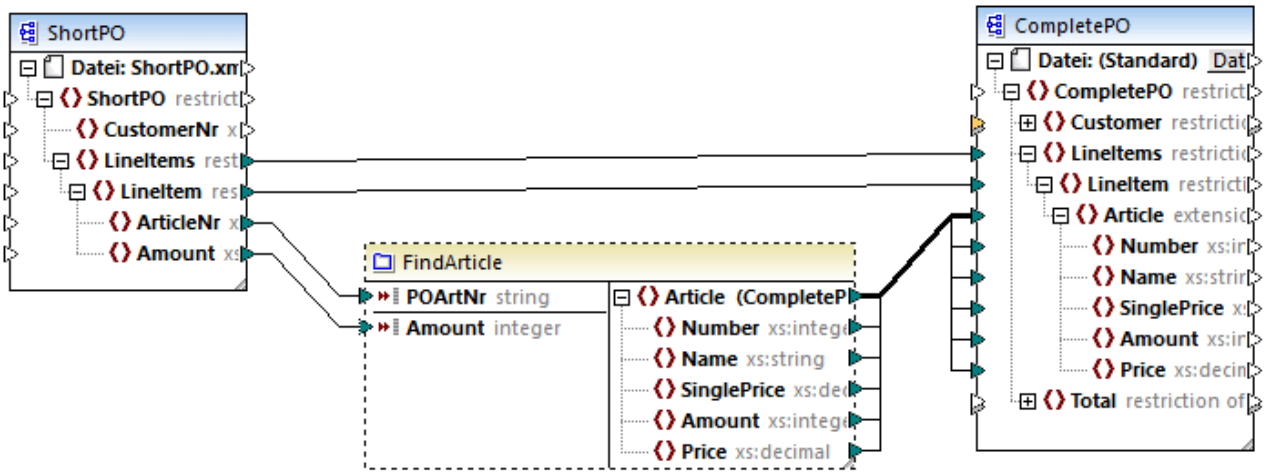
Um eine benutzerdefinierte Funktion zu löschen, gehen Sie folgendermaßen vor:

1. Doppelklicken Sie im Mapping auf die Titelleiste der Komponente.
2. Klicken Sie in der rechten oberen Ecke des Mappingfensters auf die Schaltfläche .
3. Falls die Funktion im aktuell geöffneten Mapping verwendet wird, werden Sie gefragt, ob alle Instanzen mit der internen Komponente gelöscht werden sollen. Klicken Sie auf **Ja**, wenn die Funktion gelöscht werden soll und alle Instanzen, in denen diese aufgerufen wird, durch die Komponenten der Funktion ersetzt werden sollen. Auf diese Art bleibt das Hauptmapping gültig, auch wenn die Funktion gelöscht wird. Wenn die gelöschte Funktion jedoch in anderen externen Mappings verwendet wird, werden diese ungültig. Klicken Sie auf **Nein**, wenn Sie die Funktion und alle ihre internen Komponenten permanent löschen möchten. In diesem Fall werden alle Mappings, in denen die Funktion verwendet wird, ungültig.

5.7.3.2 Parameter von benutzerdefinierten Funktionen

Wenn Sie eine benutzerdefinierte Funktion erstellen, müssen Sie angeben, welche Input-Parameter (falls überhaupt) diese erhalten soll und welchen Output die Funktion erzeugen soll. Während Input-Parameter manchmal nicht benötigt werden, ist ein Output-Parameter in jedem Fall erforderlich.. Funktionsparameter können einen simpleType (wie z.B. `String` oder `Ganzzahl`) oder eine komplexe Struktur 226 haben. So hat etwa die unten gezeigte benutzerdefinierte Funktion `FindArticle` zwei Input- und einen Output-Parameter:

- `POArtNr` ist ein Input-Parameter vom simpleType `string`.
- `Amount` ist ein Input-Parameter vom Typ `integer`.
- `CompletePO` ist ein Output-Parameter mit einer komplexen XML-Struktur.



Parameterreihenfolge
Wenn eine benutzerdefinierte Funktion mehrere Input- oder Output-Parameter hat, können Sie die Reihenfolge ändern, in der die Parameter für aufrufende Komponenten dieser Funktion erscheinen. Die Reihenfolge der Parameter im Funktionsmapping (von oben nach unten) bestimmt die Reihenfolge, in der diese für aufrufende Komponenten dieser Funktion angezeigt werden.

Achtung

- Input- und Output-Parameter werden nach ihrer Position von oben nach unten gereiht. Wenn Sie also den Parameter `input3` im Funktionsmapping an die oberste Stelle verschieben, wird dieser zum ersten Parameter dieser Funktion.
- Wenn zwei Parameter dieselbe vertikale Position haben, so wird zuerst der am weitesten links liegende Parameter verarbeitet.
- In den seltenen Fällen, in denen zwei Parameter genau die gleiche Position haben, wird automatisch die interne Komponenten-ID verwendet.

Strukturen vom Typ "complexType"

In der Liste unten sehen Sie, auf welchen Strukturen ein Parameter in einer benutzerdefinierten Funktion basieren kann.

MapForce Basic Edition
- XML-Schema-Struktur

MapForce Professional Edition
- XML-Schema-Struktur
- Datenbankstruktur

MapForce Enterprise Edition
- XML-Schema-Struktur
- Datenbankstruktur
- EDI-Struktur
- FlexText Structure

 © 2019-2025 Altova GmbH

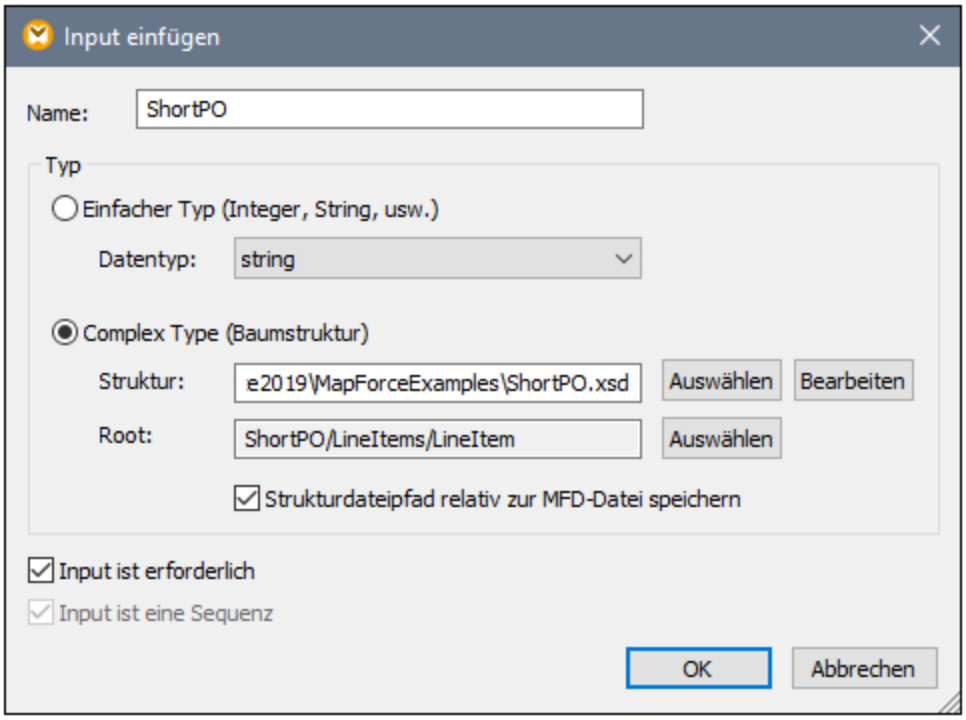
- JSON-Schema-Struktur

Auf Datenbankstrukturen basierende benutzerdefinierte Funktionen (Professional und Enterprise Edition)
Sie können in MapForce datenbankbasierte Parameter von benutzerdefinierten Funktionen mit einer Struktur damit in Zusammenhang stehender Tabellen erstellen und verwenden. Die Struktur dieser Tabellen bildet eine speicherresidente Struktur, die keine Verbindung zur Datenbank zur Laufzeit hat. Das heißt auch, dass es keine automatische Behandlung von Sekundärschlüsseln und keine Tabellenaktionen in Parametern oder Variablen gibt.

Hinzufügen von Parametern

Um einen Input- oder Output-Parameter hinzuzufügen, gehen Sie folgendermaßen vor:

1. Erstellen Sie eine benutzerdefinierte Funktion 220 oder öffnen Sie eine vorhandene 223.
2. Wählen Sie den Menübefehl **Funktion | Input-Komponente einfügen** bzw. **Funktion | Output-Komponente einfügen** (*siehe Abbildung unten*). Klicken Sie alternativ dazu in der Symbolleiste auf die Schaltfläche (**Input-Komponente einfügen**) oder (**Output-Komponente einfügen**).



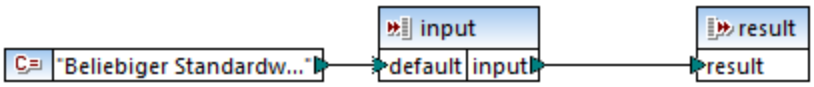
3. Wählen Sie beim Input- bzw. Output-Parameter um einen simpleType oder einen complexType handeln soll (*siehe Dialogfeld oben*). Siehe die Liste der verfügbaren komplexen Strukturen weiter oben. Um einen Parameter zu erstellen, der ein komplexer XML-Typ ist, klicken Sie neben *Struktur* auf **Auswählen** und navigieren Sie zum XML-Schema, das die erforderliche Struktur beschreibt.

Wenn das Funktionsmapping bereits XML-Schemas enthält, stehen sie als Strukturen zur Auswahl. Andernfalls können Sie ein komplett neues Schema für die Struktur des Parameters auswählen. Dasselbe gilt für Datenbanken und andere komplexe Strukturen, falls diese von Ihrer MapForce Edition unterstützt werden. Sie können bei XML-Strukturen ein Root-Element für Ihre Struktur auswählen, wenn es das XML-Schema erlaubt. Um ein Root-Element zu definieren, klicken Sie neben *Root* auf **Auswählen** und wählen Sie die Root im daraufhin angezeigten Dialogfeld aus.

Falls das Kontrollkästchen *Strukturdateipfad relativ zur MFD-Datei speichern* aktiviert ist, wird der absolute Pfad der Strukturdatei beim Speichern des Mappings in einen Pfad umgewandelt, der relativ zum aktuellen Mapping ist. Nähere Informationen dazu finden Sie unter Relative und absolute Pfade 40. Eine Erläuterung zu den Kontrollkästchen *Input ist erforderlich* und *Input ist eine Sequenz* finden Sie weiter unten.

Input ist erforderlich
Um einen Parameter in einer benutzerdefinierten Funktion zu einem obligatorischen Parameter zu machen, aktivieren Sie das Kontrollkästchen *Input ist erforderlich* (*siehe Dialogfeld oben*). Wenn Sie das Kontrollkästchen *Input ist erforderlich* deaktivieren, wird der Parameter optional und im Mapping mit einem strichlierten Rahmen angezeigt.

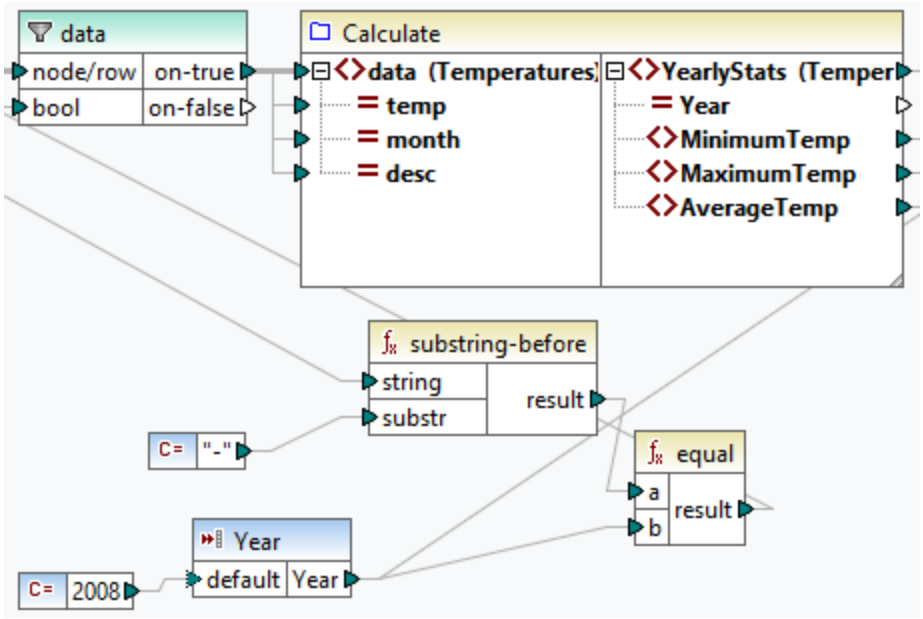
Außerdem können Sie einen Standardparameterwert definieren, indem Sie ihn mit dem Input "`default`" eines Parameters verbinden (*siehe Beispiel unten*). Der Standardwert wird nur wirksam, wenn kein anderer Wert vorhanden ist. Wenn der optionale Parameter bei Aufruf der Funktion einen Wert erhält, so hat dieser Wert Vorrang vor dem Standardwert.



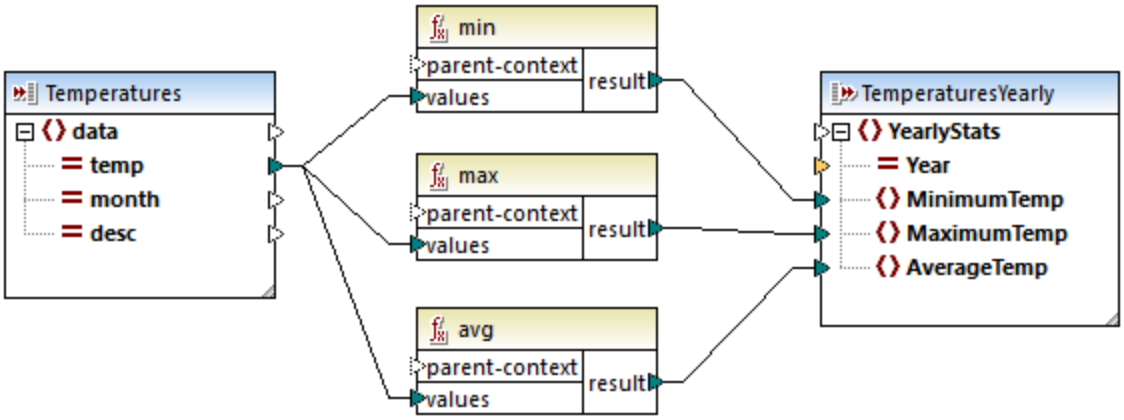
Input ist eine Sequenz
Sie können optional festlegen, ob ein Funktionsparameter als Einzelwert (Standardverhalten) oder als Sequenz behandelt werden soll. Eine Sequenz ist ein Bereich von null oder mehr Werten. Eine Sequenz kann sich als nützlich erweisen, wenn in Ihrer benutzerdefinierten Funktion Input-Daten in Form einer Sequenz erwartet werden, um in dieser Sequenz Werte zu berechnen (z.B. durch Aufruf von Funktionen wie **`avg`**, **`min`**, **`max`**. Damit der Input des Parameters als Sequenz behandelt wird, aktivieren Sie das Kontrollkästchen *Input ist eine Sequenz*. Beachten Sie, dass dieses Kontrollkästchen nur dann aktiv ist, wenn es sich um eine reguläre 222 benutzerdefinierte Funktion handelt.

Ein Beispiel für die Verwendung einer Sequenz sehen Sie im folgenden Mapping: **`MapForceExamples\InputIsSequence.mfd`**. In dem Ausschnitt aus diesem Mapping (*siehe Abbildung unten*) ist der Filter `data` mit der benutzerdefinierten Funktion `Calculate` verbunden. Das Ergebnis des Filters ist eine Sequenz von Datenelementen, sodass der Input-Parameter der Funktion als Sequenz definiert wurde.

 © 2019-2025 Altova GmbH



Unten sehen Sie die Implementierung der Funktion `Calculate`, die alle Sequenzwerte aggregiert: Sie führt, die Funktionen **`avg`**, **`min`**, **`max`** an der Input-Sequenz aus. Um die interne Struktur der Funktion `Calculate` zu sehen, doppelklicken Sie im Mapping oben auf die Überschrift der `Calculate`-Komponente.



Im Allgemeinen gilt, dass es von den Input-Daten, bei denen es sich um eine Sequenz oder nicht um eine Sequenz handelt, abhängt, wie oft die Funktion aufgerufen wird:

- Wenn Input-Daten mit einem *Sequenzparameter* verbunden sind, wird die benutzerdefinierte Funktion nur *einmal* aufgerufen und die vollständige Sequenz wird an die benutzerdefinierte Funktion übergeben.
- Wenn Input-Daten mit einem *Nicht-Sequenzparameter* verbunden sind, wird die benutzerdefinierte Funktion *für jedes Datenelement in der Sequenz einmal* aufgerufen.
- Wenn Sie eine leere Sequenz mit einem Nicht-Sequenz-Parameter verbinden, wird die Funktion gar nicht aufgerufen. Dies kann vorkommen, wenn die Quellstruktur optionale Datenelemente hat oder wenn eine Filterbedingung keine übereinstimmenden Datenelemente zurückgibt. Um dies zu vermeiden, verwenden Sie vor dem Funktions-Input entweder die Funktion `substitute-missing` 325,

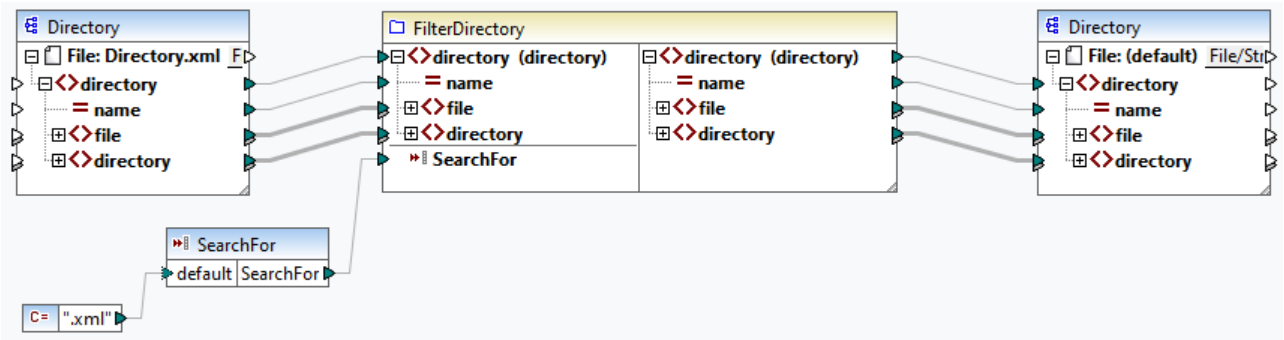
um sicherzustellen, dass die Sequenz nie leer ist oder setzen Sie den Parameter auf Sequenz und fügen Sie innerhalb der Funktion eine Behandlung für die leere Sequenz hinzu.

Das Kontrollkästchen *Output ist eine Sequenz* kann auch für Output-Parameter erforderlich sein. Wenn eine Funktion eine Sequenz aus mehreren Werten an ihre Output-Komponente übergibt und die Output-Komponente nicht auf Sequenz gesetzt ist, gibt die Funktion nur das erste Datenelement in der Sequenz zurück.

5.7.3.3 Rekursive benutzerdefinierte Funktionen

In diesem Beispiel wird beschrieben, wie Sie mit Hilfe einer rekursiven benutzerdefinierten Funktion nach Daten in einer XML-Quelldatei suchen. Um die rekursive benutzerdefinierte Funktion auszuprobieren, benötigen Sie das folgende Mapping: **`MapForceExamples\RecursiveDirectoryFilter.mfd`**. Im unten gezeigten Mapping erhält die benutzerdefinierte Funktion `FilterDirectory` Daten aus der Quelldatei `Directory.xml` und der einfachen Input-Komponente `SearchFor`, die die `.xml`-Erweiterung liefert. Nachdem die Daten von der benutzerdefinierten Funktion verarbeitet wurden, werden sie auf die Zieldatei gemappt.

Im Hauptmapping (*siehe Abbildung unten*) ist das allgemeine Mapping-Layout beschrieben. Wie die benutzerdefinierte Funktion die Daten verarbeitet, ist separat im Funktionsmapping definiert (*siehe UDF-Implementierung weiter unten*).



Aufgabenstellung
Ziel ist es, Dateien mit der Erweiterung **`.xml`** in der Ausgabe aufzulisten, wobei die gesamte Verzeichnisstruktur erhalten bleiben soll. In den folgenden Unterabschnitten wird das Mapping im Detail erläutert.

Quelldatei

Unten sehen Sie einen Ausschnitt aus der XML-Quelldatei (**`Directory.xml`**), die Informationen über Dateien und Verzeichnisse enthält. Beachten Sie, dass die Dateien in dieser Liste unterschiedliche Erweiterungen haben (z.B. **`.xml`**, **`.dtd`**, **`.sps`**). Gemäß dem Schema (**`Directory.xsd`**), kann das Element `directory` `file`- und `directory`-Children haben. Alle `directory`-Elemente sind rekursiv.

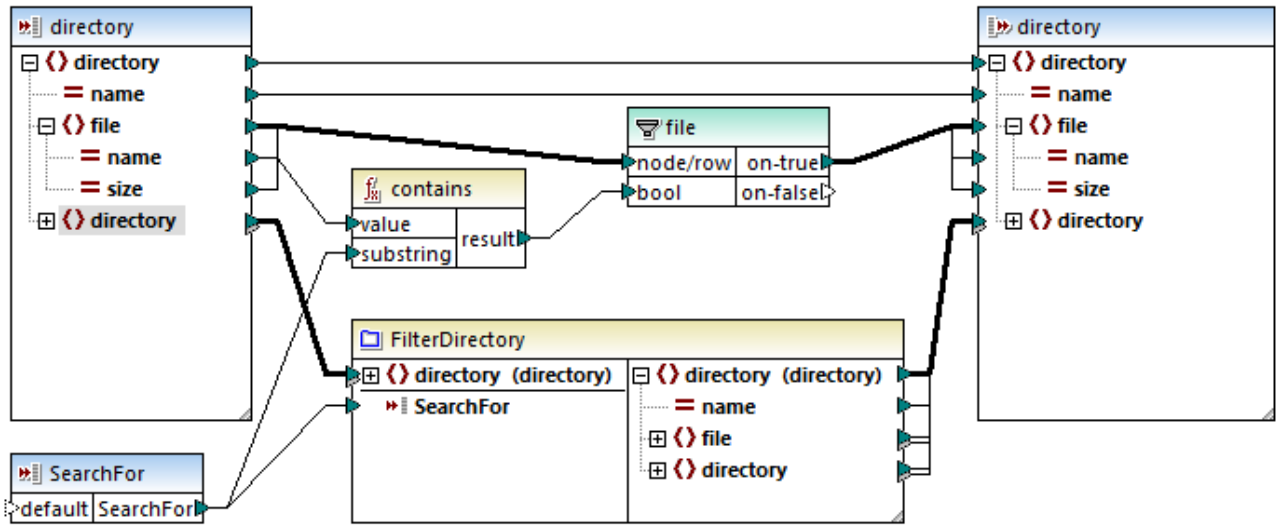
```
<directory name="ExampleSite">
  <file name="blocks.sps" size="7473"/>
  <file name="blocks.xml" size="670"/>
  <file name="block_file.xml" size="992"/>
  <file name="block_schema.xml" size="1170"/>
  <file name="contact.xml" size="453"/>
  <file name="dictionaries.xml" size="206"/>
```

 © 2019-2025 Altova GmbH

```
  <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>
```

Implementierung der benutzerdefinierten Funktion

Um die interne Implementierung der benutzerdefinierten Funktion zu sehen, doppelklicken Sie im Hauptmapping auf ihre Titelleiste. Die Funktion ist rekursiv, d.h. sie ruft sich selbst auf. Da die Funktion mit dem rekursiven Element `directory` verbunden ist, wird diese Funktion so oft aufgerufen, wie sich in der XML-Quellinstanz verschachtelte `directory`-Elemente befinden. Damit rekursive Aufrufe unterstützt werden, muss die Funktion regulär 222 sein.



Die Implementierung der benutzerdefinierten Funktion erfolgt in zwei Phasen: (i) der Definition der Dateien und (ii) der Definition der zu durchsuchenden Verzeichnisse.

Definition der Dateien

Die Dateien werden von der benutzerdefinierten Funktion folgendermaßen verarbeitet: Die Funktion **`contains`** überprüft, ob der erste String (der Dateiname) den (von der einfachen Input-Komponente `SearchFor`) gelieferten Substring `.xml` enthält. Wenn das Ergebnis der Funktion `true` ist, wird der Dateiname mit einer `.xml`-Erweiterung in die Ausgabe geschrieben.

Verarbeitung von untergeordneten Verzeichnissen
Die untergeordneten Verzeichnisse des aktuellen Verzeichnisses werden als Input an die aktuelle benutzerdefinierte Funktion gesendet. Die benutzerdefinierte Funktion iteriert somit durch alle `directory`-Elemente und überprüft, ob Dateien mit der Erweiterung **`.xml`** vorhanden sind.

Ausgabe

Wenn Sie auf das Fenster **Ausgabe** klicken, werden nur Dateien mit der Erweiterung **`.xml`** angezeigt (*siehe Codefragment unten*).

```
<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>
```

5.7.3.4 Kontext in benutzerdefinierten Funktionen

Benutzerdefinierte Funktionen (UDFs = User Defined Functions) sind eigens erstellte Funktionen, die in ein Mapping eingebettet werden, in denen Sie die Inputs, Outputs und Verarbeitungslogik definieren. Jede benutzerdefinierte Funktion kann dieselben Komponentenarten wie ein Hauptmapping, darunter auch Webservices und Datenbanken, enthalten.

> Wenn eine UDF eine Datenbank oder einen Webservice enthält und es sich bei den Input-Daten für die UDF um eine Sequenz aus mehreren Werten handelt, wird die UDF standardmäßig mit jedem Input-Wert aufgerufen, was einen Datenbank- oder Webservice-Aufruf zur Folge hat.

Das oben beschriebene Verhalten ist unter Umständen in Mappings, in denen die UDF wirklich so oft, wie Input-Werte vorhanden sind, aufgerufen werden muss, und in denen es keine andere Methode gibt, akzeptabel.

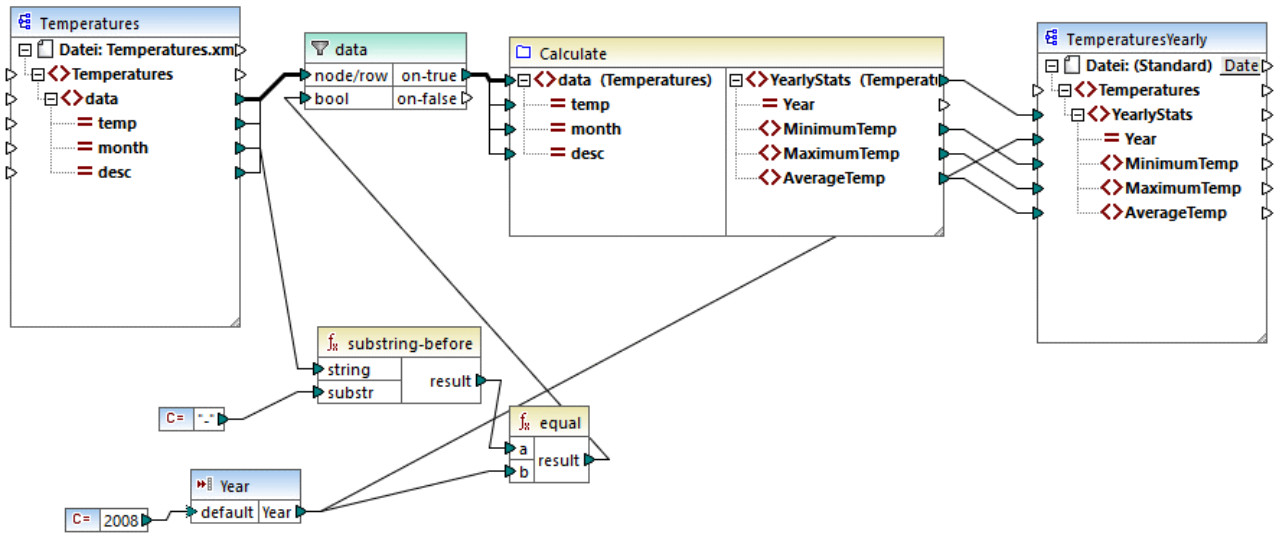
Wenn Sie das oben beschriebene Verhalten vermeiden möchten, können Sie die UDF so konfigurieren, dass sie, selbst wenn sie als Input eine Sequenz von Werten erhält, nur einmal aufgerufen wird. Normalerweise empfiehlt sich dies bei den UDFs, die an einer Gruppe von Werte ausgeführt werden, bevor sie ein Ergebnis zurückgeben (z.B. Funktionen, die Durchschnittswerte oder Summen berechnen).

Eine UDF so zu konfigurieren, dass sie im selben Funktionsaufruf mehrere Input-Werte akzeptiert, ist dann möglich, wenn es sich bei der UDF um eine reguläre UDF und nicht um eine "Inline"-UDF handelt (Nähere Informationen dazu finden Sie im Kapitel Benutzerdefinierte Funktionen 219.) Bei regulären Funktionen können Sie durch Aktivierung des Kontrollkästchens **Input ist eine Sequenz** definieren, dass der Input-Parameter eine Sequenz ist. Dieses Kontrollkästchen finden Sie in den Komponenteneinstellungen, die angezeigt werden, wenn Sie auf die Titelleiste eines Input-Parameters doppelklicken. Das Kontrollkästchen wirkt sich folgendermaßen darauf aus, wie oft die Funktion aufgerufen wird:

 © 2019-2025 Altova GmbH

- Wenn Input-Daten mit einem **Sequenz**-Parameter verbunden werden, wird die benutzerdefinierte Funktion *nur einmal* aufgerufen und die vollständige Sequenz wird an die benutzerdefinierte Funktion übergeben.
- Wenn Input-Daten mit einem **Nicht-Sequenz**-Parameter verbunden werden, wird die benutzerdefinierte Funktion *für jedes einzelne Datenelement in der Sequenz nur einmal* aufgerufen.

Ein Beispiel dazu finden Sie im folgenden Demo-Mapping:
<Dokumente>\Altova\MapForce2025\MapForceExamples\InputIsSequence.mfd.

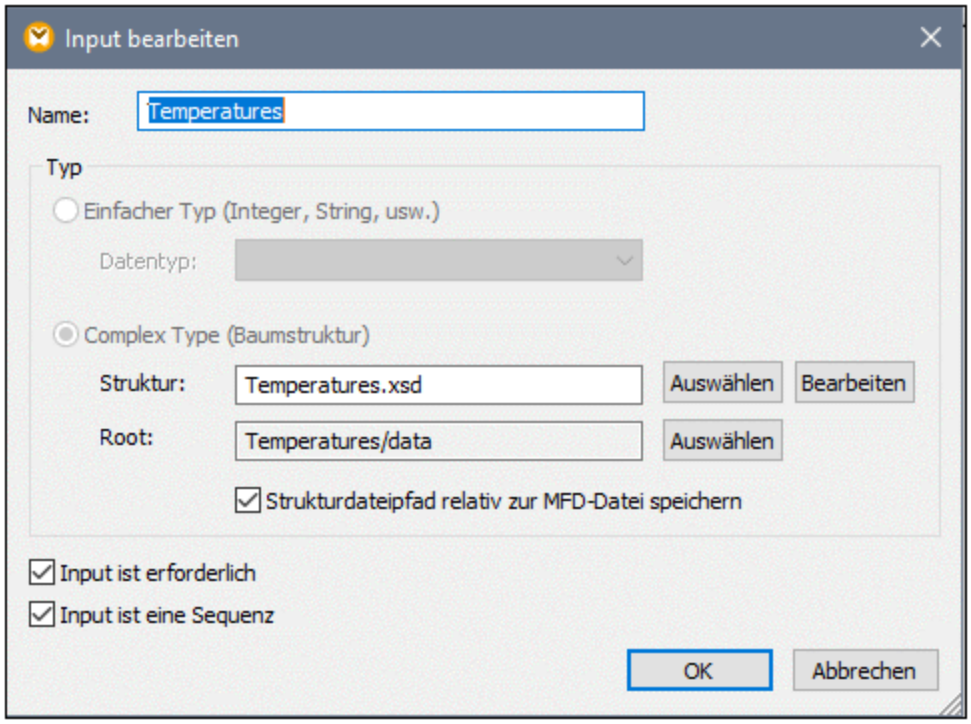


Im obigen Mapping sehen Sie ein Beispiel für einen typischen Fall einer UDF, die an einer Gruppe von Werten ausgeführt wird und für die daher alle Input-Werte in einem einzigen Aufruf abgerufen werden müssen. Die benutzerdefinierte Funktion `Calculate` gibt hier die Temperaturminima, -maxima und die Durchschnittstemperaturen zurück. Die Input-Daten dafür stammen aus einer XML-Quelldatei. Die erwartete Mapping-Ausgabe sieht folgendermaßen aus:

```
<Temperatures>
    <YearlyStats Year="2008">
        <MinimumTemp>-0.5</MinimumTemp>
        <MaximumTemp>24</MaximumTemp>
        <AverageTemp>11.6</AverageTemp>
    </YearlyStats>
</Temperatures>
```

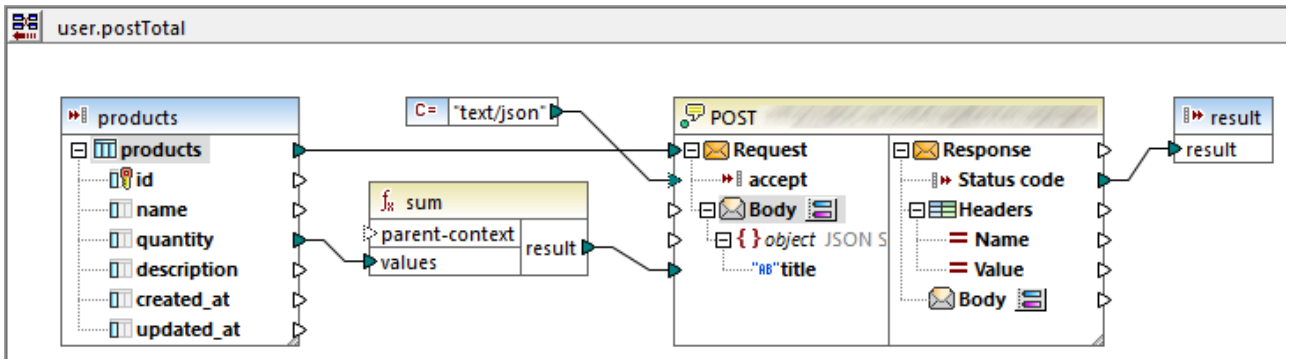
Wie gewöhnlich beginnt die Mapping-Ausführung beim obersten Datenelement der Zielkomponente (in diesem Beispiel **YearlyStats)**. Um diesen Node zu befüllen, versucht das Mapping die Quelldaten aus der UDF zu holen, welche wiederum den Filter auslöst. Die Rolle des Filters ist es, nur Temperaturen aus dem Jahr 2008 an die UDF zu übergeben.

Für den Input-Parameter der UDF wurde das Kontrollkästchen **Input ist eine Sequenz** aktiviert (um dieses Kontrollkästchen zu sehen, doppelklicken Sie auf die Titelleiste der Funktion `Calculate,` um das Mapping der Funktion aufzurufen. Doppelklicken Sie anschließend auf die Titelleiste des Input-Parameters). Wie zuvor erwähnt, wird aufgrund der Option **Input ist eine Sequenz** die vollständige Sequenz der Werte als Input an die Funktion geliefert und die Funktion wird nur einmal aufgerufen.



Wäre das Kontrollkästchen **Input ist eine Sequenz** nicht aktiviert, wäre die UDF für jeden Wert in der Quellkomponente einzeln aufgerufen worden, sodass die Minima, Maxima und Durchschnittstemperaturen für jeden Wert einzeln berechnet würden, wodurch eine falsche Ausgabe erzeugt würde.

Durch Anwendung derselben Logik in komplexeren UDFs, die Datenbank- oder Webservice-Aufrufe enthalten, kann die Ausführung unter Umständen optimiert und unnötige Datenbank- oder Webservice-Aufrufe können vermieden werden. Bedenken Sie jedoch, dass das Kontrollkästchen **Input ist eine Sequenz** keinen Einfluss darauf hat, was mit der Wertesequenz *nach* ihrem Eintritt in die Funktion passiert, d.h. die eingehende Wertesequenz könnte auch mit dem Input eines Webservice verbunden werden, wodurch dieser mehrmals aufgerufen würde. Werfen Sie einen Blick auf das folgende Beispiel:

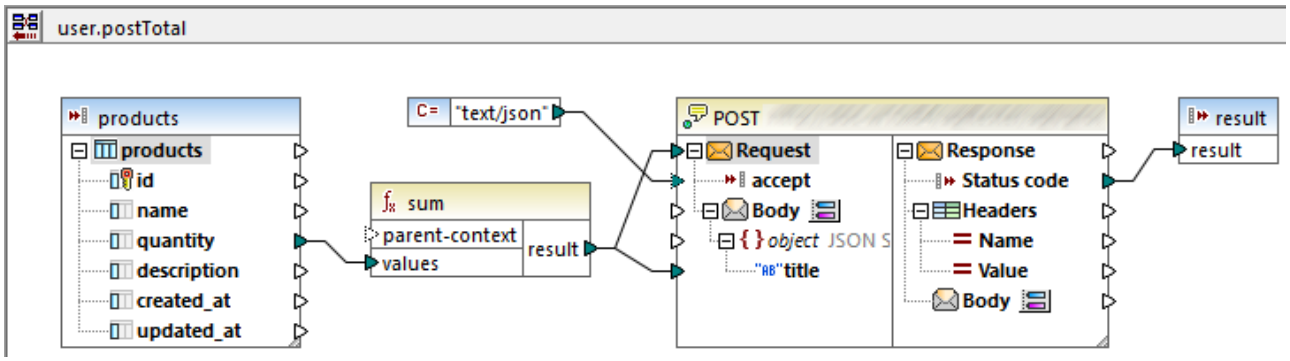


Die oben gezeigte UDF erhält aus dem externen Mapping eine Sequenz von Werten. Die an den Input-Parameter gelieferten Daten stammen in diesem Fall aus einer Datenbank. Für den Input-Parameter wurde die Option **Input ist eine Sequenz** aktiviert, daher wird die gesamte Sequenz in einem einzigen Aufruf an die Funktion geliefert. Mit der Funktion sollen mehrere **quantity**-Werte addiert und an einen Webservice gesendet

 © 2019-2025 Altova GmbH

werden. Es wird genau ein Webservice-Aufruf erwartet. Bei Ausführung des Mappings wird der Webservice jedoch fehlerhafter Weise mehrmals aufgerufen. Der Grund dafür ist, dass der **Request**-Input des Webservice eine *Sequenz von Werten* anstatt eines einzigen Werts erhält

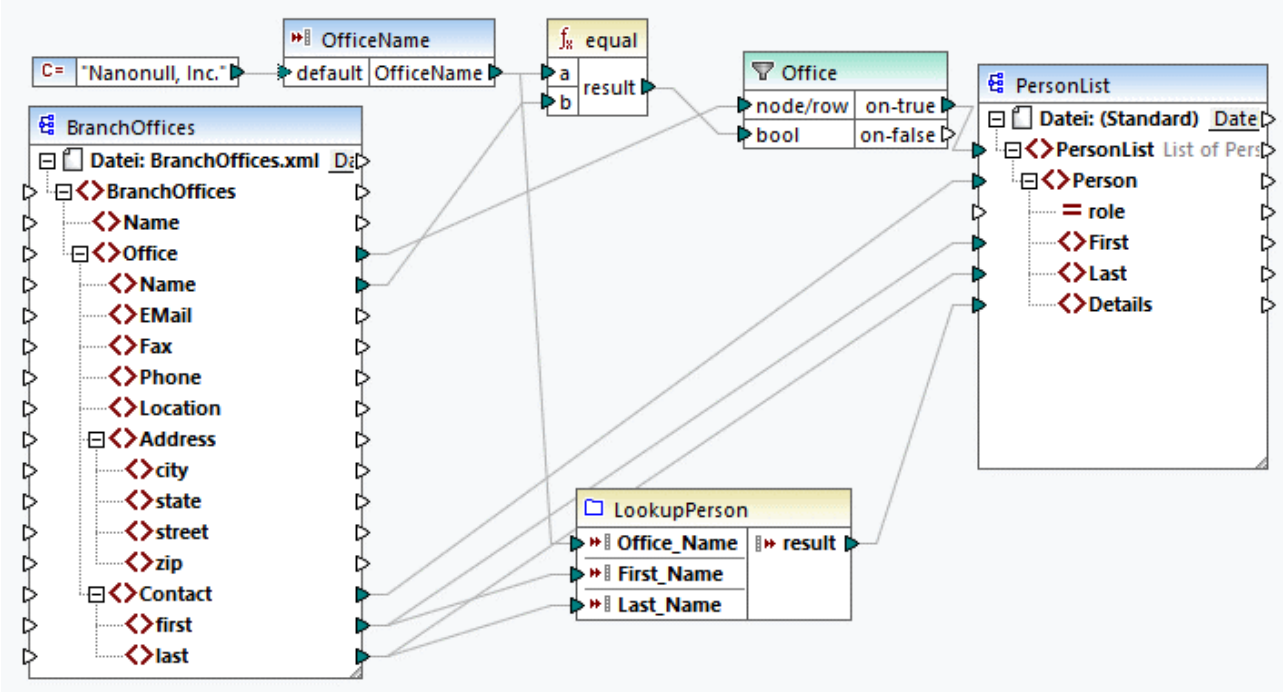
Um dieses Problem zu beheben, verbinden Sie den **Request**-Input des Webservice mit dem Ergebnis (result) der **sum**-Funktion. Das Ergebnis der Funktion ist ein einziger Wert, sodass auch der Webservice nur einmal aufgerufen wird:



Normalerweise erzeugen Aggregatsfunktionen wie **sum**, **count**, einen einzigen Wert. Wenn es jedoch eine übergeordnete Verbindung gibt, die dies zulässt, können sie auch, wie im Beispiel: Ändern des Parent-Kontexts 83 näher beschrieben, eine Sequenz von Werten erzeugen.

5.7.3.5 Look-up-Implementierung

In diesem Kapitel wird erläutert, wie Sie Mitarbeiterdaten abrufen (Look-up) und auf geeignete Weise darstellen können. Um die Look-up-Implementierung auszuprobieren, benötigen Sie das folgende Mapping: **MapForceExamples\PersonListByBranchOffice.mfd**.



Aufgabenstellung
Wir möchten Folgendes erreichen:

- Suchen bestimmter Daten über die einzelnen Mitarbeiter (Telnr., E-Mail-Adresse, Position) mittels Look-up in einer separaten XML-Datei.
- Darstellung dieser Daten in Form einer kommagetrennten Liste und Mappen dieser Liste auf das Element `Details` der XML-Zieldatei.
- Extraktion von Informationen über Mitarbeiter nur aus der Zweigniederlassung namens Nanonull, Inc.

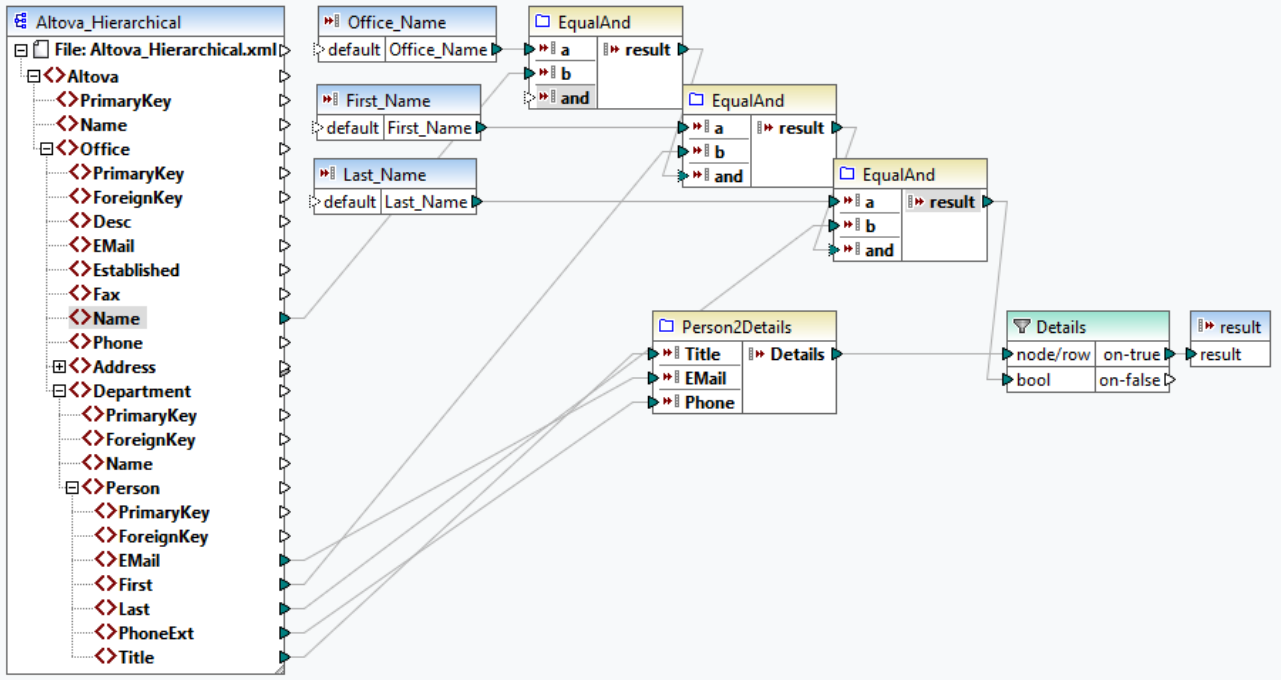
Wir haben unser Mapping zu diesem Zweck folgendermaßen erstellt:

- Mit Hilfe des Filters `Office` werden nur die Mitarbeiter aus Nanonull, Inc. herausgefiltert.
- Um mittels Look-up Informationen aus einer anderen XML-Datei abzurufen, wird im Mapping die benutzerdefinierte Funktion `LookupPerson` aufgerufen. Die Implementierung dieser benutzerdefinierten Funktion wird im folgenden Unterabschnitt beschrieben.
- Zur Verarbeitung der Mitarbeiterdaten ruft die Funktion `LookupPerson` intern weitere Funktionen auf, die Informationen über die einzelnen Mitarbeiter abrufen und miteinander verketten. All diese Operationen befinden sich im Mapping dieser Funktion und sind im Hauptmapping nicht sichtbar. Mit Hilfe der Funktion `LookupPerson` werden anschließend die Mitarbeiterdaten auf das Datenelement `Details` in `PersonList` gemappt.

LookupPerson: Interne Struktur

Die Look-up-Funktionalität wird über die Funktion `LookupPerson`, deren Definition Sie unten sehen, bereitgestellt. Um die interne Implementierung der benutzerdefinierten Funktion zu sehen, doppelklicken Sie im Hauptmapping auf deren Titelleiste.

 © 2019-2025 Altova GmbH

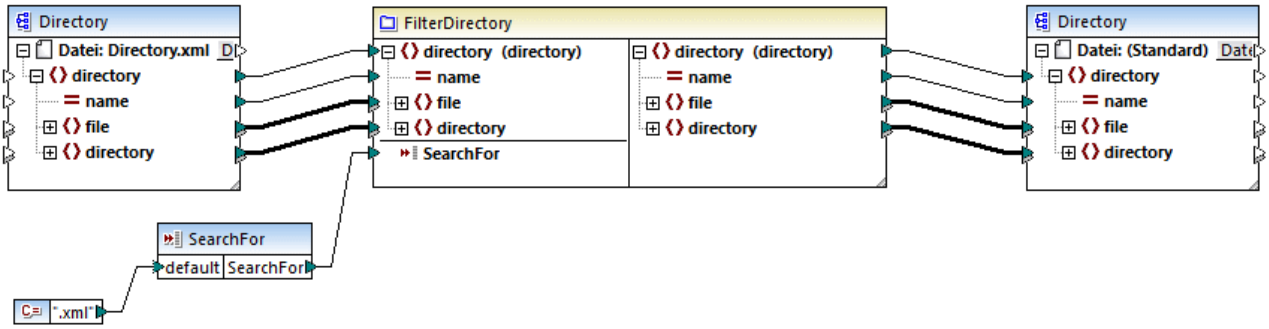


Die benutzerdefinierte Funktion ist folgendermaßen definiert:

- Die Daten werden aus der XML-Datei **`Altova_Hierarchical.xml`** abgerufen: (i) der Name des Büros und die Vor- und Nachnamen der Mitarbeiter, anhand derer Mitarbeiter nur aus Nanonull, Inc. ausgewählt werden und (ii) die E-Mail, der Titel und die Durchwahl, die zu einem String verkettet werden. Unten finden Sie eine Beschreibung der Definition der Funktionen `EqualAnd` und Person2`Detail`.
- Außerdem hat die benutzerdefinierte Funktion drei Input-Parameter, die die Look-up-Werte `Office_Name`, `First_Name` und `Last_Name` bereitstellen. Der Wert des Parameters `Office_Name` wird aus dem `OfficeName`-Input aus dem Hauptmapping abgerufen und die Werte von `First_Name` und `Last_Name` stammen aus der Komponente `BranchOffices` aus dem Hauptmapping.
- Der Wert der Funktion `EqaulAnd` (`true` oder `false`) wird jedes Mal, wenn die Daten eines neuen Mitarbeiters (title, email, phone) verarbeitet werden, an den `Details`-Filter übergeben. Wenn der `Details`-Filter den Wert `true` erhält, ist die Look-up-Operation erfolgreich und die Mitarbeiterdaten werden abgerufen und an das Hauptmapping übergeben. Andernfalls wird das nächste Datenelement im Kontext überprüft usw., bis die Schleife fertig verarbeitet ist.

EqualAnd-Implementierung

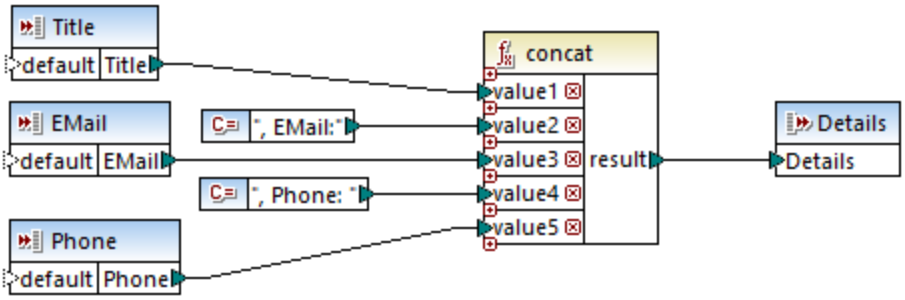
Die `EqualAnd`-Funktion (*siehe unten*) ist eine separate innerhalb der `LookupPerson`-Funktion definierte benutzerdefinierte Funktion. Die interne Struktur der benutzerdefinierten Funktion `EqualAnd` sehen Sie bei Doppelklick auf die Überschrift der Funktion.



Die benutzerdefinierte Funktion `EqualAnd` überprüft zuerst, ob `a` gleich `b` ist; wenn das Ergebnis `true` ist, wird es als erster Parameter der **`logical-and`**-Funktion übergeben. Wenn beide Werte in der **`logical-and`**-Funktion "true" sind, ist das Ergebnis ebenfalls "true" und wird an die nächste `EqualAnd`-Funktion übergeben. Das Ergebnis der dritten `EqaulAnd`-Funktion (*siehe LookupPerson Funktion oben*) wird an den `Details`-Filter übergeben.

Person2Detail-Implementierung

Eine weitere in die benutzerdefinierte Funktion `LookupPerson` verschachtelte Funktion ist die Funktion `Person2Details`. Die benutzerdefinierte Funktion `Person2Details` (*siehe unten*) verkettet drei Werte (aus **`Altova_Hierarchical.xml`**) und zwei Textkonstanten miteinander.



Ausgabe

Wenn Sie auf das Fenster **Ausgabe** klicken, werden in MapForce nur die Vor- und Nachnamen sowie die Daten der Mitarbeiter aus Nanonull, Inc angezeigt (*siehe Codefragment unten*).

```
<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>
  ...
```

 © 2019-2025 Altova GmbH

```
</PersonList>
```

5.7.4 Spezielle benutzerdefinierte Funktionen

In diesem Abschnitt wird erläutert, wie Sie benutzerdefinierte XSLT-239 Funktionen importieren.

5.7.4.1 Importieren benutzerdefinierter XSLT-Funktionen

In MapForce haben Sie die Möglichkeit, die XSLT 1.0-, XSLT 2.0-, und XSLT 3.0-Funktionsbibliothek durch Ihre eigenen benutzerdefinierten Funktionen zu ergänzen, vorausgesetzt, das Ergebnis Ihrer benutzerdefinierten Funktionen sind simpleTypes.

> Es werden nur benutzerdefinierte Funktionen unterstützt, deren Ergebnis einfache Datentypen (z.B. Strings) sind.

So importieren Sie Funktionen aus einer XSLT-Datei:

1. Klicken Sie im unteren Bereich des Fensters Bibliotheken 20 auf die Schaltfläche **Bibliotheken hinzufügen/entfernen**. Daraufhin wird das Fenster **Bibliotheken verwalten** geöffnet (*siehe Abbildung unten*).

2. Um Funktionen als *lokale* Bibliothek (nur im Geltungsbereich des aktuellen Dokuments) zu importieren, klicken Sie auf die Schaltfläche **Hinzufügen** unterhalb des aktuellen Mapping-Namens. Um Funktionen als *globale* Bibliothek (auf Programmebene zu importieren, klicken Sie auf die Schaltfläche **Hinzufügen** neben **Globale Bibliotheksimporte**. Wenn Sie eine Bibliothek *lokal* importieren, können Sie den Pfad zur Bibliotheksdatei als relativ zur Mapping-Datei definieren. Bei gobal importierten Bibliotheken ist der Pfad zur importierten Bibliothek immer absolut.

3. Navigieren Sie zur .xsl-Datei, die die Funktionen enthält und klicken Sie auf **Öffnen**. Daraufhin wird ein Meldungsfeld, in dem Sie informiert werden, dass eine neue Bibliothek hinzugefügt wurde, angezeigt.

Importierte XSLT-Dateien werden im Fenster "Bibliotheken" als Bibliotheken angezeigt. Unterhalb des Bibliotheksnamens werden alle Named Templates als Funktionen angezeigt. Wenn Sie die importierte Bibliothek nicht sehen, stellen Sie sicher, dass Sie XSLT als Transformationssprache 16 ausgewählt haben. Siehe Verwalten von Funktionsbibliotheken 215.

Beachten Sie dazu Folgendes:

- Funktionen müssen als Named Templates deklariert werden, die der XSLT-Spezifikation in der XSLT-Datei entsprechen, damit die Funktionen in MapForce importiert werden können. Sie können auch Funktionen, die in einem XSLT 2.0-Dokument in der Form **`<xsl:function name="MyFunction">`** vorkommen, importieren. Wenn die importierte XSLT-Datei andere XSLT-Dateien importiert oder beinhaltet, so werden auch diese XSLT-Dateien und Funktionen importiert.
- Die mapbaren Input-Konnektoren von importierten benutzerdefinierten Funktionen hängen von der Anzahl der Parameter ab, die im Template Call verwendet werden; auch optionale Parameter werden unterstützt.
- Namespaces werden unterstützt.
- Wenn Sie Änderungen an XSLT-Dateien, die bereits in MapForce importiert wurden, vornehmen, so werden die Änderungen automatisch erkannt und MapForce fragt Sie, ob die Dateien neu geladen werden sollen.
- Stellen Sie beim Erstellen von Named Templates sicher, dass die im Template verwendeten XPath-Anweisungen an den/die richtigen Namespace(s) gebunden sind. Um die Namespace Bindings des Mappings zu sehen, zeigen Sie eine Vorschau des generierten XSLT-Codes 65 an.

Datentypen in XPath 2.0

Wenn Ihr XML-Dokument ein XML-Schema referenziert und gemäß diesem Schema gültig ist, müssen Sie Datentypen, die nicht durch eine Operation implizit in den benötigten Datentyp konvertiert werden, explizit konstruieren oder konvertieren.

Im XPath 2.0 Datenmodell, das vom Altova XSLT 2.0-Prozessor verwendet wird, wird allen **atomized** Nodewerten aus dem XML-Dokument der Datentyp `xs:untypedAtomic` zugewiesen. Der Typ `xs:untypedAtomic` funktioniert gut mit impliziten Typkonvertierungen.

Beispiel:

- Der Ausdruck `xs:untypedAtomic("1") + 1` hat als Ergebnis den Wert 2, da der Wert `xdt:untypedAtomic` *implizit* vom Plus-Zeichen in `xs:double` geändert wird.
- Arithmetische Operatoren konvertieren Operanden implizit in `xs:double`.
- Operatoren zum Vergleich von Werten konvertieren Operanden vor dem Vergleich in `xs:string`.

Siehe auch:

Beispiel: Hinzufügen benutzerdefinierter XSLT-Funktionen 240
Beispiel: Summieren von Node-Werten 243
Implementierung des XSLT 1.0-Prozessors 527
Implementierung des XSLT 2.0-Prozessors 528

5.7.4.1.1 Beispiel: Hinzufügen benutzerdefinierter XSLT-Funktionen

In diesem Beispiel wird gezeigt, wie Sie benutzerdefinierte XSLT 1.0-Funktionen in MapForce importieren. Die für dieses Beispiel benötigten Dateien befinden sich im Verzeichnis **`C:\Users\<username>\Documents\Altova\MapForce2025\MapForceExamples`**.

 © 2019-2025 Altova GmbH

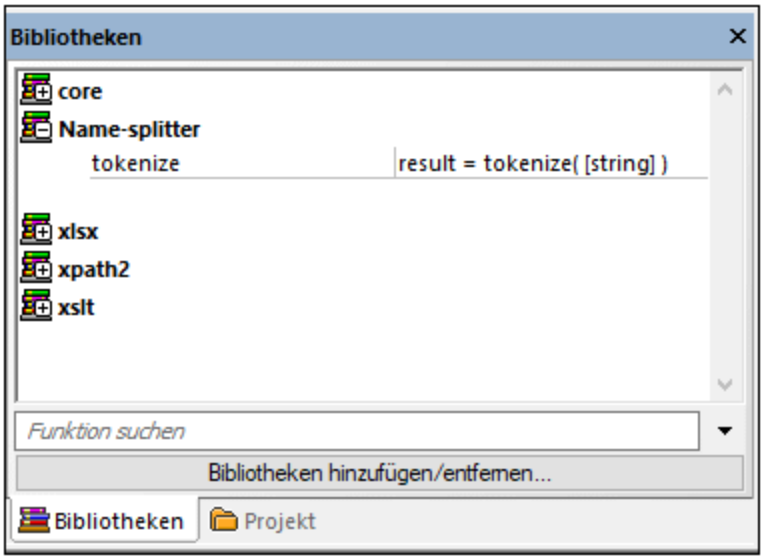
- **Name-splitter.xslt**. In dieser XSLT-Datei ist ein Named Template namens **tokenize** mit einem einzigen Parameter "string" definiert. Das Template funktioniert über einen Input-String und trennt Großbuchstaben durch ein Leerzeichen für jede Instanz.
- **Name-splitter.xml** (die zu verarbeitende XML-Quellinstanzdatei)
- **Customers.xsd** (das XML-Quellschema)
- **CompletePO.xsd** (das XML-Zielschema)

So fügen Sie eine benutzerdefinierte XSLT-Funktion hinzu:

1. Klicken Sie im unteren Bereich des Fensters Bibliotheken 20 auf die Schaltfläche **Bibliotheken hinzufügen/entfernen**. Daraufhin wird das Fenster **Bibliotheken verwalten** geöffnet (*siehe Abbildung unten*).

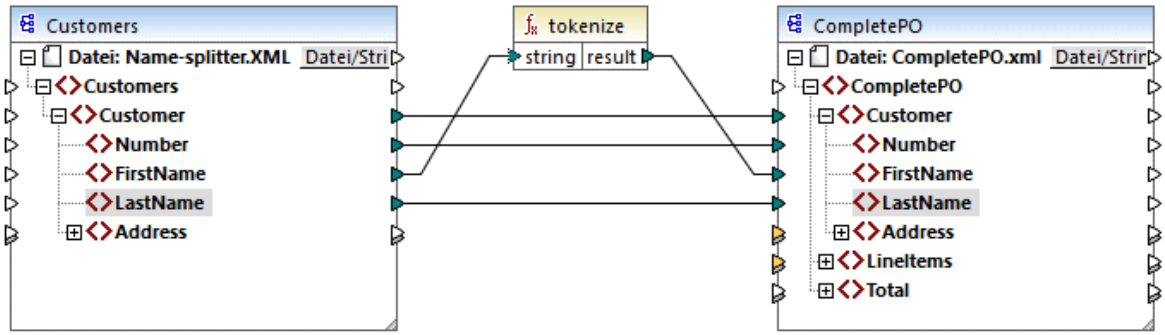
2. Um Funktionen als *lokale* Bibliothek (nur im Geltungsbereich des aktuellen Dokuments) zu importieren, klicken Sie auf die Schaltfläche **Hinzufügen** unterhalb des aktuellen Mapping-Namens. Um Funktionen als *globale* Bibliothek (auf Programmebene zu importieren, klicken Sie auf die Schaltfläche **Hinzufügen** neben **Globale Bibliotheksimporte**. Wenn Sie eine Bibliothek *lokal* importieren, können Sie den Pfad zur Bibliotheksdatei als relativ zur Mapping-Datei definieren. Bei gobal importierten Bibliotheken ist der Pfad zur importierten Bibliothek immer absolut.

3. Navigieren Sie zur .xsl- oder .xslt-Datei, die das Named Template, das Sie als Funktion hinzufügen möchten, enthält - in diesem Fall zu **Name-splitter.xslt** und klicken Sie auf **Öffnen**. Daraufhin erscheint ein Meldungsfeld, in dem Sie darüber informiert werden, dass eine neue Bibliothek hinzugefügt wurde und der Name der XSLT-Datei sowie die als Named Templates definierten Funktionen (in diesem Beispiel **Name-splitter** mit der Funktion `tokenize`) werden im Fenster "Bibliotheken" angezeigt.



So verwenden Sie die XSLT-Funktion in Ihrem Mapping:

1. Ziehen Sie die **`tokenize`**-Funktion in das Mapping-Fenster und mappen Sie die Datenelemente wie folgt.



2. Klicken Sie auf das Fenster **XSLT**, um den erzeugten XSLT-Code zu sehen.

© 2019-2025 Altova GmbH

```
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
```

Anmerkung: Sobald eine Named Template in einem Mapping verwendet wird, wird die XSLT-Datei, die den Namen des Named Template enthält, im generierten XSLT-Code **inkludiert** (**xsl:include href...**) und mit dem Befehl **xsl:call-template aufgerufen**.

3. Klicken Sie auf das Fenster **Ausgabe,** um das Ergebnis des Mappings zu sehen.

So entfernen Sie benutzerdefinierte XSLT-Funktionen aus MapForce:

1. Klicken Sie im unteren Bereich des Fensters "Bibliotheken" auf die Schaltfläche **Bibliotheken hinzufügen/entfernen**. Daraufhin wird das Fenster "Bibliotheken verwalten" geöffnet.
2. Klicken Sie neben der zu löschenden XSLT-Bibliothek auf **Bibliothek löschen** ✕.

5.7.4.1.2 Beispiel: Summieren von Node-Werten

In diesem Beispiel wird gezeigt, wie Sie mehrere Nodes eines XML-Dokuments verarbeiten und die Ergebnisse als einen einzigen Wert auf ein XML-Zieldokument mappen können. Ziel des Mappings ist es, den Preis aller Produkte in einer XML-Quelldatei zu berechnen und diesen als einen einzigen Wert in eine XML-Ausgabedatei zu schreiben. Die in diesem Beispiel verwendeten Dateien stehen im Ordner **<Dokumente>\Altova\MapForce2025\MapForceExamples** zur Verfügung.

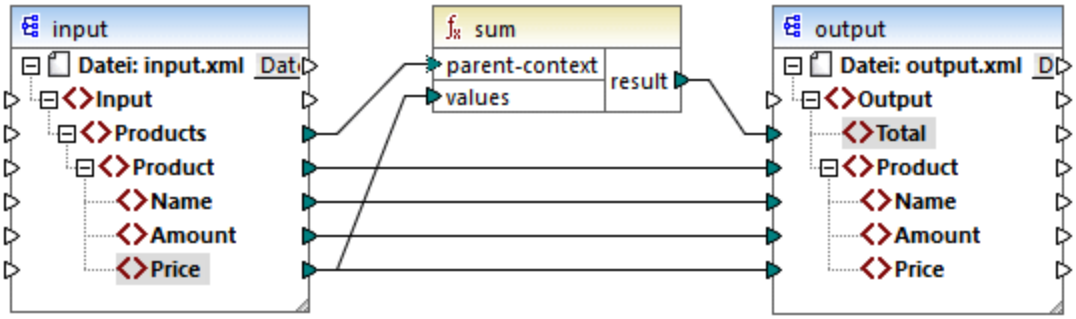
- **Summing-nodes.mfd** -die Mapping-Datei
- **input.xml** - die XML-Quelldatei
- **input.xsd** - das XML-Quellschema
- **output.xsd** - die XML-Zieldatei
- **Summing-nodes.xslt** - ein benutzerdefiniertes XSLT-Stylesheet, das eine benannte Vorlage zum Summieren der einzelnen Nodes enthält.

Es gibt zwei verschiedene Methoden, mit denen Sie das Ziel des Mappings erreichen können:

- Verwendung der sum[260]-Funktion. Diese vordefinierte MapForce-Funktion steht im Fenster "Bibliotheken" zur Verfügung.
- durch Import eines benutzerdefinierten XSLT-Stylesheet in MapForce.

Lösungsvariante 1: Verwendung der Aggregatfunktion "sum"

Um die Funktion **sum** im Mapping zu verwenden, ziehen Sie sie aus dem Fenster Bibliotheken in das Mapping. Beachten Sie, dass es von der ausgewählten XSLT-Version abhängt (XSLT1 oder XSLT2), welche Funktionen im Fenster Bibliotheken zur Verfügung stehen. Erstellen Sie nun die Mapping-Verbindungen wie unten gezeigt.



Nähere Informationen zu Aggregatfunktionen der core-Bibliothek finden Sie unter core | aggregate functions (Aggregatfunktionen)[253].

Lösungsvariante 2: Verwendung eines benutzerdefinierten XSLT-Stylesheet

Wie oben erwähnt, ist das Ziel des Beispiels, die `Price`-Felder von Produkten in der XML-Quelldatei, in diesem Fall der Produkte A und B, zu summieren.

```
<?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>
```

 © 2019-2025 Altova GmbH

Im Codefragment unten sehen Sie ein benutzerdefiniertes XSLT-Stylesheet, in dem die benannte Vorlage (named template) "Total" und ein einziger Parameter `string` verwendet werden. Die Vorlage verarbeitet die XML-Input-Datei und summiert alle durch den XPath-Ausdruck `/Product/Price` bereitgestellten Werte.

```
<?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>
```

Anmerkung: Um die Nodes in XSLT 2.0 zu summieren, ändern Sie die Stylesheet-Deklaration in `version="2.0"`.

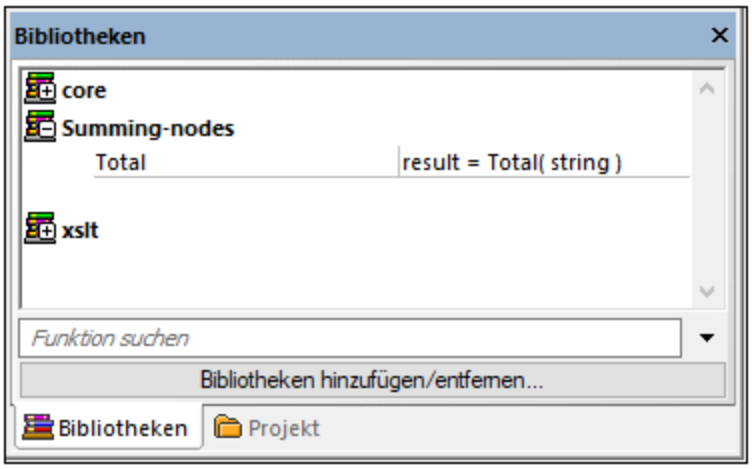
Bevor Sie das XSLT-Stylesheet in MapForce importieren, wählen Sie als Transformationssprache 16 XSLT 1.0. aus. Sie können die benutzerdefinierte Funktion nun folgendermaßen importieren:

1. Klicken Sie im unteren Bereich des Fensters Bibliotheken 20 auf die Schaltfläche **Bibliotheken hinzufügen/entfernen**. Daraufhin wird das Fenster **Bibliotheken verwalten** geöffnet (*siehe Abbildung unten*).

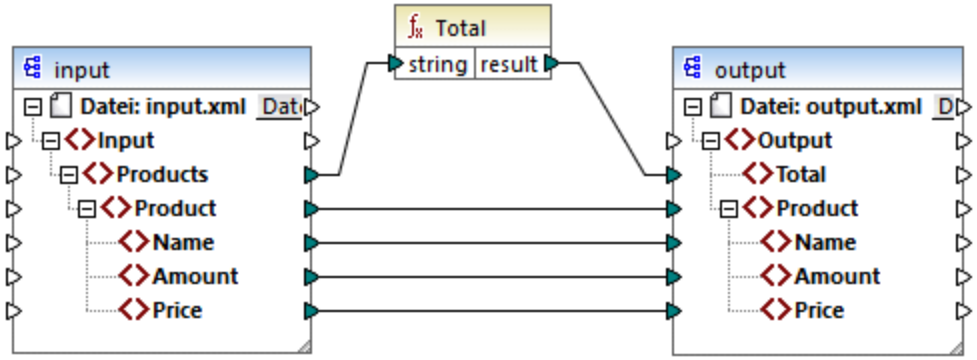
2. Um Funktionen als *lokale* Bibliothek (nur im Geltungsbereich des aktuellen Dokuments) zu importieren, klicken Sie auf die Schaltfläche **Hinzufügen** unterhalb des aktuellen Mapping-Namens. Um Funktionen als *globale* Bibliothek (auf Programmebene zu importieren, klicken Sie auf die Schaltfläche **Hinzufügen** neben **Globale Bibliotheksimporte**. Wenn Sie eine Bibliothek *lokal* importieren, können Sie den Pfad zur Bibliotheksdatei als relativ zur Mapping-Datei definieren. Bei gobal importierten Bibliotheken ist der Pfad zur importierten Bibliothek immer absolut.

3. Navigieren Sie zur Datei **<Dokumente>\Altova\MapForce2025\MapForceExamples\Summing-nodes.xslt** und klicken Sie auf **Öffnen**. Daraufhin erscheint ein Meldungsfeld, in dem Sie darüber

informiert werden, dass eine neue Bibliothek hinzugefügt wurde, und die neue Bibliothek wird im Fenster "Bibliotheken" angezeigt.



4. Ziehen Sie die Funktion **`Total`** aus der Bibliothek ins Mapping und erstellen Sie die Mapping-Verbindungen, wie unten gezeigt.



Um eine Vorschau auf das Mapping-Ergebnis zu sehen, klicken Sie auf das Fenster **Ausgabe**. Im Feld `Total` wird nun die Summe der zwei `Price`-Felder angezeigt.

```
<?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>
```

 © 2019-2025 Altova GmbH

5.7.5 Regular Expressions

Sie können Regular Expressions ("regex") bei der Erstellung eines MapForce Mappings in folgenden Situationen verwenden:

- im **pattern**-Parameter der Funktion **`tokenize-regexp`** 336.

Die Syntax und Semantik von Regular Expressions für XSLT und XQuery entspricht den in Appendix F von "XML Schema Part 2: Datatypes Second Edition" definierten Regeln.

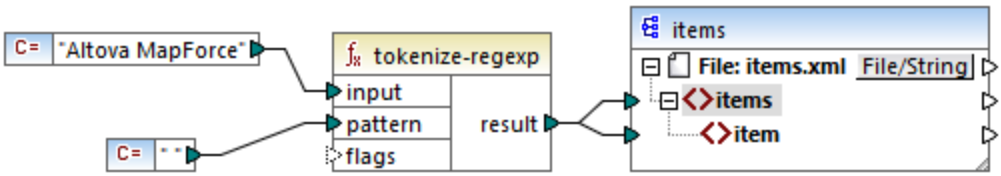
Anmerkung: Die komplexen Funktionalitäten der Regular Expression-Syntax können bei der Generierung von C++-, C#- oder Java-Code etwas unterschiedlich sein. Nähere Informationen dazu finden Sie in der regex-Dokumentation zu den einzelnen Sprachen.

Terminologie

Die grundlegende Terminologie von Regular Expressions wird hier anhand des Beispiels **`tokenize-regexp`** erläutert. Diese Funktion teilt Text mit Hilfe von Regular Expressions in eine Sequenz von Strings auf. Zu diesem Zweck erhält die Funktion die folgenden Input-Parameter:

input	Der Input-String, der von der Funktion verarbeitet werden soll. Die Regular Expression wird an diesem String angewendet.
pattern	Das eigentliche Muster der Regular Expression, die angewendet werden soll.
flags	Dies ist ein optionaler Parameter, der zusätzliche Optionen (Flags) definiert, die festlegen, wie die Regular Expression interpretiert werden soll, siehe "Flags" weiter unten.

Der Input-String im Mapping unten ist "Altova MapForce". Der Parameter **pattern** ist ein Leerzeichen. Es werden keine Regular Expression Flags verwendet.



Der Text wird dadurch überall dort, wo ein Leerzeichen steht, aufgeteilt, daher ist die Ausgabe des Mappings:

```
<items>
   <item>Altova</item>
   <item>MapForce</item>
</items>
```

Beachten Sie, dass die Funktion **`tokenize-regexp`** die übereinstimmenden Zeichen aus dem Ergebnis ausschließt. D.h. das Leerzeichen wird in diesem Beispiel aus der Ausgabe entfernt.

Das obige Beispiel ist sehr einfach gehalten. Dasselbe Ergebnis kann auch ohne Regular Expressions, mit Hilfe der Funktion tokenize 335 erzielt werden. In einem realistischeren Szenario würde der Parameter **pattern**

eine komplexere Regular Expression enthalten. Die Regular Expression kann aus beliebigen der folgenden Bestandteile bestehen:

- Literale
- Zeichenklassen
- Zeichenbereiche
- Negierte Klassen
- Metazeichen
- Quantifizierer

Literale

Literale dienen zum Finden von Zeichen in exakt dieser Schreibweise. Wenn der Input-String z.B. **abracadabra** lautet und **pattern** das Literal **br** ist, so ist die Ausgabe die folgende:

```
<items>
   <item>a</item>
   <item>acada</item>
   <item>a</item>
</items>
```

Der Grund dafür ist, dass das Literal **br** im Input-String **abracadabra** zwei Übereinstimmungen hat. Nachdem die übereinstimmenden Zeichen aus der Ausgabe entfernt wurden, wird die oben gezeigte Sequenz von drei Strings erzeugt.

Zeichenklassen

Wenn Sie eine Gruppe von Zeichen in eckige Klammern (**[** und **]**) setzen, wird dadurch eine Zeichenklasse erstellt. Es wird immer nur jeweils ein einziges der Zeichen innerhalb der Zeichenklasse gefunden, z.B:

- Mit dem Muster **[aeiou]** werden alle klein geschriebenen Vokale gefunden.
- Mit dem Muster **[mj]ust** werden "must" und "just" gefunden.

Anmerkung: Die Groß- und Kleinschreibung spielt beim Muster eine Rolle, d.h. ein kleingeschriebenes "a" stimmt nicht mit dem Großbuchstaben "A" überein. Damit Groß- und Kleinschreibung ignoriert werden, verwenden Sie das Flag **i**, siehe unten.

Zeichenbereiche

Mit **[a-z]** erstellen Sie einen Bereich zwischen den beiden Zeichen. Es wird immer nur jeweils eines der Zeichen gefunden. So wird etwa mit dem Muster **[a-z]** jeder Kleinbuchstabe zwischen "a" und "z" gefunden.

Negierte Klassen

Mit dem Zikumflexzeichen (**^**) als erstem Zeichen nach der öffnenden Klammer wird die Zeichenklasse negiert. Mit dem Muster **[^a-z]** wird z.B. jedes Zeichen außerhalb der Zeichenklasse, einschließlich Zeilenschaltungen (newline-Zeichen) gefunden.

 © 2019-2025 Altova GmbH

Übereinstimmung mit jedem beliebigen Zeichen.

Mit Hilfe des Punkt-Metazeichens (`.`) wird jedes beliebige Einzelzeichen mit Ausnahme einer Zeilenschaltung (newline) gefunden. So wird z.B. mit dem Muster `.` jedes einzelne Zeichen gefunden.

Quantifizierer

Quantifizierer definieren in einer Regular Expression, wie oft das vorangestellte Zeichen oder der vorangestellte Unterausdruck vorkommen darf, damit eine Übereinstimmung gefunden wird.

?	Steht für null oder eine Instanz des direkt davor stehenden Elements. Das Muster `mo?` steht z.B. für "m" und "mo".
+	Steht für eine oder mehr Instanzen des direkt davor stehenden Elements. Das Muster `mo+` steht z.B. für "mo", "moo", "mooo", usw.
*****	Steht für null oder mehr Instanzen direkt davor stehenden Elements.
{min,max}	Steht für beliebig viele Instanzen zwischen *min* und *max*. So steht etwa das Muster `mo{1,3}` für "mo", "moo" und "mooo".

Klammern

Mit Hilfe von Klammern `(` und `)` werden Teile eines regex-Ausdrucks gruppiert. Sie können damit Quantifizierer auf einen Unterausdruck (anstatt auf nur ein Zeichen) anwenden. Oder Sie verwenden diese mit einer Alternative (siehe unten).

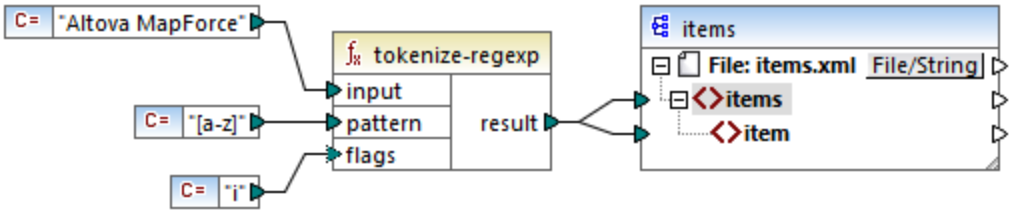
Alternative

Der senkrechte Strich (Pipe-Zeichen) `|` bedeutet "oder". Er findet einen beliebigen von mehreren durch `|` getrennte Unterausdrücke. So findet etwa das Muster `(horse|make) sense` sowohl "horse sense" und "make sense".

Flags

Flags sind optionale Parameter, die definieren, wie die Regular Expression interpretiert werden soll. Jedes Flag entspricht einem Buchstaben. Die Buchstaben können in jeder Reihenfolge vorkommen und auch wiederholt werden.

s	Falls dieses Flag vorhanden ist, wird der Suchvorgang im "dot-all"-Modus durchgeführt. Wenn der Input-String "hello" und "world" in zwei *verschiedenen* Zeilen vorkommt, findet die Regular Expression `hello*world` nur dann eine Übereinstimmung, wenn das Flag **s** gesetzt wurde.
m	Wenn dieses Flag vorhanden ist, wird der Suchvorgang im mehrzeiligen Modus durchgeführt. Im mehrzeiligen Modus steht das Zirkumflexzeichen `^` für den Beginn jeder Zeile, d.h. den Beginn des gesamten Strings und das erste Zeichen nach einem "newline"-Zeichen.

	Das Dollarzeichen **$** steht für das Ende jeder Zeile, d.h. das Ende des gesamten Strings und das Zeichen unmittelbar vor einem "newline" Zeichen. Newline (Neue Zeile) ist das Zeichen **#x0A**.
i	Wenn dieses Flag vorhanden ist, wird die Groß- und Kleinschreibung ignoriert. So würden etwa mit der Regular Expression **[a-z]** plus dem **i** Flag alle Buchstaben von a-z und A-Z gefunden. 
x	Falls dieses Flag vorhanden ist, werden Leerzeichen und andere Whitespace-Zeichen vor dem Suchvorgang aus der Regular Expression entfernt. Whitespace-Zeichen sind **#x09**, **#x0A**, **#x0D** und **#x20**. **Anmerkung:** Whitespace-Zeichen innerhalb einer Zeichenklasse z.B. **[#x20]** werden nicht entfernt.

5.7.6 Referenz Funktionsbibliothek

In diesem Kapitel werden die vordefinierten MapForce-Funktionen, die im Fenster "Bibliotheken" [20] zur Verfügung stehen, nach Bibliothek geordnet, beschrieben. Welche Funktionsbibliotheken im Fenster **Bibliotheken** verfügbar sind, hängt von der ausgewählten Transformationssprache ab. Nähere Informationen über die Liste der verfügbaren Transformationssprachen finden Sie in diesem Kapitel [16].

Die folgenden Unterabschnitte enthalten Informationen über die Kompatibilität von Funktionen mit Transformationssprachen.

core functions (core-Funktionen)

Die nachfolgenden Listen enthalten eine Übersicht über die Kompatibilität von core-Funktionen mit Transformationssprachen.

core | aggregate functions (Aggregatfunktionen)

- **avg**, **max**, **max-string**, **min**, **min-string**: XSLT 2.0, XSLT 3.0, XQuery 1.0, C#, C++, Java, Built-In.
- **count**, **sum**: alle Transformationssprachen.

core | conversion functions (Konvertierungsfunktionen)

- **boolean**, string, **number**: alle Transformationssprachen.
- **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.

 © 2019-2025 Altova GmbH

core | file path functions (Dateipfadfunktionen)
Alle Dateipfadfunktionen sind mit allen Transformationssprachen kompatibel.

core | generator functions (Generierungsfunktionen)
Die Funktion **auto-number** steht für alle Transformationssprachen zur Verfügung.

core | logical functions (logische Funktionen)
Die logischen Funktionen sind mit allen Transformationssprachen kompatibel.

core | math functions (mathematische Funktionen)

- **add**, **ceiling**, **divide**, **floor**, **modulus**, **multiply**, **round**, **subtract**: alle Transformationssprachen.
- **round-precision**: C#, C++, Java, Built-In.

core | node functions (Node-Funktionen)

- **is-xsi-nil**, **local-name**, **static-node-annotation**, **static-node-name**: alle Transformationssprachen.
- **node-name**, **set-xsi-nil**, **substitute-missing-with-xsi-nil**: XSLT 2.0, XSLT 3.0, XQuery 1.0, C#, C++, Java, Built-In.

core | QName functions (QName-Funktionen)
Die `QName`-Funktionen sind mit allen Transformationssprachen mit Ausnahme von XSLT 1.0 kompatibel.

core | sequence functions (Sequenzfunktionen)

- **exists**, **not-exists**, **position**, **substitute-missing**: alle Transformationssprachen.
- **distinct-values**, **first-items**, **generate-sequence**, **item-at**, **items-from-till**, **last-items**, **replicate-item**, **replicate-sequence**, **set-empty**, **skip-first-items**: XSLT 2.0, XSLT 3.0, XQuery 1.0, C#, C++, Java, Built-In.
- **group-adjacent**, **group-by**, **group-ending-with**, **group-into-blocks**, **group-starting-with**: XSLT 2.0, XSLT 3.0, C#, C++, Java, Built-In.

core | string functions (String-Funktionen)

- **concat**, **contains**, **normalize-space**, **starts-with**, **string-length**, **substring**, **substring-after**, **substring-before**, **translate**: alle Transformationssprachen.
- **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.

bson functions (BSON-Funktionen) (nur MapForce Enterprise Edition)

Alle BSON-Funktionen sind nur mit Built-in kompatibel.

db functions (DB-Funktionen) (MapForce Professional und Enterprise Edition)

Die `db`-Funktionen sind mit C#, C++, Java, Built-In kompatibel.

edifact functions (EDIFACT-Funktionen) (nur MapForce Enterprise Edition)

Die `edifact`-Funktionen sind mit C#, C++, Java, Built-In kompatibel.

lang functions (MapForce Professional und Enterprise Edition)

Die nachfolgenden Listen enthalten eine Übersicht über die Kompatibilität von `lang`-Funktionen mit Transformationssprachen.

lang | datetime functions (Datums- und Uhrzeitfunktionen)

Die `lang | datetime`-Funktionen sind mit C#, C++, Java, Built-In kompatibel.

lang | file functions (Dateifunktionen)

Die Funktionen **`read-binary-file`** und **`write-binary-file`** sind nur mit Built-in kompatibel.

lang | generator functions (Generierungsfunktionen)

Die **`create-guid`**-Funktionen stehen für C#, C++, Java, Built-In zur Verfügung.

lang | logical functions (logische Funktionen)

Die `lang | logical`-Funktionen stehen für C#, C++, Java, Built-In zur Verfügung.

lang | math functions (mathematische Funktionen)

Die `lang | math`-Funktionen stehen für C#, C++, Java, Built-In zur Verfügung.

lang | QName functions (QName-Funktionen)

Die `lang | QName`-Funktionen stehen für C#, C++, Java, Built-In zur Verfügung.

lang | string functions (String-Funktionen)

- **`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.

mime functions (MIME-Funktionen) (nur MapForce Enterprise Edition)

Die `mime`-Funktionen stehen nur für Built-In zur Verfügung.

xbrl functions (XBRL-Funktionen) (nur MapForce Enterprise Edition)

Die `xbrl`-Funktionen sind mit C#, C++, Java, Built-In kompatibel.

xslx functions (XSLT-Funktionen) (nur MapForce Enterprise Edition)

Die `xlsx`-Funktionen sind mit XSLT 2.0, XSLT 3.0, C#, Java und Built-In kompatibel.

xpath2 functions (XPath2-Funktionen)

Alle `xpath2`-Funktionen sind mit XSLT 2.0, XSLT 3.0 und XQuery 1.0 kompatibel.

xpath3 functions (XPath3-Funktionen)

Alle `xpath3`-Funktionen sind nur mit XSLT 3.0 kompatibel.

© 2019-2025 Altova GmbH

xslt10 functions (XSLT10-Funktionen)

Die nachfolgenden Listen enthalten eine Übersicht über die Kompatibilität von `xslt10`-Funktionen mit Transformationssprachen.

xslt10 | xpath functions (XPath-Funktionen)

- **`local-name`**, **`name`**, **`namespace-uri`**: XSLT 1.0, XSLT 2.0 und XSLT 3.0.
- **`lang`**, **`last`**, **`position`**: XSLT 1.0.

xslt10 | xslt functions (XSLT-Funktionen)

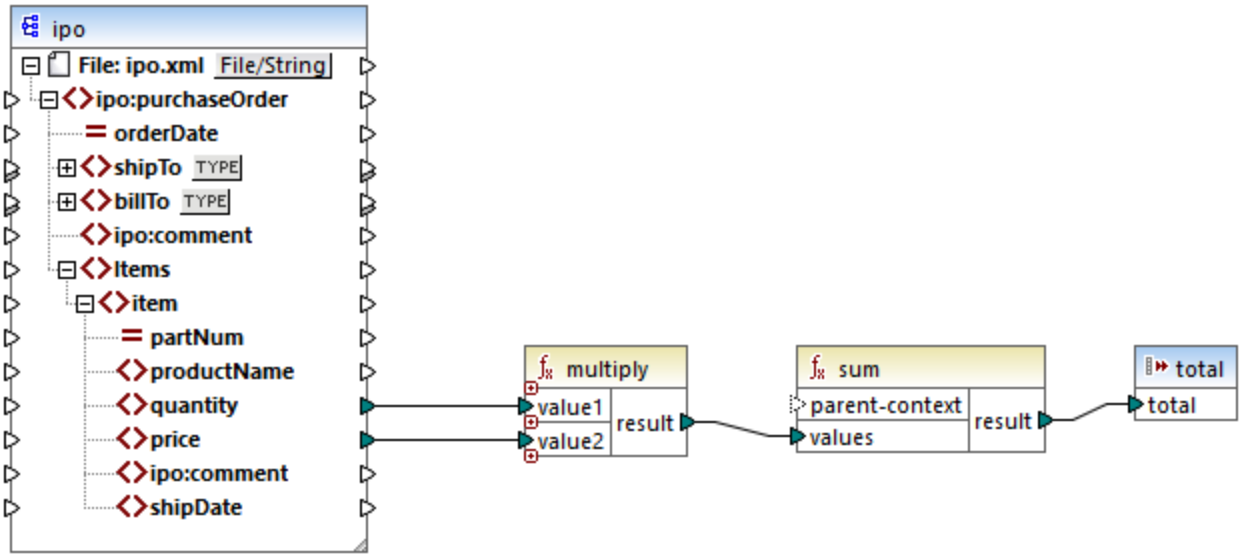
- **`generate-id`**, **`system-property`**: XSLT 1.0, XSLT 2.0 und XSLT 3.0.
- **`current`**, **`document`**, **`element-available`**, **`function-available`**, **`unparsed-entity-uri`**: XSLT 1.0.

5.7.6.1 core | aggregate functions (Aggregatfunktionen)

"Aggregieren" bedeutet mehrere Werte desselben Typs zu verarbeiten, um ein einziges Resultat wie z.B. eine Summe, eine Anzahl oder einen Durchschnitt zu erhalten. Datenaggregationen können in MapForce mit Hilfe von Aggregatfunktionen wie `avg`, `count`, `max` und anderen durchgeführt werden.

Die folgenden beiden Argumente werden in allen Aggregatfunktionen verwendet:

1. **parent-context**. Dieses Argument ist optional. Sie können damit den Standard-Mapping-Kontext außer Kraft setzen (und dadurch den Geltungsbereich der Funktion oder die Werte, über die die Funktion iterieren muss, ändern). Ein Arbeitsbeispiel dazu finden Sie unter Beispiel: Ändern des Parent-Kontexts 83.
2. **values**. Dieses Argument muss mit einem Quell-Datenelement, das die zu verarbeitenden Werte bereitstellt, verbunden sein. So erhält etwa die **`sum`**-Funktion im unten gezeigten Mapping als Input eine Sequenz numerischer Werte, die aus einer XML-Quelldatei stammen. Für jedes Datenelement in der XML-Quelldatei ermittelt die **`multiply`**-Funktion den Preis (price) des Datenelements mal Anzahl (quantity) und übergibt das Ergebnis an die **`sum`**-Funktion. Die **`sum`**-Funktion aggregiert alle Input-Werte und erzeugt eine Gesamtsumme, die auch den Output des Mappings bildet. Sie finden das Mapping im Ordner **`MapForceExamples`**.

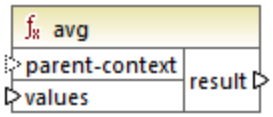


SimpleTotal.mfd

Einige Aggregatfunktionen wie `min`, `max`, `sum` und `avg` funktionieren ausschließlich mit numerischen Werten. Die Input-Daten dieser Funktionen werden für die Verarbeitung in den Datentyp **decimal** konvertiert.

5.7.6.1.1 avg

Gibt den Durchschnittswert aller Werte in der Input-Sequenz zurück. Der Durchschnittswert einer leeren Gruppe ist eine leere Gruppe.



Sprachen

Built-in, C++, C#, Java, XQuery, XSLT 2.0, XSLT 3.0.

Parameter

Argument	Beschreibung
parent-context	Optionales Argument. Liefert den parent context. Siehe auch Beispiel: Ändern des Parent-Kontexts 83. Das Argument `parent-context` ist ein optionales Argument in einigen MapForce Aggregatfunktionen der core-Bibliothek wie z.B. **`min`**, **`max`**, **`avg`**, **`count`**. Der parent-context bestimmt in einer Quellkomponente mit mehreren hierarchischen Sequenzen,

 © 2019-2025 Altova GmbH

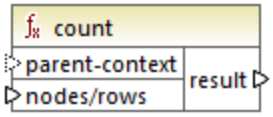
Argument	Beschreibung
	an welcher Node-Gruppe die Funktion ausgeführt werden soll.
values	Dieses Argument muss mit einem Quelldatenelement verbunden sein, aus dem die eigentlichen Daten stammen. Beachten Sie, dass der bereitgestellte Argumentwert numerisch sein muss.

Beispiel

Siehe Beispiel: Gruppieren von Datensätzen nach Schlüssel 305

5.7.6.1.2 count

Gibt die Anzahl der einzelnen Datenelemente zurück, aus denen die Input-Sequenz besteht. Die Anzahl einer leeren Gruppe ist Null.



Sprachen

Built-in, C++, C#, Java, XQuery, XSLT 1.0, XSLT 2.0, XSLT 3.0.

Beachten Sie, dass diese Funktion in XSLT 1.0 eingeschränkte Funktionen hat.

Parameter

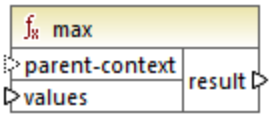
Argument	Beschreibung
parent-context	Optionales Argument. Liefert den parent-context. Siehe auch Beispiel: Ändern des Parent-Kontexts 83. Das Argument `parent-context` ist ein optionales Argument in einigen MapForce Aggregatfunktionen der core-Bibliothek wie z.B. **`min`**, **`max`**, **`avg`**, **`count`**. Der parent-context bestimmt in einer Quellkomponente mit mehreren hierarchischen Sequenzen, an welcher Node-Gruppe die Funktion ausgeführt werden soll.
nodes/rows	Dieses Argument muss mit einem Quelldatenelement verbunden sein, damit die Zählung erfolgen kann.

Beispiel

Siehe Beispiel: Ändern des Parent-Kontexts 83.

5.7.6.1.3 max

Gibt den Maximalwert aller numerischen Werte in der Input-Sequenz zurück. Der Maximalwert einer leeren Gruppe ist eine leere Gruppe.



Sprachen

Built-in, C++, C#, Java, XQuery, XSLT 2.0, XSLT 3.0.

Parameter

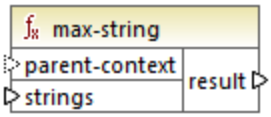
Argument	Beschreibung
parent-context	Optionales Argument. Liefert den parent-context. Siehe auch Beispiel: Ändern des Parent-Kontexts 83. Das Argument `parent-context` ist ein optionales Argument in einigen MapForce Aggregatfunktionen der core-Bibliothek wie z.B. **`min`**, **`max`**, **`avg`**, **`count`**. Der parent-context bestimmt in einer Quellkomponente mit mehreren hierarchischen Sequenzen, an welcher Node-Gruppe die Funktion ausgeführt werden soll.
values	Dieses Argument muss mit einem Quelldatenelement verbunden sein, aus dem die eigentlichen Daten stammen. Beachten Sie, dass der bereitgestellte Argumentwert numerisch sein muss. Um den Maximalwert einer Sequenz von Strings zu ermitteln, verwenden Sie die Funktion max-string 256.

Beispiel

Siehe Beispiel: Gruppieren von Datensätzen nach Schlüssel 305

5.7.6.1.4 max-string

Gibt den Maximalwert aller String-Werte in der Input-Sequenz zurück. So gibt **`max-string("a", "b", "c")`** z.B. **`"c"`** zurück. Die Funktion gibt eine leere Gruppe zurück, wenn das Argument **strings** eine leere Gruppe ist.



 © 2019-2025 Altova GmbH

Sprachen

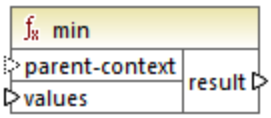
Built-in, C++, C#, Java, XQuery, XSLT 2.0, XSLT 3.0..

Parameter

Argument	Beschreibung
parent-context	Optionales Argument. Liefert den parent-context. Siehe auch Beispiel: Ändern des Parent-Kontexts 83. Das Argument `parent-context` ist ein optionales Argument in einigen MapForce Aggregatfunktionen der core-Bibliothek wie z.B. **`min`**, **`max`**, **`avg`**, **`count`**. Der parent-context bestimmt in einer Quellkomponente mit mehreren hierarchischen Sequenzen, an welcher Node-Gruppe die Funktion ausgeführt werden soll.
strings	Dieses Argument muss mit einem Quelldatenelement verbunden sein, aus dem die eigentlichen Daten stammen. Der bereitgestellte Argumentwert muss eine Sequenz (Null oder viele) von `xs:string` sein.

5.7.6.1.5 min

Gibt den Minimalwert aller numerischen Werte in der Input-Sequenz zurück. Der Minimalwert einer leeren Gruppe ist eine leere Gruppe.



Sprachen

Built-in, C++, C#, Java, XQuery, XSLT 2.0, XSLT 3.0..

Parameter

Argument	Beschreibung
parent-context	Optionales Argument. Liefert den parent-context. Siehe auch Beispiel: Ändern des Parent-Kontexts 83. Das Argument `parent-context` ist ein optionales Argument in einigen MapForce Aggregatfunktionen der core-Bibliothek wie z.B. **`min`**, **`max`**, **`avg`**, **`count`**. Der parent-context bestimmt in einer Quellkomponente mit mehreren hierarchischen Sequenzen, an welcher Node-Gruppe die Funktion ausgeführt werden soll.

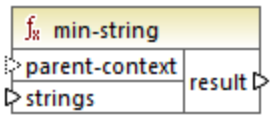
Argument	Beschreibung
values	Dieses Argument muss mit einem Quelldatenelement verbunden sein, aus dem die eigentlichen Daten stammen. Beachten Sie, dass der bereitgestellte Argumentwert numerisch sein muss. Um den Minimalwert einer Sequenz von Strings zu ermitteln, verwenden Sie die Funktion min-string 258.

Beispiel

Siehe Beispiel: Gruppieren von Datensätzen nach Schlüssel 305

5.7.6.1.6 min-string

Gibt den Minimalwert aller String-Werte in der Input-Sequenz zurück. So gibt `min-string("a", "b", "c")` z.B. `"a"` zurück. Die Funktion gibt eine leere Gruppe zurück, wenn das Argument **strings** eine leere Gruppe ist.



Sprachen

Built-in, C++, C#, Java, XQuery, XSLT 2.0, XSLT 3.0..

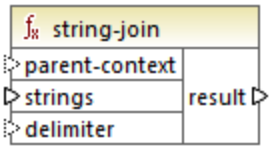
Parameter

Argument	Beschreibung
parent-context	Optionales Argument. Liefert den parent-context. Siehe auch Beispiel: Ändern des Parent-Kontexts 83. Das Argument `parent-context` ist ein optionales Argument in einigen MapForce Aggregatfunktionen der core-Bibliothek wie z.B. `min`, `max`, `avg`, `count`. Der parent-context bestimmt in einer Quellkomponente mit mehreren hierarchischen Sequenzen, an welcher Node-Gruppe die Funktion ausgeführt werden soll.
strings	Dieses Argument muss mit einem Quelldatenelement verbunden sein, aus dem die eigentlichen Daten stammen. Der bereitgestellte Argumentwert muss eine Sequenz (Null oder viele) von `xs:string` sein.

© 2019-2025 Altova GmbH

5.7.6.1.7 string-join

Verkettet alle Werte der Input-Sequenz zu einem String, der durch ein beliebiges von Ihnen ausgewähltes Trennzeichen getrennt ist. Die Funktion gibt einen leeren String zurück, wenn das Argument **strings** eine leere Gruppe ist.



Sprachen

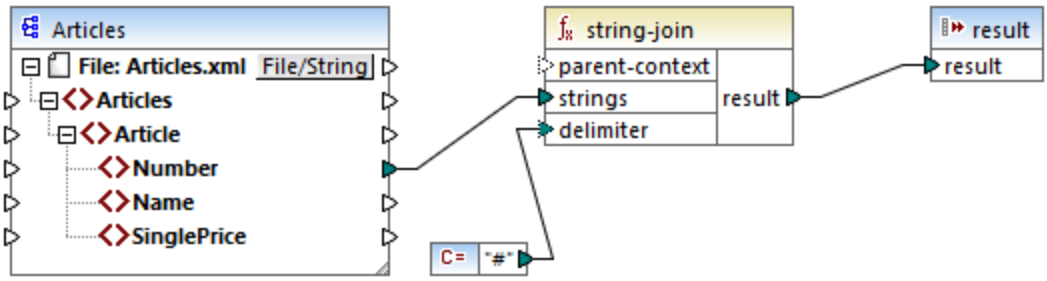
Built-in, C++, C#, Java, XQuery, XSLT 2.0, XSLT 3.0..

Parameter

Argument	Beschreibung
parent-context	Optionales Argument. Liefert den parent-context. Siehe auch Beispiel: Ändern des Parent-Kontexts 83. Das Argument `parent-context` ist ein optionales Argument in einigen MapForce Aggregatfunktionen der core-Bibliothek wie z.B. **`min`**, **`max`**, **`avg`**, **`count`**. Der parent-context bestimmt in einer Quellkomponente mit mehreren hierarchischen Sequenzen, an welcher Node-Gruppe die Funktion ausgeführt werden soll.
strings	Dieses Argument muss mit einem Quelldatenelement verbunden sein, aus dem die eigentlichen Daten stammen. Der bereitgestellte Argumentwert muss eine Sequenz (Null oder viele) von `xs:string` sein.
delimiter	Optionales Argument. Definiert, welches Trennzeichen zwischen zwei aufeinander folgende Strings eingefügt werden soll.

Beispiel

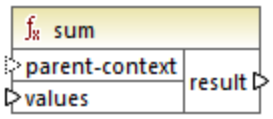
So enthält etwa die XML-Quelldatei vier **Article**-Datenelemente mit den folgenden Zahlen. 1, 2, 3 und 4.



Die Konstante liefert als Trennzeichen eine Raute "#". Das Mapping-Ergebnis lautet daher **1#2#3#4**. Wenn Sie kein Trennzeichen angeben, wird das Ergebnis zu **1234**.

5.7.6.1.8 sum

Gibt die arithmetische Summe aller Werte in der Input-Sequenz zurück. Die Summe einer leeren Gruppe ist Null.



Sprachen

Built-in, C++, C#, Java, XQuery, XSLT 1.0, XSLT 2.0, XSLT 3.0..

Parameter

Argument	Beschreibung
parent-context	Optionales Argument. Liefert den parent-context. Siehe auch Beispiel: Ändern des Parent-Kontexts 83. Das Argument `parent-context` ist ein optionales Argument in einigen MapForce Aggregatfunktionen der core-Bibliothek wie z.B. **`min`**, **`max`**, **`avg`**, **`count`**. Der parent-context bestimmt in einer Quellkomponente mit mehreren hierarchischen Sequenzen, an welcher Node-Gruppe die Funktion ausgeführt werden soll.
values	Dieses Argument muss mit einem Quelldatenelement verbunden sein, aus dem die eigentlichen Daten stammen. Beachten Sie, dass der bereitgestellte Argumentwert numerisch sein muss.

Beispiel

Siehe Beispiel: Summieren von Node-Werten 243.

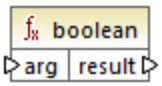
5.7.6.2 core | conversion functions (Konvertierungsfunktionen)

Zur Unterstützung expliziter Datentypkonvertierungen steht in der Bibliothek **conversion** eine ganze Reihe von Konvertierungsfunktionen zur Verfügung. Beachten Sie, dass Konvertierungsfunktionen nicht immer notwendig sind, da MapForce die erforderlichen Konvertierungen in den meisten Fällen automatisch durchführt. Konvertierungsfunktionen dienen normalerweise zum Formatieren von Datums- und Uhrzeitwerten oder zum Vergleichen von Werten. Wenn einige Mapping-Datenelemente z.B. einen unterschiedlichen Typ haben (wie z.B. String und Integer), können Sie mit Hilfe der number 270-Konvertierungsfunktionen einen numerischen Vergleich erzwingen.

 © 2019-2025 Altova GmbH

5.7.6.2.1 boolean

Konvertiert den Wert von **arg** in einen Booleschen Wert. Dies eignet sich für die Arbeit mit logischen Funktionen (wie z.B. `equal`, `greater`, usw.) sowie Filtern und if-else-Bedingungen194. Um den Booleschen Wert **false** zu erhalten, geben Sie als Argument einen leeren String oder den numerischen Wert 0 an. Um den Booleschen Wert **true** zu erhalten, geben Sie als Argument einen nicht leeren String oder den numerischen Wert 1 an.



Sprachen

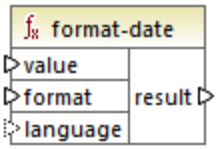
Built-in, C++, C#, Java, XQuery, XSLT 1.0, XSLT 2.0, XSLT 3.0..

Parameter

Argument	Beschreibung
arg	Obligatorisches Argument. Liefert den konvertierende Wert.

5.7.6.2.2 format-date

Konvertiert einen Datumswert vom Typ `xs:date` in einen String und formatiert ihn gemäß den definierten Optionen.



Sprachen

Built-in, C++, C#, Java, XSLT 2.0, XSLT 3.0..

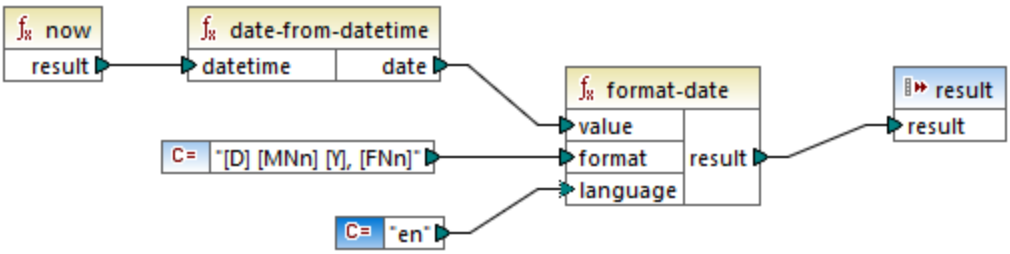
Parameter

Argument	Beschreibung
value	Der zu formatierende `xs:date`-Wert
format	Ein Formatstring, der angibt, wie das Datum formatiert werden soll. Dieses Argument wird auf dieselbe Art, wie das Argument **format** in der Funktion format-dateTime262 verwendet.

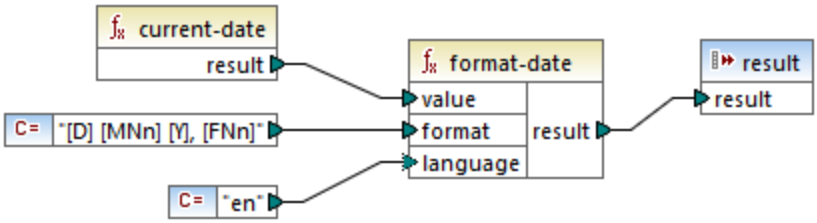
Argument	Beschreibung
language	Optionales Argument. Wird dieses Argument zur Verfügung gestellt, werden der Name des Monats und der Wochentag in einer bestimmten Sprache zurückgegeben. Gültige Werte sind: **de** Deutsch **en (Standardeinstellung)** Englisch **es** Spanisch **fr** Französisch **ja** Japanisch

Beispiel

Im folgenden Mapping wird das aktuelle Datum in einem Format wie "25 March 2020, Wednesday" ausgegeben. Um diesen Wert ins Spanische zu übersetzen, setzen Sie den Wert des Arguments **language** auf **es**.



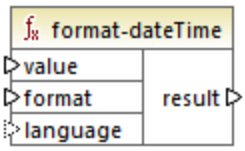
Beachten Sie, dass das obige Mapping für die Transformationssprachen Built-in, C++, C# oder Java erstellt wurde. In XSLT 2.0 kann dasselbe Ergebnis mit dem folgenden Mapping erzielt werden:



5.7.6.2.3 format-dateTime

Konvertiert einen Wert vom Typ `xs:dateTime` in einen String. Die String-Darstellung von Datum und Uhrzeit wird entsprechend dem Wert des Arguments `format` formatiert.

 © 2019-2025 Altova GmbH



Sprachen

Built-in, C++, C#, Java, XSLT 2.0, XSLT 3.0.

Parameter

Argument	Beschreibung
`value`	Der zu formatierenden `xs:dateTime`-Wert.
`format`	Ein Formatstring, der definiert, wie `value` formatiert werden soll. Siehe *Formatierung* weiter unten.
`language`	Optionales Argument. Wird dieses Argument zur Verfügung gestellt, werden der Name des Monats und der Wochentag in einer bestimmten Sprache zurückgegeben. Gültige Werte: `de` Deutsch `en` (Standardeinstellung) Englisch `es` Spanisch `fr` Französisch `ja` Japanisch

Anmerkung: Wenn die Ausgabe der Funktion (result) mit einem Node verbunden ist, dessen Typ nicht `string` ist, kann die Formatierung verloren gehen, da der Wert in den Zieltyp konvertiert wird. Diese automatische Konvertierung kann durch Deaktivieren des Kontrollkästchens *Zielwerte in Zieltypen konvertieren* in den Komponenteneinstellungen 38 der Zielkomponente deaktiviert werden.

Formatierung

Das Argument `format` besteht aus einem String, der so genannte *Variablen-Marker* enthält, die innerhalb von eckigen Klammern stehen, z.B. **`[J]/[M]/[T]`**. Zeichen außerhalb von eckigen Klammern sind Literalzeichen. Wenn das Ergebnis eckige Klammern als Literalzeichen enthalten soll, so sollten diese eckigen Klammern verdoppelt werden.

Jeder Variablen-Marker besteht aus den folgenden Teilen:

- einem Specifier, der angibt, welcher Teil des Datentyps angezeigt werden soll (d.h. Datum oder Uhrzeit),
- einem optionalen Formatierungs-Modifikator,
- einem optionalen Darstellungs-Modifikator und
- einem optionalen Breiten-Modifikator, dem, falls vorhanden, ein Komma vorangestellt werden muss.

```
[variable marker] = [specifier(format)?(presentation)?(width)?]

wobei

width := , min-width ("-" max-width)?
```

Specifier
Die verfügbaren Specifier sind in der Tabelle unten beschrieben.

Specifier	Beschreibung	Standarddarstellung
Y	Jahr (absoluter Wert)	vier Stellen (2010)
M	Monat des Jahres	1-12
D	Tag des Monats	1-31
d	Tag des Jahres	1-366
F	Wochentag	Name des Tages (sprachabhängig)
W	Woche des Jahres	1-53
w	Woche des Monats	1-5
H	Stunde (24 Stunden)	0-23
h	Stunde (12 Stunden)	1-12
P	A.M. oder P.M.	alphabetisch (sprachabhängig)
m	Minuten in Stunde	00-59
s	Sekunden in Minute	00-59
f	Sekundenbruchteile	numerisch, eine Dezimalstelle
Z	Zeitzone, als Zeitabstand von UTC	+08:00
z	Zeitzone als Zeitabstand von GMT	GMT+n

Formatierungs-Modifikatoren
Als Formatierungs-Modifikator kann eines der folgenden Zeichen verwendet werden:

 © 2019-2025 Altova GmbH

Formatierungs-Modifikator	Beschreibung	Beispiel
1	Numerisches Dezimalformat ohne vorangestellte Null:	1, 2, 3
01	Dezimalformat, zwei Stellen:	01, 02, 03
N	Name der Komponente, Großbuchstaben	MONTAG, DIENSTAG
n	Name der Komponente, Kleinbuchstaben	monday, tuesday
Nn	Name der Komponente, Groß/Kleinbuchstaben	Montag, Dienstag

Achtung
Die Modifikatoren **N**, **n** und **Nn** werden von den folgenden Specifiern unterstützt: **M**, **F** und **P**. Die Modifikatoren **N**, **n** und **Nn** werden auch von den **z**-Specifiern unterstützt, jedoch nur in Built-In und bei der Codegenerierung: (*Professional und Enterprise Edition*).

Wenn Sie einen Breiten-Modifier benötigen, stellen Sie diesem ein Komma voran. Der Breiten-Modifier ist eine Zahl, die für die Mindestbreite steht. Optional können Sie einen Bindestrich und eine weitere Zahl, die die Maximalbreite angibt, hinzufügen. Beispiel:

- **`[D,2]`** ist der Tag des Monats mit einer vorangestellten Null (zwei Stellen).
- **`[MNn,3-3]`** ist der Name des Monats in Form einer Abkürzung bestehend aus drei Buchstaben, z.B. *Jan*, *Feb*, *Mar* usw.

Beispiele

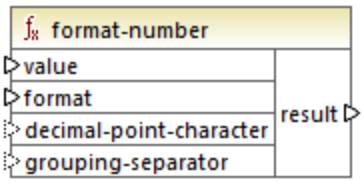
In der nachstehenden Tabelle sehen Sie einige Beispiele für die Formatierung von `xs:dateTime`-Werten mit Hilfe der **`format-dateTime`**-Funktion. In der Spalte *Wert* finden Sie den Wert, der für das Argument `value` bereitgestellt wird. In der Spalte *Format* ist der Wert des Arguments `format` angegeben. In der Spalte *Ergebnis* finden Sie das Ergebnis der Funktion.

Wert	Format	Ergebnis
2003-11-03T00:00:00	`[D]/[M]/[Y]`	3/11/2003
2003-11-03T00:00:00	`[Y]-[M,2]-[D,2]`	2003-11-03
2003-11-03T00:00:00	`[Y]-[M,2]-[D,2] [H,2]:[m]:[s]`	2003-11-03 00:00:00
2010-06-02T08:02:12.054	`[Y] [MNn] [D01] [FNn,3-3] [d] [H]:[m]:[s].[f]`	2010 June 02 Wed 153 8:02:12.054
2010-06-02T08:02:12.054+02:00	`[Y] [MNn] [D01] [FNn,3-3] [d] [H]:[m]:[s].[f] [z]`	2010 June 02 Wed 153 8:02:12.054 GMT+02:00
2010-06-02T08:02:12.054+02:00	`[Y] [MNn] [D1] [FNn] [H]:[m]:[s].[f] [Z]`	2010 June 2 Wednesday 8:02:12.054 +02:00

Wert	Format	Ergebnis
2010-06-02T08:02:12.054	`[Y] [MNn] [D] [FNn,3-3] [H01]:[m]:[s]`	2010 June 2 Wed 08:02:12

5.7.6.2.4 format-number

Konvertiert eine Zahl in einen String und formatiert ihn gemäß den definierten Optionen.



Sprachen

Built-in, C++, C#, Java, XSLT 1.0, XSLT 2.0, XSLT 3.0..

Parameter

Argument	Beschreibung
value	Obligatorisches Argument. Liefert die zu formatierende Zahl.
format	Obligatorisches Argument. Liefert einen Formatierungsstring, der festlegt, wie die Zahl formatiert werden soll. Siehe Anmerkungen weiter unten.
decimal-point-format	Optionales Argument. Liefert das Zeichen, das für das Dezimalzeichen, also im Deutschen das Komma, verwendet wird. Der Standardwert ist der Punkt (.).
grouping-seperator	Optionales Argument. Liefert das Trennzeichen, das zur Trennung von Tausenderstellen verwendet werden soll. Der Standardwert ist das Komma (,).

Anmerkung: Wenn die Ausgabe der Funktion (result) mit einem Node verbunden ist, dessen Typ nicht "string" ist, kann die Formatierung verloren gehen, da der Wert in den Zieltyp konvertiert wird. Diese automatische Konvertierung kann durch Deaktivieren des Kontrollkästchens **Zielwerte in Zieltypen konvertieren** in den Komponenteneinstellungen 38 der Zielkomponente deaktiviert werden.

Anmerkungen

Das Argument **format** hat die folgende Form:

 © 2019-2025 Altova GmbH

```
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)
```

Das erste *subformat* dient zum Formatieren positiver Zahlen, das zweite zum Formatieren negativer Zahlen. Wenn nur ein *subformat* definiert ist, so wird dasselbe Unterformat auch für negative Zahlen verwendet, wobei aber vor dem *prefix* ein Minuszeichen steht.

Sonderzeichen	Standardwert	Beschreibung
zero-digit	0	Eine Stelle, die immer an dieser Stelle im Ergebnis angezeigt wird
digit	#	An dieser Stelle erscheint im Ergebnisstring eine Ziffer, es sei denn, es handelt sich um eine nicht benötigte vorangestellte oder nachgestellte Null.
decimal-point	.	Trennt den Ganzzahlenbereich in der Zahl von den Kommastellen.
grouping-seperator	,	Trennt Zifferngruppen (Tausender-Trennzeichen).
percent-sign	%	Multipliziert die Zahl mit 100 und zeigt sie als Prozentwert an
per-mille	‰	Multipliziert die Zahl mit 1000 und zeigt sie als Promilwert an

In der Tabelle unten finden Sie Beispiele für Formatierungsstrings und deren Ergebnis.

Anmerkung: Die für die Funktion `format-number` verwendete Rundungsmethode ist, Aufrunden ab der Hälfte, d.h. wenn der Wert hinter dem Komma größer oder gleich, 0,5 ist, so wird aufgerundet. Wenn der Wert hinter dem Komma kleiner als 0,5 ist, wird abgerundet. Diese Rundungsmethode gilt nur für generierten Programmcode und für den Built-in-Ausführungsprozessor. In XSLT 1.0 ist die Rundungsmethode nicht definiert. In XSLT 2.0 ist die Rundungsmethod die mathematisch unverzerrte Rundung (half-to-even), also die Auf- oder Abrundung auf die nächste gerade Ziffer.

Zahl	Formatierungsstring	Ergebnis
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

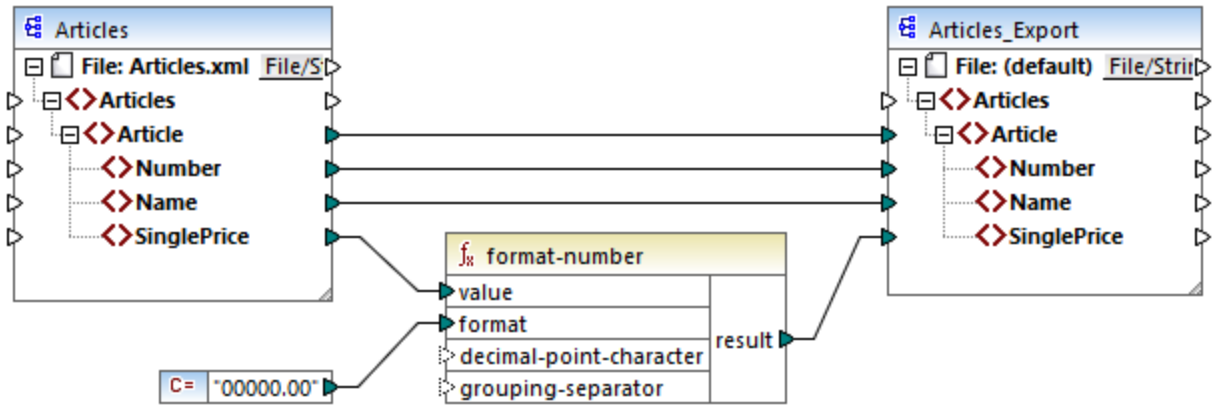
Zahl	Formatierungsstring	Ergebnis
1234.5	###0.0###	1234.5
.00025	###0.0###	0.0003
.00035	###0.0###	0.0004
0.25	#00%	25%
0.736	#00%	74%
1	#00%	100%
-42	#00%	-4200%
-3.12	#.00;(#.00)	(3.12)
-3,12	#.00;#.00CR	3.12CR

Beispiel

Im unten gezeigten Mapping werden Daten aus der XML-Quelldatei gelesen und in eine XML-Zieldatei geschrieben. Die Quellkomponente enthält mehrere **SinglePrice**-Elemente, die die folgenden Dezimalwerte enthalten: **25**, **2.30**, **34**, **57.50**. Das Mapping hat zwei Aufgaben:

1. Allen Werte links Nullen voranstellen, so dass der wichtige Teil genau 5 Stellen hat.
2. Allen Werte rechts Nullen hinzufügen, so dass der Kommastellenteil genau 2 Stellen hat.

Zu diesem Zweck wurde der Formatierungsstring `00000.00` als Argument für die Funktion `format-number` bereitgestellt.



PreserveFormatting.mfd

Die Werte in der Zielkomponente wurden folglich zu:

```
00025.00
00002.30
00034.00
```

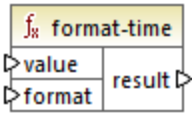
 © 2019-2025 Altova GmbH

```
00057.50
```

Sie finden die Mapping-Design-Datei unter dem folgenden Pfad:
<Dokumente>\Altova\MapForce2025\MapForceExamples\PreserveFormatting.mfd.

5.7.6.2.5 format-time

Konvertiert einen `xs:time`-Eingabewert in einen String.



Sprachen

Built-in, C++, C#, Java, XSLT 2.0, XSLT 3.0..

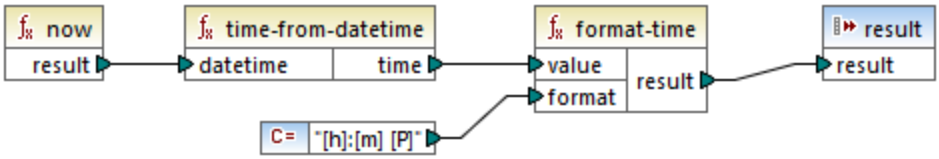
Parameter

Argument	Beschreibung
value	Obligatorisches Argument. Der zu formatierenden `xs:time`-Wert.
format	Obligatorisches Argument. Liefert einen Formatierungsstring. Dieses Argument wird auf dieselbe Art, wie das Argument **format** in der Funktion format-dateTime 262 verwendet.

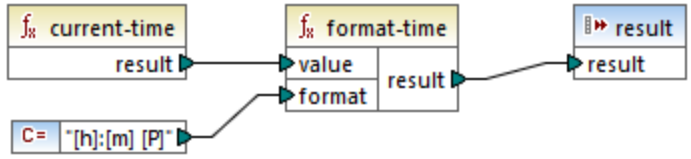
Beispiel

Im folgenden Mapping wird das aktuelle Datum in einem Format wie **`2:15 p.m.`** ausgegeben. Zu diesem Zweck wird der Formatierungsstring **`[h]:[m] [P]`** verwendet, wobei:

- **[h]** die aktuelle Stunde im 12-Stunden-Format ist.
- **[m]** die aktuelle Minute ist
- **[P]** den Teil "a.m." oder "p.m." darstellt

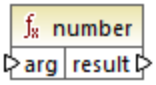


Beachten Sie, dass das obige Mapping für die Transformationssprachen Built-in, C++, C# oder Java erstellt wurde. In XSLT 2.0 kann dasselbe Ergebnis mit dem folgenden Mapping erzielt werden:



5.7.6.2.6 number

Konvertiert den Wert von **arg** in eine Zahl, wobei **arg** ein String oder ein Boolescher Wert ist. Wenn **arg** ein String ist, versucht MapForce ihn als Zahl zu parsen. So wird z.B. ein String wie `"12.56"` in den Dezimalwert `12.56` konvertiert. Wenn **arg** der Boolesche Wert **true** ist,. wird er in das numerische **1** konvertiert. Wenn **arg** der Boolesche Wert **false** ist, wird er in das numerische **0** konvertiert.



Sprachen

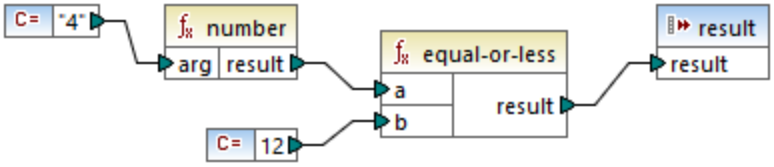
Built-in, C++, C#, Java, XQuery, XSLT 1.0, XSLT 2.0, XSLT 3.0..

Parameter

Argument	Beschreibung
arg	Obligatorisches Argument. Liefert den zu konvertierenden Wert.

Beispiel

Im Beispiel unten hat die erste Konstante den Typ `string` und enthält den String "4". Die zweite Konstante enthält die numerische Konstante 12. Um die beiden Werte als Zahlen vergleichen zu können, muss der Typ übereinstimmen.

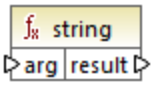


Durch Hinzufügen einer `number`-Funktion zur ersten Konstante wird der String "4" in den numerischen Wert 4 konvertiert. Das Ergebnis des Vergleich ist dann "true". Würde die Funktion `number` nicht verwendet (d.h. wenn "4" direkt mit **a** verbunden würde), würde es zu einem String-Vergleich kommen und das Ergebnis wäre "false".

 © 2019-2025 Altova GmbH

5.7.6.2.7 string

Konvertiert einen Input-Wert in einen String. Sie können mit dieser Funktion auch den Textinhalt eines Node abrufen. Wenn es sich beim Input-Node um einen XML-complexType handelt, so werden alle untergeordneten Nodes ebenfalls als einzelner String ausgegeben.



Sprachen

Built-in, C++, C#, Java, XQuery, XSLT 1.0, XSLT 2.0, XSLT 3.0..

Parameter

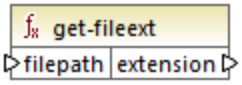
Argument	Beschreibung
arg	Obligatorisches Argument. Liefert den zu konvertierenden Wert.

5.7.6.3 core | file path functions (Dateipfadfunktionen)

Mit Hilfe von **file path**-Funktionen können Sie Dateipfaddaten wie Ordner, Dateinamen und Dateierweiterungen direkt aufrufen und für die weitere Verarbeitung in Ihren Mappings bearbeiten. Diese Funktionen können in allen Sprachen, die von MapForce unterstützt werden, verwendet werden.

5.7.6.3.1 get-fileext

Gibt die Erweiterung des Dateipfads einschließlich des Punktzeichens "." zurück.



Sprachen

Built-in, C++, C#, Java, XQuery, XSLT 1.0, XSLT 2.0, XSLT 3.0..

Parameter

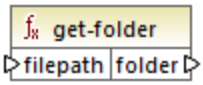
Argument	Beschreibung
filepath	Obligatorisches Argument. Gibt den zu verarbeitenden Dateipfad an.

Beispiel

Wenn Sie "c:\data\Sample.mfd" als Argument angeben, ist das Ergebnis `.mfd`.

5.7.6.3.2 get-folder

Gibt den Ordnernamen des Dateipfads einschließlich des Schrägstrichs oder umgekehrten Schrägstrichs zurück.



Sprachen

Built-in, C++, C#, Java, XQuery, XSLT 1.0, XSLT 2.0, XSLT 3.0..

Parameter

Argument	Beschreibung
filepath	Obligatorisches Argument. Gibt den zu verarbeitenden Dateipfad an.

Beispiel

Wenn Sie "c:\data\Sample.mfd" als Argument angeben, ist das Ergebnis `c:\data\`.

5.7.6.3.3 main-mfd-filepath

Gibt den vollständigen Pfad der Mapping-Design-Datei (.mfd), die das Hauptmapping enthält, zurück. Wenn die mfd-Datei derzeit noch nicht gespeichert ist, wird ein leerer String zurückgegeben.

Sprachen

Built-in, C++, C#, Java, XQuery, XSLT 1.0, XSLT 2.0, XSLT 3.0..

5.7.6.3.4 mfd-filepath

Wenn die Funktion im Hauptmapping aufgerufen wird, gibt die Funktion dasselbe wie die Funktion main-mfd-filepath 272 zurück, also den vollständigen Dateipfad der mfd-Datei, die das Haupt-Mapping enthält. Wenn die mfd-Datei derzeit noch nicht gespeichert ist, wird ein leerer String zurückgegeben. Wenn die Funktion von einer

© 2019-2025 Altova GmbH

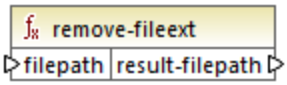
importierten benutzerdefinierten Funktion aus aufgerufen wird, gibt sie den vollständigen Pfad der *importierten* mfd-Datei, die die Definition der benutzerdefinierten Funktion enthält, zurück.

Sprachen

Built-in, C++, C#, Java, XQuery, XSLT 1.0, XSLT 2.0, XSLT 3.0..

5.7.6.3.5 remove-fileext

Entfernt die Erweiterung des Dateipfads einschließlich des Punkts.



Sprachen

Built-in, C++, C#, Java, XQuery, XSLT 1.0, XSLT 2.0, XSLT 3.0..

Parameter

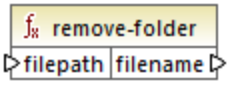
Argument	Beschreibung
filepath	Obligatorisches Argument. Gibt den zu verarbeitenden Dateipfad an.

Beispiel

Wenn Sie "c:\data\Sample.mfd" als Argument angeben, ist das Ergebnis `c:\data\Sample`.

5.7.6.3.6 remove-folder

Entfernt das Verzeichnis des Dateipfads einschließlich des nachgestellten oder umgekehrten Schrägstrichs.



Sprachen

Built-in, C++, C#, Java, XQuery, XSLT 1.0, XSLT 2.0, XSLT 3.0..

Parameter

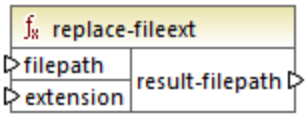
Argument	Beschreibung
filepath	Obligatorisches Argument. Gibt den zu verarbeitenden Dateipfad an.

Beispiel

Wenn Sie "c:\data\Sample.mfd" als Argument angeben, ist das Ergebnis `Sample.mfd`.

5.7.6.3.7 replace-fileext

Ersetzt die durch den **filepath**-Parameter bereitgestellte Erweiterung des Dateipfads durch die Erweiterung, die durch die Verbindung zum **extension**-Parameter bereitgestellt wird.



Sprachen

Built-in, C++, C#, Java, XQuery, XSLT 1.0, XSLT 2.0, XSLT 3.0..

Parameter

Argument	Beschreibung
filepath	Obligatorisches Argument. Gibt den zu verarbeitenden Dateipfad an.
extension	Obligatorisches Argument. Gibt die neue Erweiterung an, die verwendet werden soll.

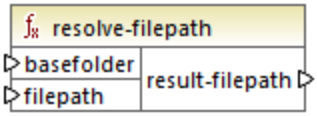
Beispiel

Wenn Sie "c:\data\Sample.log" als **filepath** und ".txt" als **extension** angeben, ist das Ergebnis `c:\data\Sample.txt`.

5.7.6.3.8 resolve-filepath

Löst einen relativen Dateipfad anhand eines Basisordners auf. Die Funktion unterstützt '.' (aktuelles Verzeichnis) und '..' (übergeordnetes Verzeichnis).

© 2019-2025 Altova GmbH



Sprachen

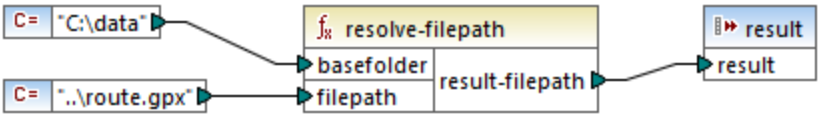
Built-in, C++, C#, Java, XQuery, XSLT 1.0, XSLT 2.0, XSLT 3.0..

Parameter

Argument	Beschreibung
basefolder	Obligatorisches Argument. Gibt das Basisverzeichnis an, relativ zu dem der Pfad aufgelöst werden soll. Dabei kann es sich um einen absoluten oder einen relativen Pfad handeln.
filepath	Obligatorisches Argument. Gibt den aufzulösenden relativen Dateipfad an.

Beispiele

Im unten gezeigten Mapping wird der relative Dateipfad **..\route.gpx** anhand des Verzeichnisses **C:\data** aufgelöst.



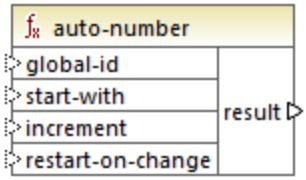
Das Ergebnis des Mappings ist `C:\route.gpx`.

5.7.6.4 core | generator functions (Generierungsfunktionen)

Die **core / generator** Funktionsbibliothek enthält Funktionen zum Generieren von Werten.

5.7.6.4.1 auto-number

Generiert Ganzzahlen in einer Sequenz (z.B. 1,2,3,4,...). Mit Hilfe von Parametern können Sie eine Anfangszahl (Ganzzahl), den Inkrementierungswert und andere Optionen angeben.



Die genaue Reihenfolge, in der Funktionen vom generierten Mapping-Code aufgerufen werden, ist nicht definiert. MapForce kann die berechneten Ergebnisse zur Wiederverwendung im Cache aufbewahren oder Ausdrücke in jeder beliebigen Reihenfolge auswerten. Im Gegensatz zu anderen Funktionen gibt die Funktion `auto-number` ein anderes Ergebnis zurück, wenn sie mehrmals mit denselben Input-Parametern aufgerufen wird. Es wird daher dringend empfohlen, die Funktion `auto-number` mit Vorsicht zu verwenden. In einigen Fällen erzielen Sie mit der Funktion position 318 dasselbe Ergebnis.

Sprachen

Built-in, C++, C#, Java, XQuery, XSLT 1.0, XSLT 2.0, XSLT 3.0..

Parameter

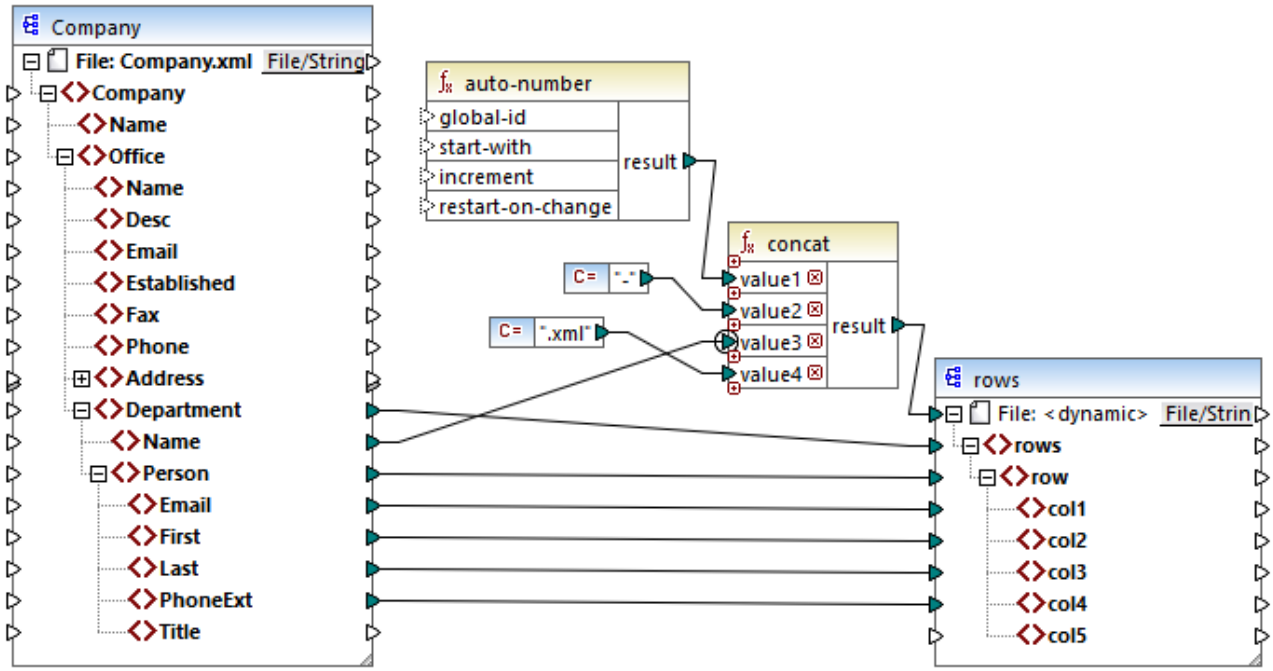
Argument	Beschreibung
global-id	Optionaler Parameter. Wenn ein Mapping mehrere `auto-number`-Funktionen enthält, generieren diese Sequenzen mit doppelt vorhandenen Zahlen. Damit alle `auto-number`-Funktionen voneinander wissen und dadurch keine Sequenzen mit doppelten Werten generieren, verbinden Sie einen gemeinsamen String (z.B. eine Konstante) mit dem **global-id**-Input jeder `auto-number`-Funktion.
start-with	Optionaler Parameter. Definiert die Ganzzahl, mit der die generierte Sequenz beginnt. Der Standardwert ist **1**.
increment	Optionaler Parameter. Definiert den Inkrementierungswert. Der Standardwert ist **1**.
restart-on-change	Optionaler Parameter. Setzt den Zähler auf **start-with** zurück, wenn sich der Inhalt des damit verbundenen Datenelements ändert.

Beispiel

Das folgende Mapping ist eine Variante des im Beispiel: Ändern des Parent-Kontexts 83 beschriebenen Mappings **ParentContext.mfd**.

Ziel des unten gezeigten Mappings ist es, mehrere XML-Dateien zu generieren, eine für jede Abteilung (department) in der XML-Quelldatei. Es gibt einige Abteilungen mit demselben Namen (da diese zu unterschiedlichen übergeordneten Büros gehören). Aus diesem Grund muss jeder generierte Dateiname mit einer sequenziellen Nummerierung beginnen, z.B. **1-Administration.xml**, **2-Marketing.xml**, usw.

 © 2019-2025 Altova GmbH



Um das gewünschte Mapping-Resultat zu erhalten, wurde die Funktion **auto-number** verwendet. Das Ergebnis dieser Funktion wird mit Hilfe eines Bindestrichs, gefolgt vom Abteilungsnamen, gefolgt vom String ".xml" verkettet, um für die generierte Datei einen eindeutigen Namen zu erstellen. Beachten Sie, dass auf den dritten Parameter der **concat**-Funktion (den Abteilungsnamen) ein Prioritätskontext 87 angewendet wurde. Dadurch wird die Funktion **auto-number** im Kontext jeder einzelnen Abteilung aufgerufen und erzeugt die erforderlichen Sequenzwerte. Würde kein Prioritätskontext verwendet, würde die Funktion **auto-number** (in Abwesenheit irgendeines Kontexts) immer wieder die Zahl 1 generieren, wodurch doppelt vorhandene Dateinamen generiert würden.

5.7.6.5 core | logical functions (logische Funktionen)

Logische Funktionen dienen (im Allgemeinen) dazu, Input-Daten miteinander zu vergleichen und als Ergebnis den Boolschen Wert "true" oder "false" zurückzugeben. Normalerweise werden Daten damit überprüft, bevor sie mittels eines Filters 194 an eine Untergruppe der Zielkomponente übergeben werden. Nahezu alle logischen Funktionen haben die folgende Struktur:

Input-Parameter = **a** | **b** oder **value1** | **value2**
Output-Parameter = result

Das ausgewertete Ergebnis hängt sowohl von den Eingabewerten als auch von den für den Vergleich verwendeten Datentypen ab. Der Vergleich "kleiner als" der Integerwerte **4** und **12** ergibt den Booleschen Wert "true", da 4 kleiner als 12 ist. Wenn die beiden Input-Parameter die String-Werte **4** und **12** enthalten, ist das Ergebnis der lexikalischen Analyse der Output-Wert "false", da '4' alphabetisch größer als das erste Zeichen '1' des zweiten Operanden (12) ist.

Wenn es sich bei allen Input-Werten um denselben Datentyp handelt, wird der Vergleich für den gemeinsamen Typ durchgeführt. Wenn die Input-Werte unterschiedliche Typen haben (z.B. `integer` und `string` oder `string`

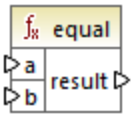
und `date`), wird als Datentyp für den Vergleich der allgemeingültigste (am wenigsten restriktive) der beiden verwendet.

Bevor zwei Werte unterschiedlichen Typs verglichen werden, werden alle Werte in einen gemeinsamen Datentyp konvertiert. Um beim vorhergehenden Beispiel zu bleiben: Der Datentyp `string` ist weniger restriktiv als `integer`. Wenn Sie den Integer-Wert **4** mit dem String **"12"**' vergleichen, wird der Integer-Wert **4** in den String **"4"** konvertiert, der anschließend mit dem String**"12"** verglichen wird.

Anmerkung: Logische Funktionen können nicht zum Prüfen des Vorhandenseins von Null-Werten verwendet werden. Wenn Sie einen Null-Wert als Argument für eine logische Funktion bereitstellen, gibt diese einen Null-Wert zurück. Nähere Informationen zur Behandlung von Nullwerten finden Sie unter Nullwerte / Nillable Werte 138.

5.7.6.5.1 equal

Die `equal`-Funktion (*siehe Abbildung unten*) gibt den Booleschen Wert `true` zurück, wenn **a** gleich **b**; false ist; gibt andernfalls `false` zurück. Die Groß- und Kleinschreibung wird beim Vergleich berücksichtigt.



Beispiel:

```
a = hi
b = hi
```

In diesem Beispiel sind beide Werte gleich. Daher ist das Ergebnis `true`. Wenn `b` z.B. gleich `Hi` wäre, wäre das Ergebnis der Funktion `false`.

Sprachen

Built-in, C++, C#, Java, XQuery, XSLT 1.0, XSLT 2.0, XSLT 3.0.

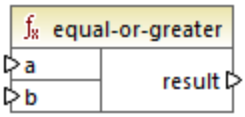
Parameter

Argument	Beschreibung
a	Obligatorischer Parameter. Gibt den ersten zu vergleichenden Wert an.
b	Obligatorischer Parameter. Gibt den zweiten zu vergleichenden Wert an.

 © 2019-2025 Altova GmbH

5.7.6.5.2 equal-or-greater

Gibt den Booleschen Wert **true** zurück, wenn *a* größer oder gleich *b* ist; gibt andernfalls **false** zurück.



Sprachen

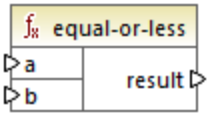
Built-in, C++, C#, Java, XQuery, XSLT 1.0, XSLT 2.0, XSLT 3.0..

Parameter

Argument	Beschreibung
a	Obligatorischer Parameter. Gibt den ersten zu vergleichenden Wert an.
b	Obligatorischer Parameter. Gibt den zweiten zu vergleichenden Wert an.

5.7.6.5.3 equal-or-less

Gibt den Booleschen Wert **true** zurück, wenn *a* kleiner oder gleich *b* ist; gibt andernfalls **false** zurück.



Sprachen

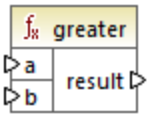
Built-in, C++, C#, Java, XQuery, XSLT 1.0, XSLT 2.0, XSLT 3.0..

Parameter

Argument	Beschreibung
a	Obligatorischer Parameter. Gibt den ersten zu vergleichenden Wert an.
b	Obligatorischer Parameter. Gibt den zweiten zu vergleichenden Wert an.

5.7.6.5.4 greater

Gibt den Booleschen Wert **true** zurück, wenn *a* größer als *b* ist; gibt andernfalls **false** zurück.



Sprachen

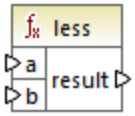
Built-in, C++, C#, Java, XQuery, XSLT 1.0, XSLT 2.0, XSLT 3.0..

Parameter

Argument	Beschreibung
a	Obligatorischer Parameter. Gibt den ersten zu vergleichenden Wert an.
b	Obligatorischer Parameter. Gibt den zweiten zu vergleichenden Wert an.

5.7.6.5.5 less

Gibt den Booleschen Wert **true** zurück, wenn *a* kleiner als *b* ist; gibt andernfalls **false** zurück.



Sprachen

Built-in, C++, C#, Java, XQuery, XSLT 1.0, XSLT 2.0, XSLT 3.0..

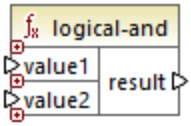
Parameter

Argument	Beschreibung
a	Obligatorischer Parameter. Gibt den ersten zu vergleichenden Wert an.
b	Obligatorischer Parameter. Gibt den zweiten zu vergleichenden Wert an.

© 2019-2025 Altova GmbH

5.7.6.5.6 logical-and

Gibt den Booleschen Wert **true** nur dann zurück, wenn jeder Input-Wert true ist; gibt andernfalls **false** zurück. Sie können das Ergebnis mit einer weiteren `logical-and`-Funktion verbinden und dadurch eine beliebig gewählte Anzahl von Bedingungen mit einem logischen UND verbinden, um zu überprüfen, ob alle **true** zurückgeben. Diese Funktion außerdem erweitert werden und zusätzliche Argumente erhalten, siehe Hinzufügen oder Löschen von Funktionsargumenten 214.



Sprachen

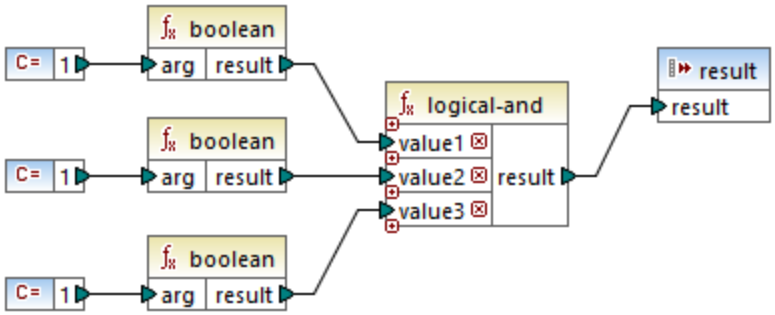
Built-in, C++, C#, Java, XQuery, XSLT 1.0, XSLT 2.0, XSLT 3.0..

Parameter

Argument	Beschreibung
value1	Obligatorischer Parameter. Gibt den ersten zu vergleichenden Wert an.
value2	Obligatorischer Parameter. Gibt den zweiten zu vergleichenden Wert an.

Beispiel

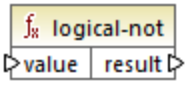
Das Ergebnis im unten gezeigten Mapping ist **true**, da alle Input-Werte für die `logical-and`-Funktion ebenfalls **true** sind. Wäre einer der Input-Werte **false**, wäre auch das Ergebnis des Mappings **false**.



Siehe auch Beispiel: Look-up und Verkettung 235.

5.7.6.5.7 logical-not

Invertiert oder spiegelt das logische Ergebnis des Input-Werts. Wenn *value* z.B. **true** ist, ist das Ergebnis der "false". Wenn *value* **false** ist, ist das Ergebnis "true".



Sprachen

Built-in, C++, C#, Java, XQuery, XSLT 1.0, XSLT 2.0, XSLT 3.0..

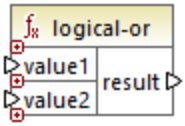
Parameter

Argument	Beschreibung
value	Obligatorischer Parameter. Liefert den Input-Wert.

5.7.6.5.8 logical-or

Bei dieser Funktion müssen beide Input-Werte Boolesche Werte sein. Wenn zumindest einer der Input-Werte **true** ist, ist das Ergebnis **true**. Andernfalls ist das Ergebnis **false**.

Diese Funktion kann außerdem erweitert werden und zusätzliche Argumente erhalten, siehe Hinzufügen oder Löschen von Funktionsargumenten 214.



Sprachen

Built-in, C++, C#, Java, XQuery, XSLT 1.0, XSLT 2.0, XSLT 3.0..

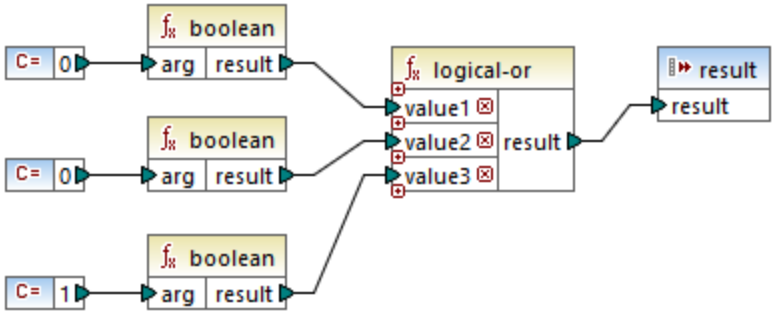
Parameter

Argument	Beschreibung
value1	Obligatorischer Parameter. Gibt den ersten zu vergleichenden Wert an.
value2	Obligatorischer Parameter. Gibt den zweiten zu vergleichenden Wert an.

Beispiel

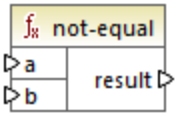
Das Ergebnis des unten gezeigten Mappings ist **true**, da mindestens eines der Argumente der Funktion **true** ist.

 © 2019-2025 Altova GmbH



5.7.6.5.9 not-equal

Gibt den Booleschen Wert **true** zurück, wenn *a* nicht gleich *b* ist; gibt andernfalls **false** zurück.



Sprachen

Built-in, C++, C#, Java, XQuery, XSLT 1.0, XSLT 2.0, XSLT 3.0..

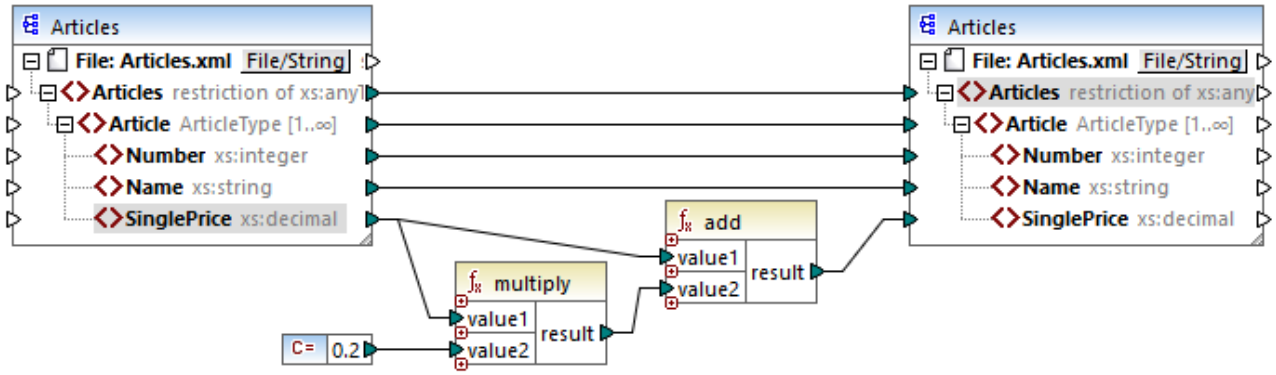
Parameter

Argument	Beschreibung
a	Obligatorischer Parameter. Gibt den ersten zu vergleichenden Wert an.
b	Obligatorischer Parameter. Gibt den zweiten zu vergleichenden Wert an.

5.7.6.6 core | math functions (mathematische Funktionen)

Mathematische Funktionen dienen zur Durchführung grundlegender mathematischer Operationen an Daten. Beachten Sie, dass diese Funktionen nicht zur Berechnung einer Zeitdauer oder zur Berechnung von `datetime`-Werten verwendet werden können.

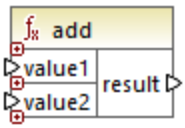
Die meisten mathematischen Funktionen erhalten zwei Input-Parameter (**value1**, **value2**), die Operanden der mathematischen Operation sind. Die Input-Werte werden zur weiteren Verarbeitung automatisch in Werte vom Typ `decimal` (Dezimalwerte) konvertiert. Das Ergebnis von mathematischen Funktionen hat ebenfalls den Typ `decimal`.



Im obigen Beispiel werden 20 % Umsatzsteuer zu den einzelnen auf die Zielkomponente gemappten Artikeln hinzugefügt.

5.7.6.6.1 add

Addiert **value1** zu **value2** und gibt das Ergebnis als Dezimalwert zurück. Diese Funktion kann außerdem erweitert werden und zusätzliche Argumente erhalten, siehe Hinzufügen oder Löschen von Funktionsargumenten 214.



Sprachen

Built-in, C++, C#, Java, XQuery, XSLT 1.0, XSLT 2.0, XSLT 3.0..

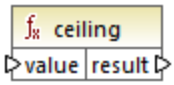
Parameter

Argument	Beschreibung
value1	Obligatorischer Parameter. Gibt den ersten Operanden an.
value2	Obligatorischer Parameter. Gibt den zweiten Operanden an.

© 2019-2025 Altova GmbH

5.7.6.6.2 ceiling

Gibt die kleinste Ganzzahl zurück, die größer oder gleich **value** ist, zurück.



Sprachen

Built-in, C++, C#, Java, XQuery, XSLT 1.0, XSLT 2.0, XSLT 3.0..

Parameter

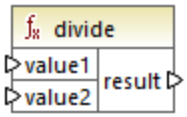
Argument	Beschreibung
value	Obligatorischer Parameter. Liefert den Input-Wert der Funktion.

Beispiel

Wenn der Input-Wert **11.2** ist, wird das Ergebnis durch Anwendung der `ceiling`-Funktion zu **12**, d.h. zur kleinsten Ganzzahl, die größer als **11.2** ist.

5.7.6.6.3 divide

Dividiert **value1** durch **value2** und gibt das Ergebnis als Dezimalwert zurück. Die Präzision des Ergebnisses hängt von der Zielsprache ab. Mit Hilfe der round-precision 288 Funktion können Sie die Präzision des Ergebnisses definieren.



Sprachen

Built-in, C++, C#, Java, XQuery, XSLT 1.0, XSLT 2.0, XSLT 3.0..

Parameter

Argument	Beschreibung
value1	Obligatorischer Parameter. Gibt den ersten Operanden an.
value2	Obligatorischer Parameter. Gibt den zweiten Operanden an.

5.7.6.6.4 floor

Gibt die größte Ganzzahl zurück, die kleiner oder gleich **value** ist, zurück.

Sprachen

Built-in, C++, C#, Java, XQuery, XSLT 1.0, XSLT 2.0, XSLT 3.0..

Parameter

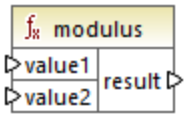
Argument	Beschreibung
value	Obligatorischer Parameter. Liefert den Input-Wert der Funktion.

Beispiel

Wenn der Input-Wert **11.7** ist, wird das Ergebnis durch Anwendung der `floor`-Funktion zu **11**, d.h. zur größten Ganzzahl, die kleiner als **11.7** ist.

5.7.6.6.5 modulus

Gibt den Rest, der sich bei der Division von **value1** durch **value2** ergibt, zurück.



Sprachen

Built-in, C++, C#, Java, XQuery, XSLT 1.0, XSLT 2.0, XSLT 3.0..

Parameter

Argument	Beschreibung
value1	Obligatorischer Parameter. Gibt den ersten Operanden an.
value2	Obligatorischer Parameter. Gibt den zweiten Operanden an.

© 2019-2025 Altova GmbH

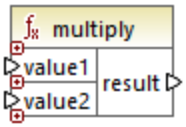
Beispiel

Wenn die Input Werte **1.5** und **1** sind, ist das Ergebnis der `modulus`-Funktion **0.5**. Die Erklärung dafür ist, dass bei `1.5 / 1` der Rest **0.5** bleibt.

Wenn die Input-Werte **9** und **3** sind, ist das Ergebnis **0**, da bei `9 / 3` kein Rest bleibt.

5.7.6.6.6 multiply

Multipliziert **value1** mit **value2** und gibt das Ergebnis als Dezimalwert zurück.



Sprachen

Built-in, C++, C#, Java, XQuery, XSLT 1.0, XSLT 2.0, XSLT 3.0..

Parameter

Argument	Beschreibung
value1	Obligatorischer Parameter. Gibt den ersten Operanden an.
value2	Obligatorischer Parameter. Gibt den zweiten Operanden an.

5.7.6.6.7 round

Gibt den auf die nächste Ganzzahl gerundeten Wert zurück. Wenn sich der Wert genau zwischen zwei Ganzzahlen befindet, wird der Algorithmus "Rundung Richtung positiv unendlich" verwendet. So wird z.B. der Wert "10,5" auf "11" und der Wert "-10,5" auf "-10" gerundet.

Sprachen

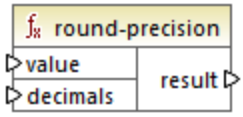
Built-in, C++, C#, Java, XQuery, XSLT 1.0, XSLT 2.0, XSLT 3.0..

Parameter

Argument	Beschreibung
value	Obligatorischer Parameter. Liefert den Input-Wert der Funktion.

5.7.6.6.8 round-precision

Rundet den Input-Wert auf *N* Dezimalstellen, wobei *N* das Argument **decimals** ist.



Sprachen

Built-in, C++, C#, Java.

Parameter

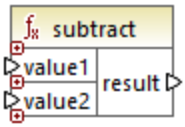
Argument	Beschreibung
value	Obligatorischer Parameter. Liefert den Input-Wert der Funktion.
decimals	Obligatorischer Parameter. Definiert die Anzahl der Stellen, auf die gerundet werden soll.

Beispiel

Die Rundung des Werts **2,777777** auf 2 Dezimalstellen ergibt **2,78**. Die Rundung des Werts **0,1234** auf 3 Dezimalstellen ergibt **0,123**.

5.7.6.6.9 subtract

Subtrahiert **value2** von **value1** und gibt das Ergebnis als Dezimalwert zurück.



Sprachen

Built-in, C++, C#, Java, XQuery, XSLT 1.0, XSLT 2.0, XSLT 3.0..

© 2019-2025 Altova GmbH

Parameter

Argument	Beschreibung
value1	Obligatorischer Parameter. Gibt den ersten Operanden an.
value2	Obligatorischer Parameter. Gibt den zweiten Operanden an.

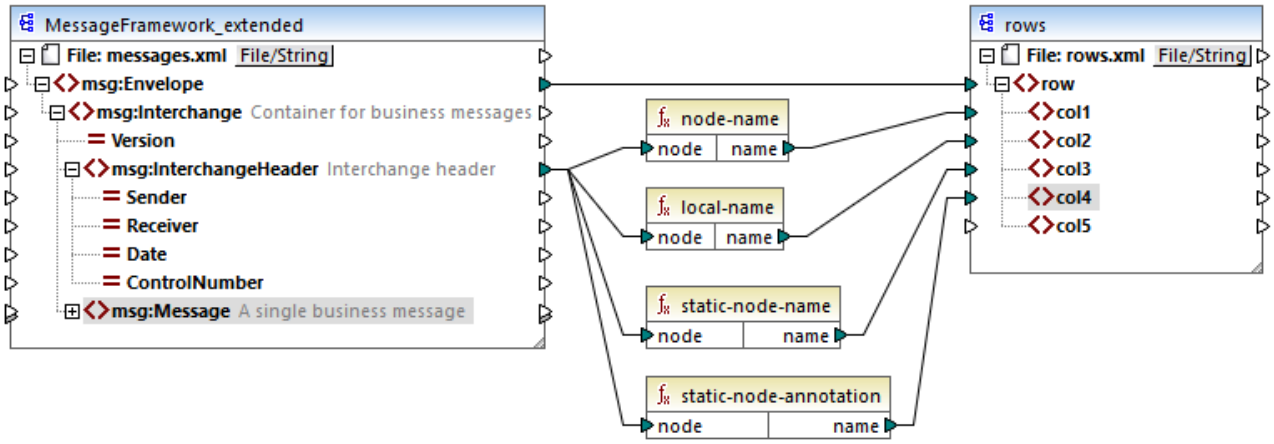
5.7.6.7 core | node functions (Node-Funktionen)

Mit Hilfe der Funktionen aus den **core | node-Funktionen** können Sie Informationen über Nodes in einer Mapping-Komponente (wie z.B. den Node-Namen oder die Annotation) aufrufen oder nillable Elemente verarbeiten, siehe auch Nullwerte / Nillable Werte 138.

Es gibt auch eine alternative Methode, um Node-Namen aufzurufen, für die gar keine Node-Funktionen benötigt werden, siehe Mappen von Node-Namen 405.

Im unten gezeigten Mapping sehen Sie einige Node-Funktionen, die Informationen aus dem Node **msg:InterchangeHeader** der XML-Quelldatei abrufen. Dabei werden die folgenden Informationen extrahiert:

1. Die Funktion **`node-name`** gibt den qualifizierten Namen des Node einschließlich des Node-Präfix zurück.
2. Die Funktion **`local-name`** gibt nur den lokalen Teil zurück.
3. Die Funktion **`static-node-name`** ist ähnlich der Funktion **`node-name`**, steht aber auch in XSLT 1.0 zur Verfügung.
4. Die Funktion **`static-node-annotation`** ruft die gemäß dem XML-Schema definierte Annotation des Elements ab.



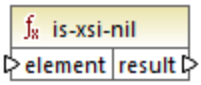
Das Mapping-Ergebnis ist das folgende (ausschließlich der XML- und Namespace-Deklaration):

```
<row>
   <col1>msg:InterchangeHeader</col1>
   <col2>InterchangeHeader</col2>
```

```
    <col3>msg:InterchangeHeader</col3>
    <col4>Interchange header</col4>
</row>
```

5.7.6.7.1 is-xsi-nil

Gibt **true** zurück, wenn das `xsi:nil`-Attribut des **element**-Node auf **true** gesetzt ist.



Sprachen

Built-in, C++, C#, Java, XQuery, XSLT 1.0, XSLT 2.0, XSLT 3.0..

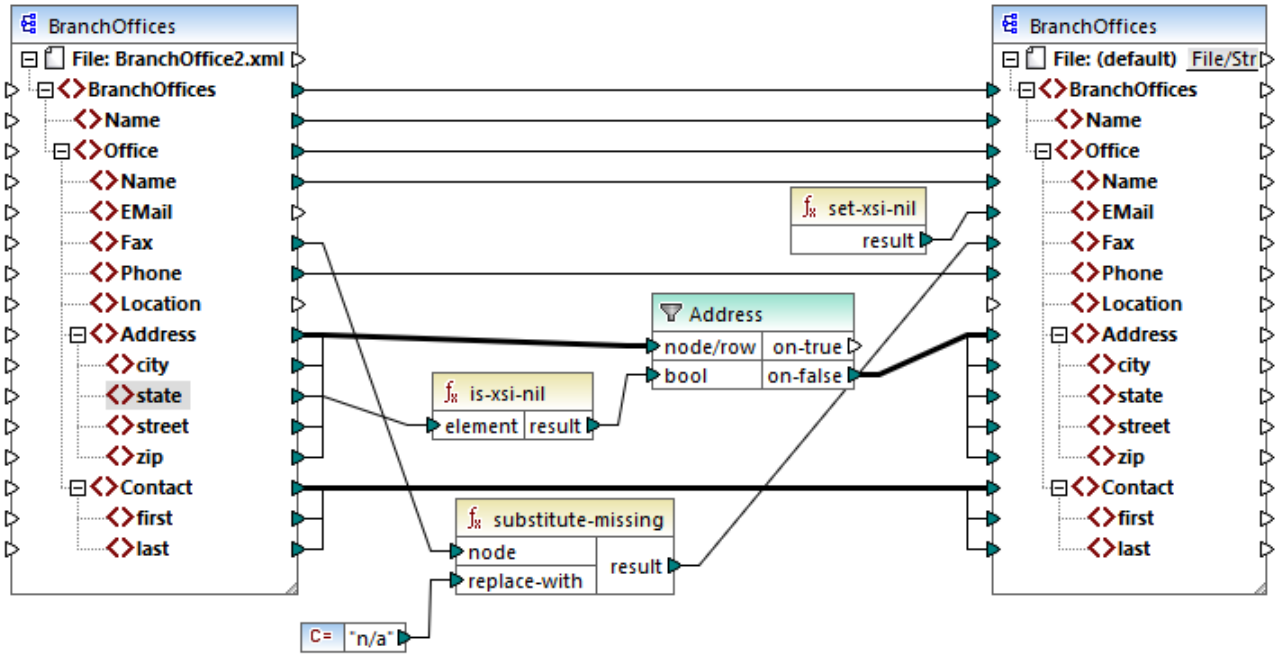
Parameter

Argument	Beschreibung
element	Obligatorischer Parameter. Muss mit dem zu überprüfenden Quell-Node verbunden sein.

Beispiel

Im unten gezeigten Mapping-Design werden Daten auf Basis von Bedingungen aus einer XML-Quelldatei in eine XML-Zieldatei kopiert. Außerdem wird hier die Verwendung einer Reihe von Funktionen, darunter der Funktion **`is-xsi-nil`** veranschaulicht. Das Mapping hat den Namen **HandlingXsiNil.mfd** und befindet sich im Verzeichnis **<Dokumente>\Altova\MapForce2025\MapForceExamples**.

 © 2019-2025 Altova GmbH



Wie oben gezeigt, überprüft die Funktion **`is-xsi-nil`**, ob das Attribut `xsi:nil` für das Datenelement **state** der Quelldatei "true" ist. Wenn dieses Attribut "false" ist, kopiert der Filter das übergeordnete **Address**-Element in die Zielkomponente. Die XML-Quelldatei sieht folgendermaßen aus (ausschließlich der XML- und Namespace-Deklaration):

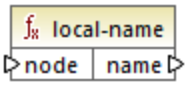
```
<BranchOffices>
   <Name>Nanonull</Name>
   <Office>
      <Name>Nanonull Research Outpost</Name>
      <EMail>sp@nanonull.com</EMail>
      <Fax xsi:nil="true"/>
      <Phone>+8817 3141 5926</Phone>
      <Address>
         <city>South Pole</city>
         <state xsi:nil="true"/>
         <street xsi:nil="true"/>
         <zip xsi:nil="true"/>
      </Address>
      <Contact>
         <first>Scott</first>
         <last>Amundsen</last>
      </Contact>
   </Office>
</BranchOffices>
```

Das Ergebnis des Mappings ist, dass gar kein **Address**-Element in die Zielkomponente kopiert wird, da die Quelldatei nur ein **Address**-Element enthält und dessen Attribut `xsi:nil` für das Element **state** auf "true" gesetzt ist. Die Mapping-Ausgabe sieht daher folgendermaßen aus:

```
<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>
```

5.7.6.7.2 local-name

Gibt den lokalen Namen des Node zurück. Im Gegensatz zur Funktion node-name 292 gibt **local-name** das Präfix des Node nicht zurück. Wenn der Node kein Präfix hat, geben **local-name** und **node-name** denselben Wert zurück.



Sprachen

Built-in, C++, C#, Java, XQuery, XSLT 1.0, XSLT 2.0, XSLT 3.0..

Parameter

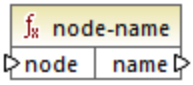
Argument	Beschreibung
node	Obligatorischer Parameter. Verbinden Sie diesen Input mit den Node, dessen Namen Sie abrufen möchten.

5.7.6.7.3 node-name

Gibt den qualifizierten Namen (QName) des verbundenen Node zurück. Wenn es sich um einen XML **text()**-Node handelt, wird ein leerer QName zurückgegeben. Die Funktion funktioniert nur mit Nodes, die einen Namen haben. Wenn XSLT 2.0 die Zielsprache ist (die **fn:node-name** aufruft), wird eine leere Sequenz für Nodes, die keinen Namen haben, zurückgegeben.

Anmerkung: Der Abruf des Node-Namens wird für "Datei Input"-Nodes, Datenbanktabellen oder -felder, XBRL, Excel, JSON, oder Protocol Buffer-Felder nicht unterstützt.

© 2019-2025 Altova GmbH



Sprachen

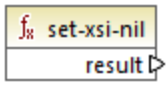
Built-in, C++, C#, Java, XQuery, XSLT 2.0, XSLT 3.0..

Parameter

Argument	Beschreibung
node	Obligatorischer Parameter. Verbinden Sie diesen Input mit den Node, dessen Namen Sie abrufen möchten.

5.7.6.7.4 set-xsi-nil

Setzt den Ziel-Node auf xsi:nil.



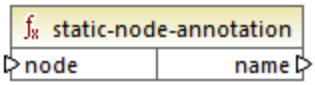
Sprachen

Built-in, C++, C#, Java, XQuery, XSLT 2.0, XSLT 3.0..

5.7.6.7.5 static-node-annotation

Gibt den String mit der Annotation des verbundenen Node zurück. Beim Input muss es sich: (i) um eine Quellkomponente oder (ii) um eine benutzerdefinierte Funktion vom Typ inline 222 handeln, die direkt mit einerm Parameter 225 verbunden ist, der wiederum direkt mit einem Node im aufrufenden Mapping verbunden ist.

Die Verbindung muss direkt erfolgen und darf nicht über einen Filter oder eine reguläre benutzerdefinierte (Nicht-Inline gesetzte) Funktion laufen. Dies ist eine Pseudofunktion, die bei der Generierung durch den aus dem verbundenen Node abgerufenen Text ersetzt wird und daher für alle Programmiersprachen zur Verfügung steht.



Sprachen

Built-in, C++, C#, Java, XQuery, XSLT 1.0, XSLT 2.0, XSLT 3.0..

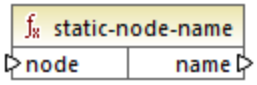
Parameter

Argument	Beschreibung
node	Obligatorischer Parameter. Verbinden Sie diesen Input mit den Node, dessen Annotation Sie abrufen möchten.

5.7.6.7.6 static-node-name

Gibt den String mit dem Names des damit verbundenen Node zurück. Beim Input muss es sich: (i) um eine Quellkomponente oder (ii) um eine benutzerdefinierte Funktion vom Typ inline 222 handeln, die direkt mit einerm Parameter 225 verbunden ist, der wiederum direkt mit einem Node im aufrufenden Mapping verbunden ist.

Die Verbindung muss direkt erfolgen und darf nicht über einen Filter oder eine benutzerdefinierte (nicht-Inline gesetzte) Funktion laufen. Dies ist eine Pseudofunktion, die bei der Generierung durch den aus dem verbundenen Node abgerufenen Text ersetzt wird und daher für alle Programmiersprachen zur Verfügung steht.



Sprachen

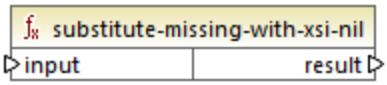
Built-in, C++, C#, Java, XQuery, XSLT 1.0, XSLT 2.0, XSLT 3.0..

Parameter

Argument	Beschreibung
node	Obligatorischer Parameter. Verbinden Sie diesen Input mit den Node, dessen Namen Sie abrufen möchten.

5.7.6.7.7 substitute-missing-with-xsi-nil

Ersetzt bei Nodes mit Simple Content alle fehlenden (oder Null-Werte) der Quellkomponente im Ziel-Node durch das `xsi:nil` Attribut.



Sprachen

Built-in, C++, C#, Java, XQuery, XSLT 2.0, XSLT 3.0..

© 2019-2025 Altova GmbH

Parameter

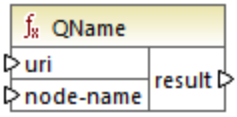
Argument	Beschreibung
input	Obligatorischer Parameter. Verbinden Sie diesen Input mit den Node, dessen Namen Sie abrufen möchten.

5.7.6.8 core | QName functions (QName-Funktionen)

QName--Funktionen bieten eine Methode, um die qualifizierten Namen (QName) in XML-Dokumenten zu bearbeiten.

5.7.6.8.1 QName

Konstruiert anhand einer Namespace URI und eines lokalen Teils einen QName. Mit Hilfe dieser Funktion können Sie einen QName in einer Zielkomponente erstellen. Die Parameter **uri** und **node-name** können mit Hilfe einer Konstantenfunktion bereitgestellt werden.



Sprachen

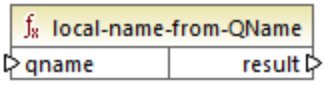
Built-in, C++, C#, Java, XQuery, XSLT 2.0, XSLT 3.0..

Parameter

Name	Beschreibung
uri	Obligatorisch. Stellt die URI bereit.
node-name	Obligatorisch. Liefert den Namen des Node.

5.7.6.8.2 local-name-from-QName

Extrahiert den lokalen Namensteil aus einem Wert vom Typ `xs:QName`. Beachten Sie, dass diese Funktion im Gegensatz zur Funktion local-name 292, die den lokalen Namen des *Node*zurückgibt, den *Inhalt* des mit dem Input **qname** verbundenen Datenelements zurückgibt.



Sprachen

Built-in, C++, C#, Java, XQuery, XSLT 2.0, XSLT 3.0..

Parameter

Name	Beschreibung
qname	Obligatorisch. Liefert den Input-Wert der Funktion, vom Typ `xs:QName`.

5.7.6.8.3 namespace-uri-from-QName

Gibt den Namespace URI-Teil des als Argument bereitgestellten QName-Werts zurück.

Sprachen

Built-in, C++, C#, Java, XQuery, XSLT 2.0, XSLT 3.0..

Parameter

Name	Beschreibung
qname	Obligatorisch. Liefert den Input-Wert der Funktion.

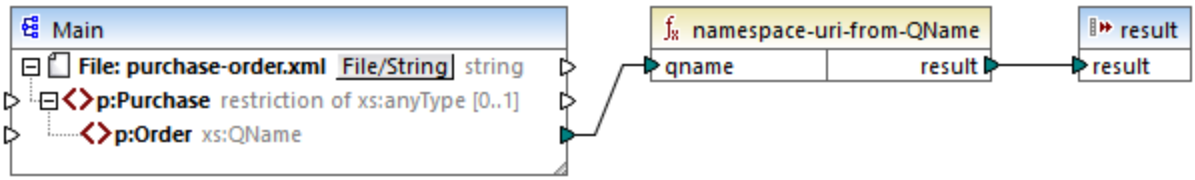
Beispiel

Die folgende XML-Datei enthält den QName-Wert **o:name**. Beachten Sie, dass das Präfix "o" auf den Namespace **`http://NamespaceTest.com/Order`** gemappt ist.

```
<?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>
```

Unten sehen Sie ein Mapping, in dem der QName-Wert verarbeitet wird und die Namespace URI abgerufen wird:

 © 2019-2025 Altova GmbH



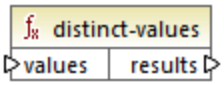
Die Ausgabe des Mappings ist `http://NamespaceTest.com/Order`.

5.7.6.9 core | sequence functions (Sequenzfunktionen)

Mit Hilfe von Sequenzfunktionen können Input-Sequenzen [78] verarbeitet und ihr Inhalt gruppiert werden.

5.7.6.9.1 distinct-values

Verarbeitet die mit dem **values** Input verbundene Wertesequenz und gibt nur die eindeutigen Werte als Sequenz zurück. Dies dient zum Entfernen doppelt vorhandener Werte aus einer Sequenz und zum Mappen der eindeutigen Datenelemente auf die Zielkomponente.



Sprachen

Built-in, C++, C#, Java, XQuery, XSLT 2.0, XSLT 3.0..

Parameter

Name	Beschreibung
values	Mit diesem Input muss ein Mapping-Datenelement verbunden werden, das eine Sequenz [78] von null oder mehr Werten liefert. So kann damit etwa ein Datenelement aus einer XML-Quelldatei verbunden werden.

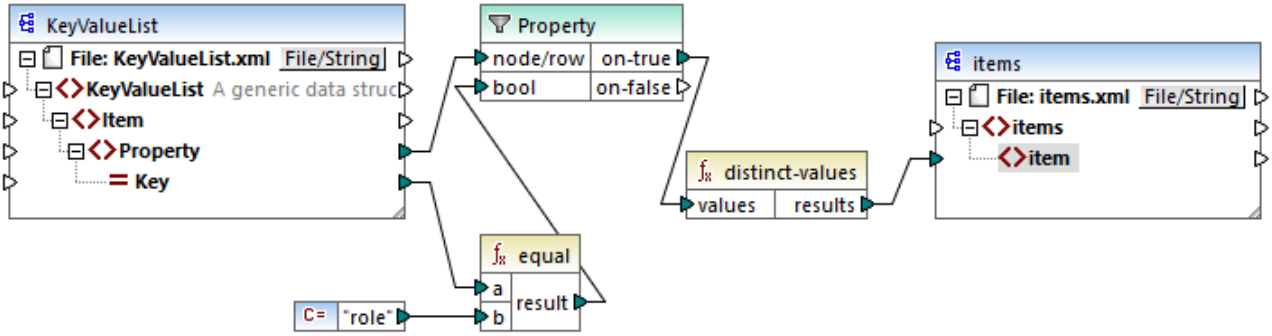
Beispiel

Die folgende XML-Datei enthält Informationen über Mitarbeiter einer Demo-Firma. Einige Mitarbeiter haben dieselbe Rolle (role), daher enthält das Attribut "role" doppelt vorhandene Werte. So haben etwa sowohl "Loby Matise" als auch "Susi Sanna" die Rolle "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>
```

Angenommen, Sie möchten eine Liste aller *eindeutigen* Rollennamen, die in dieser XML-Datei vorkommen, extrahieren. Dies lässt sich mit einem Mapping wie dem unten gezeigten, ermitteln:



Im oben gezeigten Mapping geschieht Folgendes:

- Jedes **Property**-Element aus der XML-Quelldatei wird durch einen Filter verarbeitet.
- Mit Hilfe der Verbindung mit dem Input **bool** des Filters wird sichergestellt, dass an die Zielkomponente nur **Property**-Elemente, deren **Key**-Attribut gleich "role" ist, übergeben werden. Der String "role" wird von einer Konstanten bereitgestellt. Beachten Sie, dass die Ausgabe des Filters an diesem Punkt immer noch Duplikate enthält (da es zwei "Support"-Eigenschaften gibt, die den Filterbedingungen entsprechen).
- Die vom Filter erzeuge Sequenz wird von der `distinct-values`-Funktion verarbeitet, die etwaige doppelt vorhandenen Werte ausschließt.

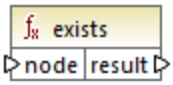
Infolgedessen erhalten wir die folgende Mapping-Ausgabe (ausschließlich der XML- und Schema-Deklaration):

```
<items>
   <item>Manager</item>
```

```
    <item>Programmer</item>
    <item>Support</item>
</items>
```

5.7.6.9.2 exists

Gibt **true** zurück, wenn der verbundene Node vorhanden ist und **false**, wenn dies nicht der Fall ist. Da ein Boolescher Wert zurückgegeben wird, wird diese Funktion normalerweise mit Filtern 194 verwendet, um nur Datensätze herauszufiltern, die ein Child-Element oder Attribut haben (oder eventuell nicht haben).



Sprachen

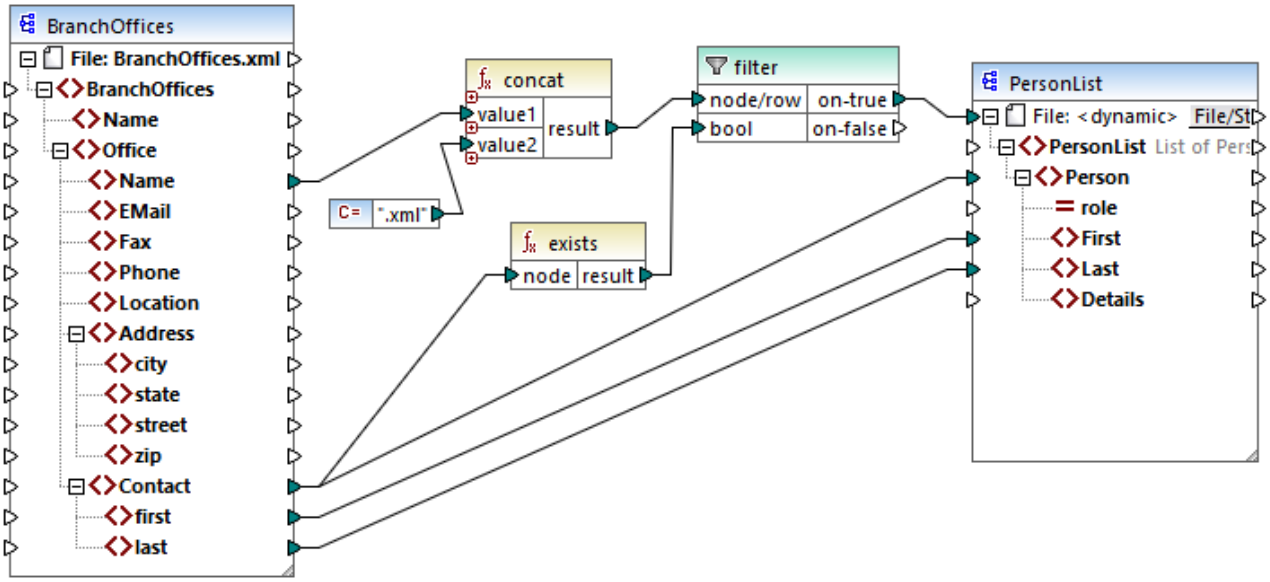
Built-in, C++, C#, Java, XQuery, XSLT 1.0, XSLT 2.0, XSLT 3.0..

Parameter

Name	Beschreibung
node	Der Node, dessen Vorhandensein überprüft werden soll.

Beispiele

Im folgenden Mapping wird gezeigt, wie Sie Daten mit Hilfe einer `exists`-Funktion filtern. Dieses Mapping hat den Namen **PersonListsForAllBranchOffices.mfd** und befindet sich im Verzeichnis **<Dokumente>\Altova\MapForce2025\MapForceExamples**.



PersonListsForAllBranchOffices.mfd

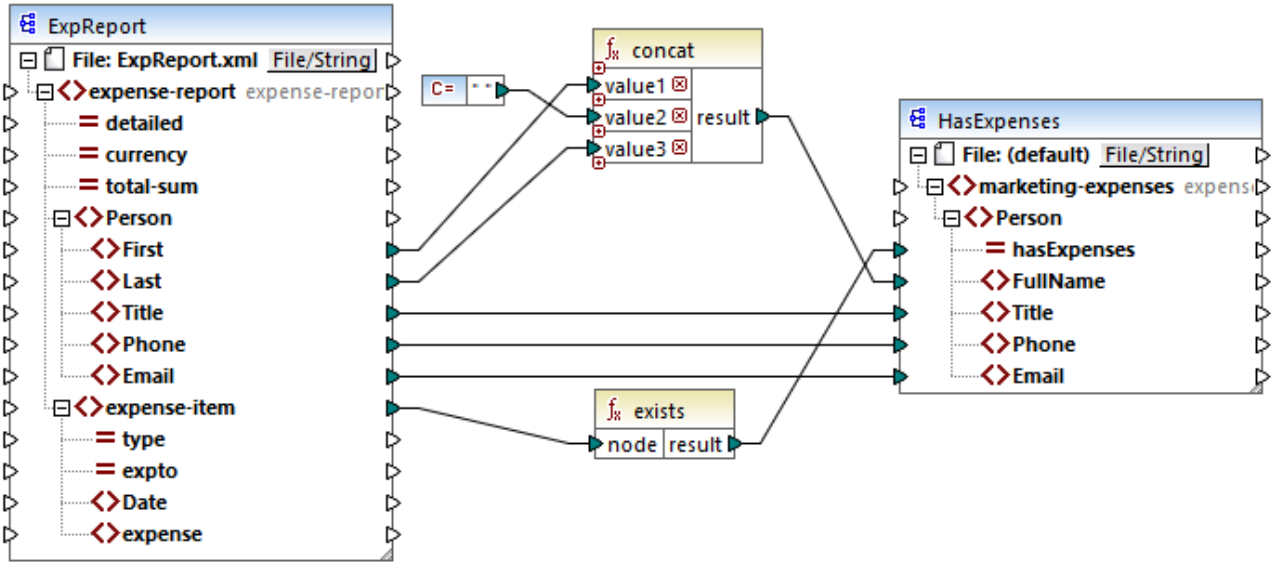
In der Quelldatei **BranchOffices.xml** gibt es drei **Office**-Elemente. Eines der Büros (office-Elemente) hat kein **Contact** Child-Element. Dieses Mapping hat mehrere Aufgaben:

a) Extraktion einer Liste der in jedem einzelnen Büro vorhandenen Kontakte.
b) Erstellung einer separaten XML-Datei für jedes Büro, wobei diese Datei denselben Namen wie das Büro hat.
c) Wenn das Büro keine Kontakte enthält, soll keine XML-Datei generiert werden.

Zu diesem Zweck wurde ein Filter zum Mapping hinzugefügt. Der Filter übergibt nur diejenigen **Office**-Datenelemente an die Zielkomponente, die mindestens ein **Contact**-Datenelement enthalten. Diese Boolesche Bedingung wird durch die `exists`-Funktion bereitgestellt. Wenn das Ergebnis der Funktion "true" ist, wird der Name des Büros (office) mit dem String `.xml` verkettet, um den Zieldateinamen zu erzeugen. Nähere Informationen zum Generieren von Dateinamen anhand des Mappings finden Sie unter Dynamische Verarbeitung mehrerer Input- oder Output-Dateien 422.

Ein weiteres Beispiel ist das folgende Mapping:
<Dokumente>\Altova\MapForce2025\MapForceExamples\HasMarketingExpenses.mfd. Wenn hier in der XML-Quelldatei ein Datenelement **expense-item** existiert, wird das Attribut **hasExpenses** in der XML-Zieldatei auf **true** gesetzt.

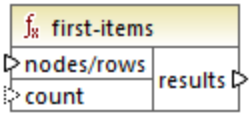
 © 2019-2025 Altova GmbH



HasMarketingExpenses.mfd

5.7.6.9.3 first-items

Gibt die ersten *N* Datenelemente der Input-Sequenz zurück, wobei *N* die vom Parameter **count** bereitgestellte Anzahl ist.



Sprachen

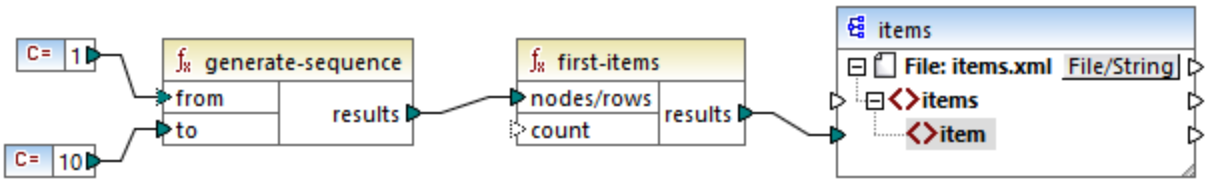
Built-in, C++, C#, Java, XQuery, XSLT 2.0, XSLT 3.0..

Parameter

Name	Beschreibung
nodes/rows	Mit diesem Input muss ein Mapping-Datenelement verbunden werden, das eine Sequenz 78 von null oder mehr Werten liefert. So kann damit etwa ein Datenelement aus einer XML-Quelldatei verbunden werden.
count	Optionaler Parameter. Definiert, wie viele Datenelemente aus der Input-Sequenz abgerufen werden sollen. Der Standardwert ist **1**.

Beispiel

Im folgenden Modell-Mapping wird eine Sequenz von 10 Werten generiert. Die Sequenz wird von der Funktion **first-items** verarbeitet und das Ergebnis wird in eine XML-Zieldatei geschrieben.



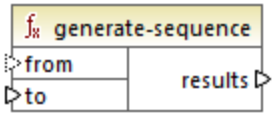
Da das Argument **count** keinen Wert hat, wird der Standardwert **1** angewendet. Infolgedessen wird nur der erste Wert aus der Sequenz in der Mapping-Ausgabe generiert:

```
<items>
   <item>1</item>
</items>
```

Ein realistisches Beispiel dazu finden Sie in dem in Bereitstellen von Parametern für das Mapping 164 beschriebenen Mapping **FindHighestTemperatures.mfd**.

5.7.6.9.4 generate-sequence

Erstellt eine Ganzzahlsequenz unter Verwendung der Parameter "from" und "to", um den Bereich einzugrenzen.



Sprachen

Built-in, C++, C#, Java, XQuery, XSLT 2.0, XSLT 3.0..

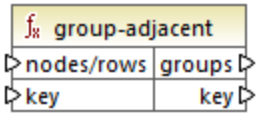
Parameter

Name	Beschreibung
from	Optionaler Parameter. Definiert die Ganzzahl, mit der die Sequenz beginnen soll (untere Grenze). Der Standardwert ist **1**.
to	Obligatorischer Parameter. Definiert die Ganzzahl, mit der die Sequenz enden soll (obere Grenze).

© 2019-2025 Altova GmbH

5.7.6.9.5 group-adjacent

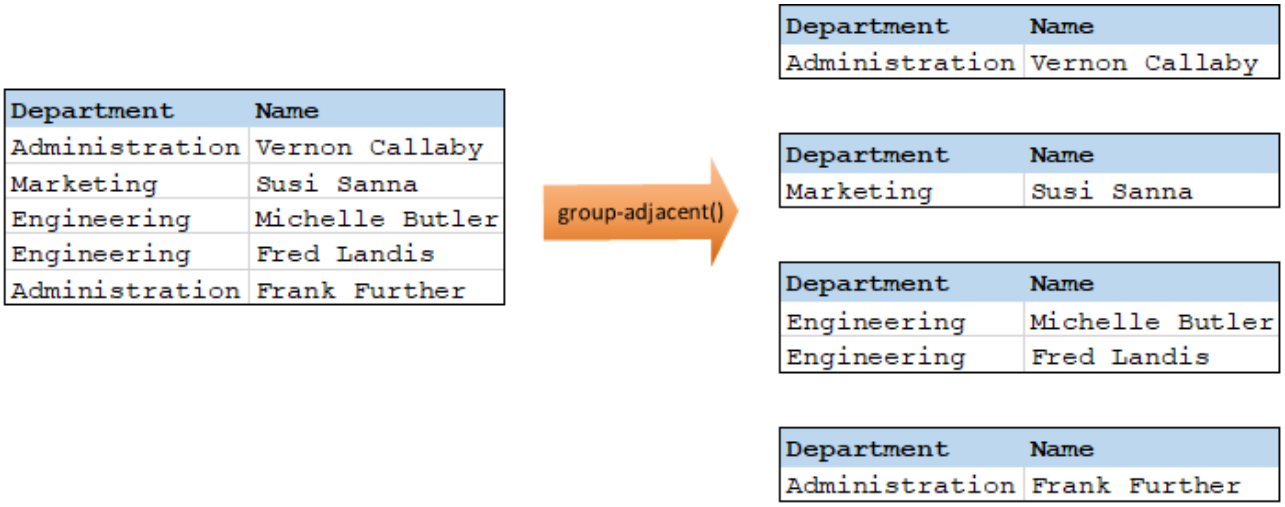
Die Funktion **`group-adjacent`** gruppiert die mit dem Input **nodes/rows** verbundenen Datenelemente nach dem mit dem **key**-Input verbundenen Schlüssel. Beachten Sie, dass Datenelemente mit demselben Schlüssel in separate Gruppen platziert werden, wenn sie nicht benachbart sind. Wenn mehrere aufeinander folgende (benachbarte) Datenelemente denselben Schlüssel haben, werden sie in dieselbe Gruppe platziert.



So ist etwa in der unten gezeigten abstrakten Transformation der Gruppierungsschlüssel "Department". Auf der linken Seite des Diagramms sehen Sie die Input-Daten, während rechts die Ausgabedaten nach der Gruppierung angezeigt werden. Bei Ausführung der Transformation geschieht Folgendes:

- Anfangs wird anhand des ersten Schlüssels "Administration" eine neue Gruppe erstellt.
- Der nächste Schlüssel ist ein anderer, daher wird eine zweite Gruppe "Marketing" erstellt.
- Der dritte Schlüssel ist ebenfalls ein anderer, daher wird eine weitere Gruppe namens "Engineering" erstellt.
- Der vierte Schlüssel ist derselbe wie der dritte, daher wird dieser Datensatz in die bereits vorhandene Gruppe platziert.
- Der fünfte Schlüssel schließlich ist anders als der vierte, weshalb die letzte Gruppe erstellt wird.

Wie unten gezeigt, landen "Michelle Butler" und "Fred Landis" in derselben Gruppe, weil sie denselben Schlüssel aufweisen und nebeneinander liegen. "Vernon Callaby" und "Frank Further" hingegen befinden sich in separaten Gruppen, weil sie nicht benachbart sind, obwohl sie denselben Schlüssel aufweisen.



Sprachen

Built-in, C++, C#, Java, XSLT 2.0, XSLT 3.0..

Parameter

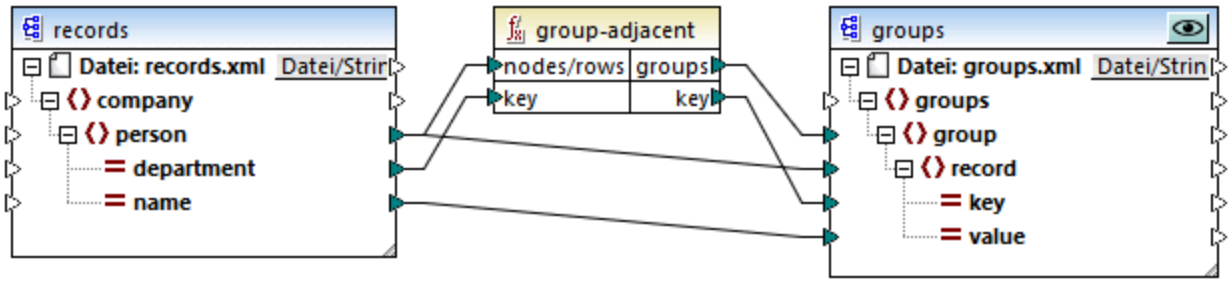
Name	Beschreibung
nodes/rows	Mit diesem Input muss ein Mapping-Datenelement verbunden werden, das eine Sequenz 78 von null oder mehr Werten liefert. So kann damit etwa ein Datenelement aus einer XML-Quelldatei verbunden werden.
key	Der Schlüssel, nach dem Datenelemente gruppiert werden sollen.

Beispiel

Angenommen, bei Ihren Quelldaten handelt es sich um eine XML-Datei folgenden Inhalts (Beachten Sie, dass der Namespace und die XML-Deklarationen im Codefragment unten aus Gründen der Einfachheit entfernt wurden).

```
<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>
```

Die Personendatensätze sollen nach Abteilung (department) gruppiert werden, vorausgesetzt die Datensätze sind benachbart. Zu diesem Zweck wird im folgenden Mapping die Funktion `group-adjacent` aufgerufen und **department** wird als **key** (Schlüssel) bereitgestellt.



Das Ergebnis des Mappings sieht folgendermaßen aus:

```
<groups>
   <group>
      <record key="Administration" value="Vernon Callaby"/>
   </group>
   <group>
      <record key="Marketing" value="Susi Sanna"/>
   </group>
   <group>
```

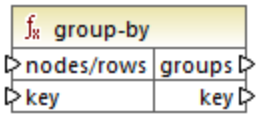
 © 2019-2025 Altova GmbH

```
      <record key="Engineering" value="Michelle Butler"/>
      <record key="Engineering" value="Fred Landis"/>
   </group>
   <group>
      <record key="Administration" value="Frank Further"/>
   </group>
</groups>
```

Sie finden dieses Beispiel sowie einige weitere Gruppierungsbeispiele in der folgenden Mapping-Datei: **<Dokumente>\Altova\MapForce2025\MapForceExamples\GroupingFunctions.mfd**. Klicken Sie auf die Schaltfläche **Vorschau** für die jeweilige Funktion, bevor Sie auf das Fenster **Ausgabe** klicken.

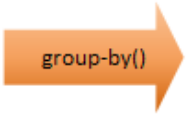
5.7.6.9.6 group-by

Mit der Funktion **group-by** werden anhand eines von Ihnen definierten Gruppierungsschlüssels Datensatzstrukturen erstellt.



So ist etwa in der unten gezeigten abstrakten Transformation der Gruppierungsschlüssel "Department". Da es insgesamt drei eindeutige Abteilungen (Departments) gibt, würden bei Anwendung der Funktion group-by drei Gruppen erstellt:

Department	Name
Administration	Vernon Callaby
Marketing	Susi Sanna
Engineering	Michelle Butler
Engineering	Fred Landis
Administration	Frank Further



Department	Name
Administration	Vernon Callaby
Administration	Frank Further

Department	Name
Marketing	Susi Sanna

Department	Name
Engineering	Michelle Butler
Engineering	Fred Landis

Sprachen

Built-in, C++, C#, Java, XSLT 2.0, XSLT 3.0.

Parameter

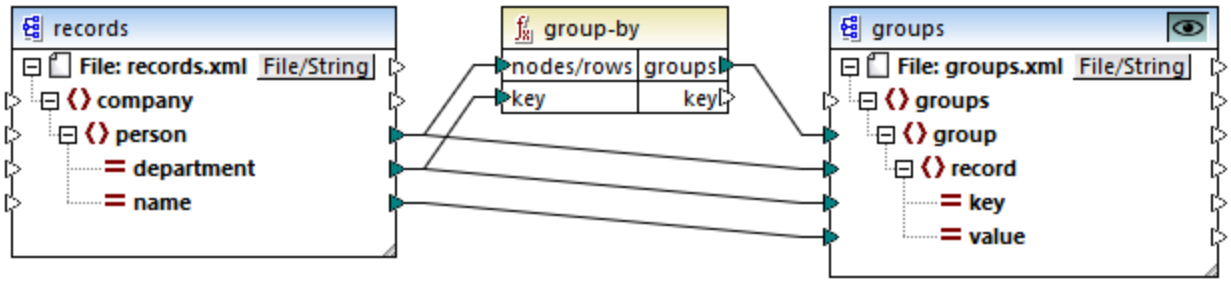
Name	Beschreibung
nodes/rows	Mit diesem Input muss ein Mapping-Datenelement verbunden werden, das eine Sequenz 78 von null oder mehr Werten liefert. So kann damit etwa ein Datenelement aus einer XML-Quelldatei verbunden werden.
key	Der Schlüssel, nach dem Datenelemente gruppiert werden sollen.

Beispiel 1

Angenommen, bei Ihren Quelldaten handelt es sich um eine XML-Datei folgenden Inhalts (Beachten Sie, dass der Namespace und die XML-Deklarationen im Codefragment unten aus Gründen der Einfachheit entfernt wurden).

```
<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>
```

Die Personendatensätze sollen nach Abteilung (department) gruppiert werden. Zu diesem Zweck wird im folgenden Mapping die Funktion `group-by` aufgerufen und **department** wird als key (Schlüssel) bereitgestellt.



Das Ergebnis des Mappings sieht folgendermaßen aus:

```
<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"/>
```

 © 2019-2025 Altova GmbH

```
      <record key="Engineering" value="Fred Landis"/>
   </group>
</groups>
```

Sie finden dieses Beispiel sowie einige weitere Gruppierungsbeispiele in der folgenden Mapping-Datei: **<Dokumente>\Altova\MapForce2025\MapForceExamples\GroupingFunctions.mfd**. Klicken Sie auf die Schaltfläche **Vorschau** für die jeweilige Funktion, bevor Sie auf das Fenster **Ausgabe** klicken.

Beispiel 2

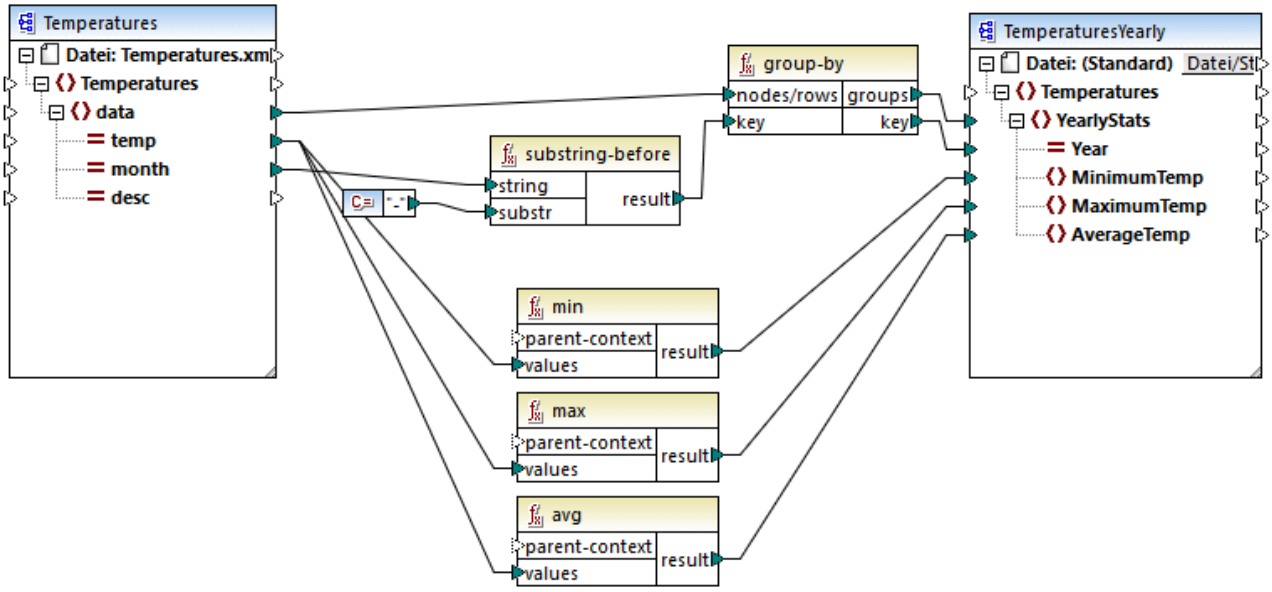
In diesem Beispiel wird gezeigt, wie Sie Datensätze mit Hilfe der `group-by`-Funktion gruppieren. Außerdem wird darin gezeigt, wie Sie Daten aggregieren. Das Demo-Mapping zu diesem Beispiel finden Sie unter dem folgenden Pfad:
<Dokumente>\Altova\MapForce2025\MapForceExamples\GroupTemperaturesByYear.mfd. In diesem Mapping werden Daten aus einer XML-Datei ausgelesen, die ein Protokoll monatlicher Temperaturen enthält, siehe Codefragment unten:

```
<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>
```

Dieses Mapping hat zwei Aufgaben:

1. Zusammengruppierung der Temperaturen jedes Jahres
2. Eruierung der jeweiligen Minima, Maxima und Durchschnittstemperaturen jedes Jahres

Für Aufgabe Nr. 1 wird die Funktion **group-by** im Mapping aufgerufen. Für Aufgabe Nr. 2 werden die Aggregierungsfunktionen **min** 257, **max** 256 und **avg** 254 aufgerufen.



GroupTemperaturesByYear.mfd

Bei der Ausführung eines MapForce-Mappings (dies ist auch die empfohlene Methode, um ein Mapping zu lesen), wird mit dem obersten Datenelement der Zielkomponente begonnen. In diesem Beispiel wird für jede von der Funktion **group-by** zurückgegebene Gruppe ein **YearlySales**-Datenelement zurückgegeben. Die Funktion **group-by** nimmt als erstes Argument alle **data**-Datenelemente aus der Quellkomponente und gruppiert diese nach dem Input, der mit dem Input **key** verbunden ist. Da die Temperaturen nach Jahr gruppiert werden sollen, muss zuerst das Jahr ermittelt werden. Zu diesem Zweck extrahiert die Funktion **substring-before** 334 das Jahr aus dem Attribut **month** der einzelnen **data**-Elemente. Dabei wird als Argument der Wert von **month** verwendet und es wird der Teil vor der ersten Instanz von **substr** zurückgegeben. Wie oben gezeigt, wird in diesem Beispiel für **substr** das Bindestrichzeichen definiert. Wenn die Funktion daher den Wert "2006-01" verarbeitet, ist das Ergebnis der Funktion "2006".

Die Werte von **MinimumTemp**, **MaximumTemp** und **AverageTemp** werden schließlich durch Verbinden dieser Datenelemente mit der jeweiligen Aggregationsfunktion **min**, **max** und **avg** ermittelt. Alle drei Funktionen erhalten als Input die aus der Quellkomponente ausgelesene Sequenz von Temperaturen. Für diese Funktionen wird kein **parent-context**-Argument benötigt, da sie bereits im Kontext der einzelnen Gruppen verwendet werden, d.h. es gibt eine übergeordnete Verbindung von **data** zu **YearlyStats,** die den Kontext für die einzelnen Aggrationsfunktionen liefert.

Klicken Sie auf das Register **Ausgabe**, um eine Vorschau auf das Mapping-Ergebnis zu sehen. Beachten Sie, dass die Anzahl der Gruppen der Anzahl der in der Quelldatei vorhandenen Jahre entspricht, z.B.:

```
<Temperatures>
   <YearlyStats Year="2006">
      <MinimumTemp>-3.6</MinimumTemp>
      <MaximumTemp>23.2</MaximumTemp>
      <AverageTemp>11.375</AverageTemp>
```

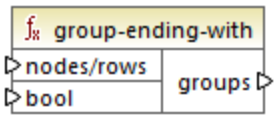
 © 2019-2025 Altova GmbH

```
    </YearlyStats>
    <YearlyStats Year="2007">
        <MinimumTemp>-3.2</MinimumTemp>
        <MaximumTemp>22.3</MaximumTemp>
        <AverageTemp>11.5</AverageTemp>
    </YearlyStats>
</Temperatures>
```

Anmerkung: Die oben gezeigten Codefragmente enthalten aus Gründen der Einfachheit weniger Daten als die tatsächliche Input- und Output-Datei aus dem Demo-Mapping.

5.7.6.9.7 group-ending-with

Die Funktion **group-ending-with** erhält eine Boolsche Bedingung als Argument. Wenn die Boolsche Bedingung "true" ergibt, wird bis inklusive dem Datensatz, auf den die Bedingung zutrifft, eine neue Gruppe erstellt.



Im Beispiel unten ist die Bedingung, dass "Key" (Schlüssel) gleich "trailing" sein soll. Diese Bedingung trifft auf den dritten und fünften Datensatz zu, daher werden als Ergebnis zwei Gruppen erstellt:

Key	Value
line	A
line	B
trailing	Total 1
line	C
trailing	Total 2

Key	Value
line	A
line	B
trailing	Total 1

Key	Value
line	C
trailing	Total 2

Anmerkung: Wenn nach dem letzten Datensatz, auf den die Bedingung zutrifft, weitere Datensätze vorhanden sind, wird eine zusätzliche Gruppe erstellt. Wenn z.B. nach dem letzten "trailing"-Datensatz z.B. weitere "line"-Datensätze vorhanden wären, würden diese in eine neue Gruppe platziert.

Sprachen

Built-in, C++, C#, Java, XSLT 2.0, XSLT 3.0..

Parameter

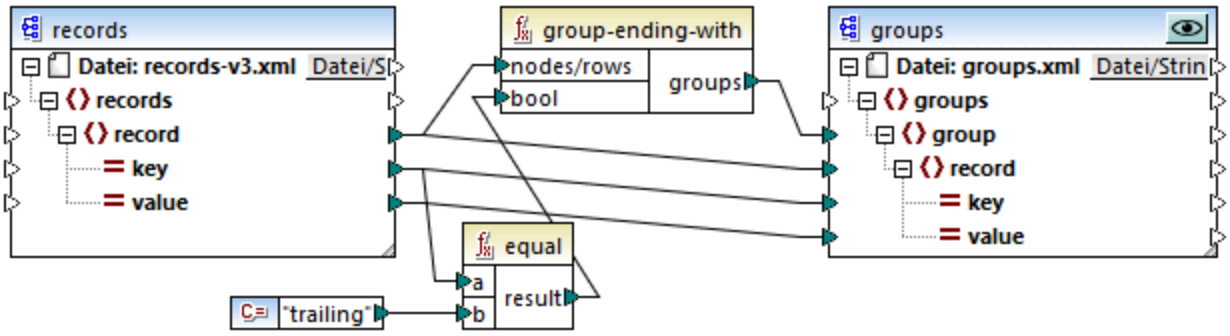
Name	Beschreibung
nodes/rows	Mit diesem Input muss ein Mapping-Datenelement verbunden werden, das eine Sequenz 78 von null oder mehr Werten liefert. So kann damit etwa ein Datenelement aus einer XML-Quelldatei verbunden werden.
bool	Liefert die Boolsche Bedingung, durch die bei **true** eine neue Gruppe begonnen werden soll.

Beispiel

Angenommen, bei Ihren Quelldaten handelt es sich um eine XML-Datei folgenden Inhalts (Beachten Sie, dass der Namespace und die XML-Deklarationen im Codefragment unten aus Gründen der Einfachheit entfernt wurden).

```
<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>
```

Die Aufgabe ist es, für jeden nachfolgenden ("trailing") Datensatz Gruppen zu erstellen. Außerdem muss jede Gruppe auch alle "line" (Zeilen)-Datensätze enthalten, die vor dem nachfolgenden Datensatz stehen. Zu diesem Zweck wird im folgenden Mapping die Funktion **group-ending-with** aufgerufen. Immer, wenn der **key**-Name im nachstehenden Mapping gleich "trailing" ist, wird das an **bool** gelieferte Argument **true** und es wird eine neue Gruppe erstellt.



Das Ergebnis des Mappings sieht folgendermaßen aus:

```
<groups>
   <group>
      <record key="line" value="A"/>
      <record key="line" value="B"/>
```

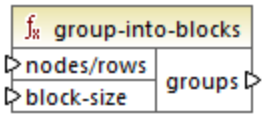
 © 2019-2025 Altova GmbH

```
      <record key="trailing" value="Total 1"/>
   </group>
   <group>
      <record key="line" value="C"/>
      <record key="trailing" value="Total 2"/>
   </group>
</groups>
```

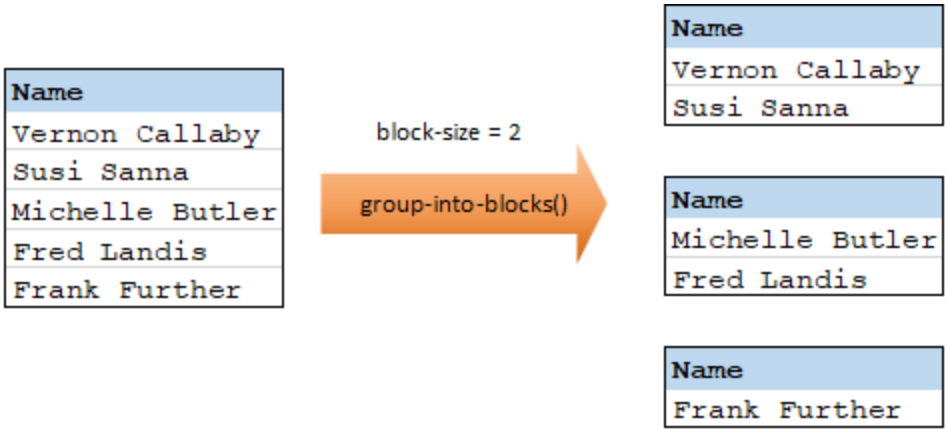
Sie finden dieses Beispiel sowie einige weitere Gruppierungsbeispiele in der folgenden Mapping-Datei: **<Dokumente>\Altova\MapForce2025\MapForceExamples\GroupingFunctions.mfd**. Klicken Sie auf die Schaltfläche **Vorschau** für die jeweilige Funktion, bevor Sie auf das Fenster **Ausgabe** klicken.

5.7.6.9.8 group-into-blocks

Mit der Funktion **`group-into-blocks`** werden gleiche Gruppen erstellt, die genau N Elemente enthalten, wobei N der an das Argument `block-size` (Blockgröße) gelieferte Wert ist. Beachten Sie, dass die letzte Gruppe, je nach Anzahl der Elemente in der Quellkomponente, N oder weniger Elemente enthalten kann.



Im Beispiel unten ist `block-size` gleich 2. Da es insgesamt fünf Elemente gibt, enthält jede Gruppe mit Ausnahme der letzten genau zwei Elemente.



Sprachen

Built-in, C++, C#, Java, XSLT 2.0, XSLT 3.0..

Parameter

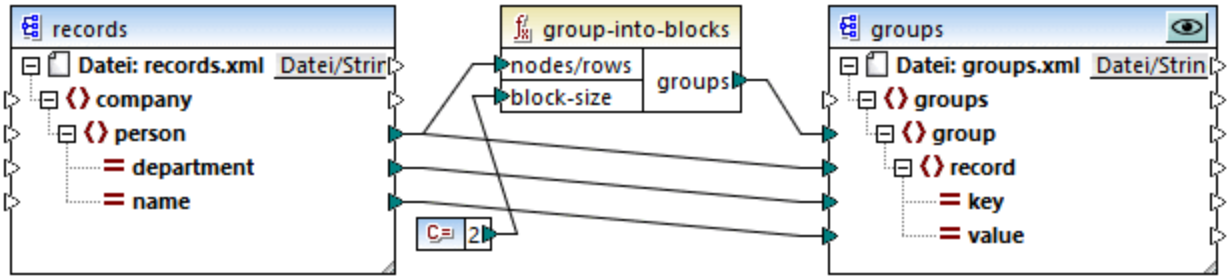
Name	Beschreibung
nodes/rows	Mit diesem Input muss ein Mapping-Datenelement verbunden werden, das eine Sequenz 78 von null oder mehr Werten liefert. So kann damit etwa ein Datenelement aus einer XML-Quelldatei verbunden werden.
block-size	Definiert die Größe der einzelnen Gruppen.

Beispiel

Angenommen, bei Ihren Quelldaten handelt es sich um eine XML-Datei folgenden Inhalts (Beachten Sie, dass der Namespace und die XML-Deklarationen im Codefragment unten aus Gründen der Einfachheit entfernt wurden).

```
<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>
```

Die Personendatensätze sollen in Blöcke zu jeweils zwei Datenelementen gruppiert werden. Zu diesem Zweck wird im folgenden Mapping die Funktion **group-into-blocks** aufgerufen und es wird der Ganzzahlwert "2" als **block-size** bereitgestellt.



Das Ergebnis des Mappings sieht folgendermaßen aus:

```
<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"/>
```

 © 2019-2025 Altova GmbH

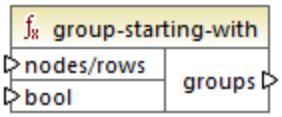
```
    </group>
    <group>
        <record key="Administration" value="Frank Further"/>
    </group>
</groups>
```

Beachten Sie, dass die letzte Gruppe nur ein Datenelement enthält, da die Summe aller Datenelemente (5) nicht gerade durch 2 dividiert werden kann.

Sie finden dieses Beispiel sowie einige weitere Gruppierungsbeispiele in der folgenden Mapping-Datei: **<Dokumente>\Altova\MapForce2025\MapForceExamples\GroupingFunctions.mfd**. Klicken Sie auf die Schaltfläche **Vorschau** für die jeweilige Funktion, bevor Sie auf das Fenster **Ausgabe** klicken.

5.7.6.9.9 group-starting-with

Die Funktion **group-starting-with** erhält eine Boolsche Bedingung als Argument. Wenn die Boolsche Bedingung "true" ergibt, wird ab dem Datensatz, auf den die Bedingung zutrifft, eine neue Gruppe erstellt.



Im Beispiel unten ist die Bedingung, dass "Key" (Schlüssel) gleich "heading" sein soll. Diese Bedingung trifft auf den ersten und vierten Datensatz zu, daher werden als Ergebnis zwei Gruppen erstellt:

Anmerkung: Wenn vor dem ersten Datensatz, auf den die Bedingung zutrifft, weitere Datensätze vorhanden sind, wird eine zusätzliche Gruppe erstellt. Wenn vor dem ersten "heading"-Datensatz z.B. weitere "line"-Datensätze vorhanden waren, würden diese in eine neue Gruppe platziert.

Sprachen

Built-in, C++, C#, Java, XSLT 2.0, XSLT 3.0..

Parameter

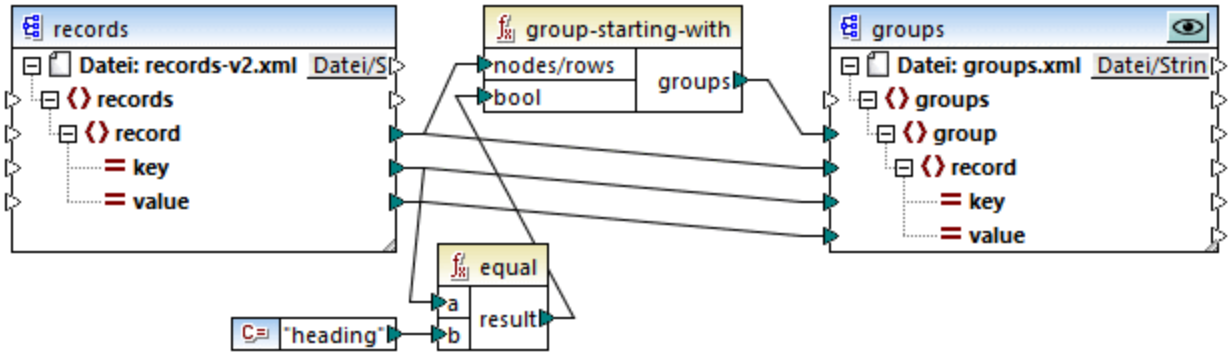
Name	Beschreibung
nodes/rows	Mit diesem Input muss ein Mapping-Datenelement verbunden werden, das eine Sequenz 78 von null oder mehr Werten liefert. So kann damit etwa ein Datenelement aus einer XML-Quelldatei verbunden werden.
bool	Liefert die Boolsche Bedingung, durch die bei **true** eine neue Gruppe begonnen werden soll.

Beispiel

Angenommen, bei Ihren Quelldaten handelt es sich um eine XML-Datei folgenden Inhalts (Beachten Sie, dass der Namespace und die XML-Deklarationen im Codefragment unten aus Gründen der Einfachheit entfernt wurden).

```
<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>
```

Die Aufgabe ist es, für jeden vorangestellten ("heading") Datensatz Gruppen zu erstellen. Außerdem muss jede Gruppe auch alle "line" (Zeilen)-Datensätze enthalten, die auf den "heading"-Datensatz folgen. Zu diesem Zweck wird im folgenden Mapping die Funktion `group-starting-with` aufgerufen. Immer, wenn der **key**-Name im nachstehenden Mapping gleich "heading" ist, wird das an **bool** gelieferte Argument **true** und es wird eine neue Gruppe erstellt.



Das Ergebnis des Mappings sieht folgendermaßen aus:

```
<groups>
   <group>
      <record key="heading" value="Intro"/>
```

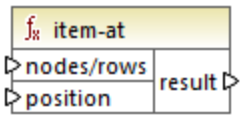
 © 2019-2025 Altova GmbH

```
      <record key="line" value="A"/>
      <record key="line" value="B"/>
   </group>
   <group>
      <record key="heading" value="Body"/>
      <record key="line" value="C"/>
   </group>
</groups>
```

Sie finden dieses Beispiel sowie einige weitere Gruppierungsbeispiele in der folgenden Mapping-Datei: **<Dokumente>\Altova\MapForce2025\MapForceExamples\GroupingFunctions.mfd**. Klicken Sie auf die Schaltfläche **Vorschau** für die jeweilige Funktion, bevor Sie auf das Fenster **Ausgabe** klicken.

5.7.6.9.10 item-at

Gibt ein Datenelement aus einer als Argument bereitgestellten Sequenz von **nodes/rows** (Nodes/Zeilen) zurück, das sich an der durch das Argument **position** angegebenen Position befindet. Das erste Datenelement befindet sich an der Position **1**.



Sprachen

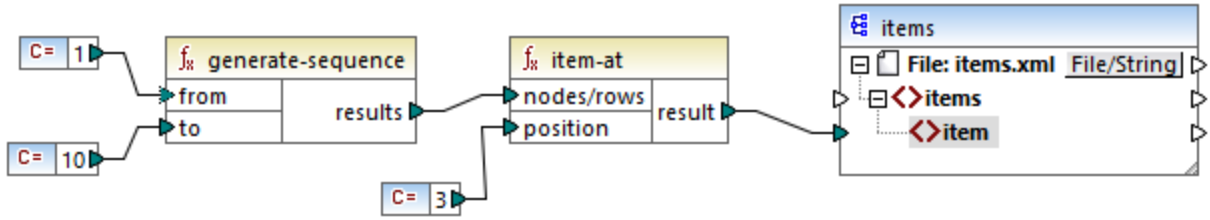
Built-in, C++, C#, Java, XQuery, XSLT 2.0, XSLT 3.0..

Parameter

Name	Beschreibung
nodes/rows	Mit diesem Input muss ein Mapping-Datenelement verbunden werden, das eine Sequenz 78 von null oder mehr Werten liefert. So kann damit etwa ein Datenelement aus einer XML-Quelldatei verbunden werden.
position	Diese Ganzzahl gibt an, welches Datenelement aus der Sequenz von Datenelementen zurückgegeben werden soll.

Beispiel

Im folgenden Modell-Mapping wird eine Sequenz von 10 Werten generiert. Die Sequenz wird von der Funktion **item-at** verarbeitet und das Ergebnis wird in eine XML-Zieldatei geschrieben.

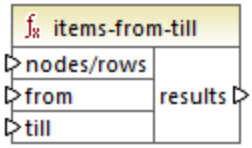


Da das **position**-Argument auf **3** gesetzt wurde, wird nur der dritte Wert aus der Sequenz an die Zielkomponente übergeben. Infolgedessen erhalten wir die folgende Mapping-Ausgabe (ausschließlich der XML- und Schema-Deklaration):

```
<items>
   <item>3</item>
</items>
```

5.7.6.9.11 items-from-till

Gibt eine Sequenz von **nodes/rows** (Nodes/Zeilen) unter Verwendung der Parameter "from" und "till" zur Eingrenzung des Bereichs zurück. Das erste Datenelement befindet sich an der Position **1**.



Sprachen

Built-in, C++, C#, Java, XQuery, XSLT 2.0, XSLT 3.0..

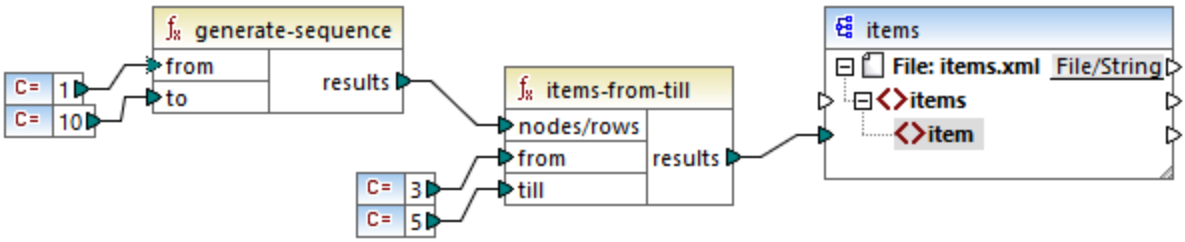
Parameter

Name	Beschreibung
nodes/rows	Mit diesem Input muss ein Mapping-Datenelement verbunden werden, das eine Sequenz 78 von null oder mehr Werten liefert. So kann damit etwa ein Datenelement aus einer XML-Quelldatei verbunden werden.
from	Diese Ganzzahl gibt die Startposition, ab welcher Datenelemente abgerufen werden sollen, an.
till	Diese Ganzzahl gibt die Position, bis zu welcher Datenelemente abgerufen werden sollen, an.

© 2019-2025 Altova GmbH

Beispiel

Im folgenden Modell-Mapping wird eine Sequenz von 10 Werten generiert. Die Sequenz wird von der Funktion **`items-from-till`** verarbeitet und das Ergebnis wird in eine XML-Zieldatei geschrieben.

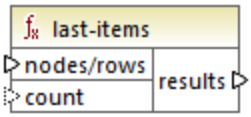


Da die Argumente **from** und **till** auf **3** bzw. **5** gesetzt wurden, wird nur die Untergruppe der Werte von **3** bis **5** an die Zielkomponente übergeben. Infolgedessen erhalten wir die folgende Mapping-Ausgabe (ausschließlich der XML- und Schema-Deklaration):

```
<items>
   <item>3</item>
   <item>4</item>
   <item>5</item>
</items>
```

5.7.6.9.12 last-items

Gibt die letzten *N* Datenelemente der Input-Sequenz zurück, wobei *N* die vom Parameter **count** bereitgestellte Anzahl ist. Das erste Datenelement befindet sich an der Position "1".



Sprachen

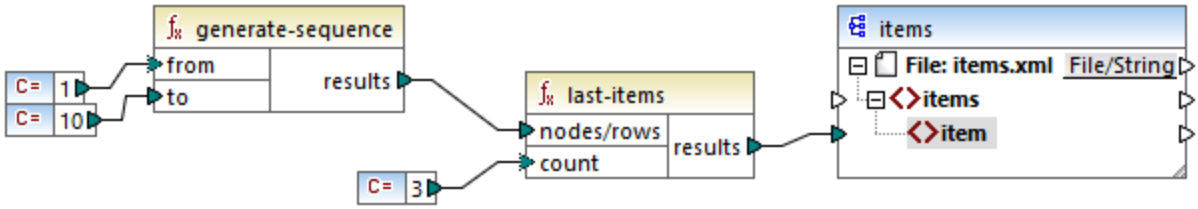
Built-in, C++, C#, Java, XQuery, XSLT 2.0, XSLT 3.0..

Parameter

Name	Beschreibung
nodes/rows	Mit diesem Input muss ein Mapping-Datenelement verbunden werden, das eine Sequenz [78] von null oder mehr Werten liefert. So kann damit etwa ein Datenelement aus einer XML-Quelldatei verbunden werden.
count	Optionaler Parameter. Definiert, wie viele Datenelemente aus der Input-Sequenz abgerufen werden sollen. Der Standardwert ist **1**.

Beispiel

Im folgenden Modell-Mapping wird eine Sequenz von 10 Werten generiert. Die Sequenz wird von der Funktion **last-items** verarbeitet und das Ergebnis wird in eine XML-Zieldatei geschrieben.

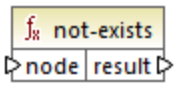


Da das count-Argument auf **3** gesetzt wurde, werden nur die drei letzten Werte aus der Sequenz an die Zielkomponente übergeben. Infolgedessen erhalten wir die folgende Mapping-Ausgabe (ausschließlich der XML- und Schema-Deklaration):

```
<items>
   <item>8</item>
   <item>9</item>
   <item>10</item>
</items>
```

5.7.6.9.13 not-exists

Gibt **false** zurück, wenn der verbundene Node vorhanden ist und **true**, wenn dies nicht der Fall ist. Diese Funktion ist das Gegenteil der Funktion exists [299] und wird ansonsten auf dieselbe Art verwendet.



Sprachen

Built-in, C++, C#, Java, XQuery, XSLT 1.0, XSLT 2.0, XSLT 3.0..

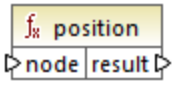
Parameter

Name	Beschreibung
node	Der Node, dessen Fehlen überprüft werden soll.

5.7.6.9.14 position

Gibt die Position eines Datenelements innerhalb der gerade verarbeiteten Sequenz von Datenelementen zurück. Damit können Datenelemente z.B. automatisch sequenziell nummeriert werden.

 © 2019-2025 Altova GmbH



Sprachen

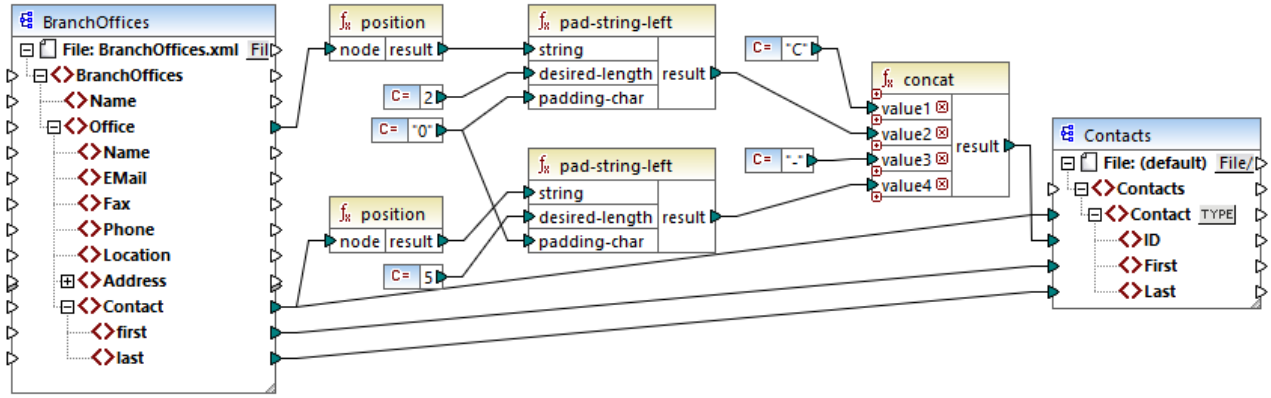
Built-in, C++, C#, Java, XQuery, XSLT 1.0, XSLT 2.0, XSLT 3.0..

Parameter

Name	Beschreibung
node	Mit diesem Input muss ein Mapping-Datenelement verbunden werden, das eine Sequenz 78 von null oder mehr Werten liefert. So kann damit etwa ein Datenelement aus einer XML-Quelldatei verbunden werden.

Beispiel

Im folgenden Mapping wird gezeigt, wie Sie mit Hilfe der `position`-Funktion in den vom Mapping generierten Daten eindeutige Identifikationswerte generieren. Die Mapping-Design-Datei zu diesem Beispiel finden Sie unter dem folgenden Pfad:
<Dokumente>\Altova\MapForce2025\MapForceExamples\ContactsFromBranchOffices.mfd.



ContactsFromBranchOffices.mfd

Die XML-Quelldatei im obigen Mapping enthält drei Niederlassungen (branch offices). Eine Niederlassung kann beliebig viele **Contact** Child-Elemente enthalten. Ziel des Mappings ist folgendes:

- Extrahierung aller **Contact**-Datenelemente aus der XML-Quelldatei und Schreiben der Daten in eine XML-Zieldatei.
- Jedem Kontakt muss eine eindeutige Idenifikationsnummer zugewiesen werden (das Datenelement **ID** in der XML-Zielkomponente).
- Die ID jedes Kontakts muss die Form `CXX-YYYYY` haben, wobei X die Nummer der Niederlassung (Office) und Y die Nummer des Kontakts (Contact) ist. Wenn die Niederlassungsnummer weniger als zwei Zeichen hat, müssen ihr links Nullen vorangestellt werden. Wenn die Kontaktnummer weniger als

fünf Zeichen hat, müssen ihr links ebenfalls Nullen vorangestellt werden. Eine gültige Identifikationsnummer des ersten Kontakts aus der ersten Niederlassung wäre folglich **C01-00001**.

Dieses Ziel wurde mit Hilfe einer Reihe von MapForce-Funktionen, darunter der Funktion **position**, erreicht. Die obere **position**-Funktion ruft die Position der einzelnen office-Elemente ab. Die untere ruft die Position der einzelnen Kontakte im Kontext der einzelnen office-Elemente ab.

Bei Verwendung der **position**-Funktion muss der aktuelle Mapping-Kontext 79 berücksichtigt werden. Das heißt, bei Ausführung des Mappings wird der Anfangs-Mapping-Kontext anhand des Root-Elements der Zielkomponente zu dem damit (auch indirekt über Funktionen) verbundenen Quelldatenelement ermittelt. In diesem Beispiel verarbeitet die obere **position**-Funktion die *Sequenz aller Niederlassungen (office)* und generiert anfangs für die erste Niederlassung in der Sequenz den Wert 1. Die untere **position**-Funktion generiert eine sequenzielle Nummerierung für die Position des Kontakts (Contact) *im Kontext dieses office-Elements* (1, 2, 3, usw.). Beachten Sie, dass diese "innere" Sequenz zurückgesetzt wird (und daher wieder mit 1 beginnt), wenn das nächste office-Element verarbeitet wird. Beide **pad-string-left**-Funktionen wenden gemäß den zuvor angeführten Anforderungen Füllzeichen auf die generierten Nummern an. Die `concat`-Funktion wird (aufgrund der übergeordneten Verbindung vom **Contact**-Element in der Quellkomponente zum **Contact**-Element in der Zielkomponente) im *Kontext der einzelnen Kontakte* angewendet. Sie verbindet alle berechneten Werte und gibt die eindeutige Identifikationsnummer der einzelnen Kontakte zurück.

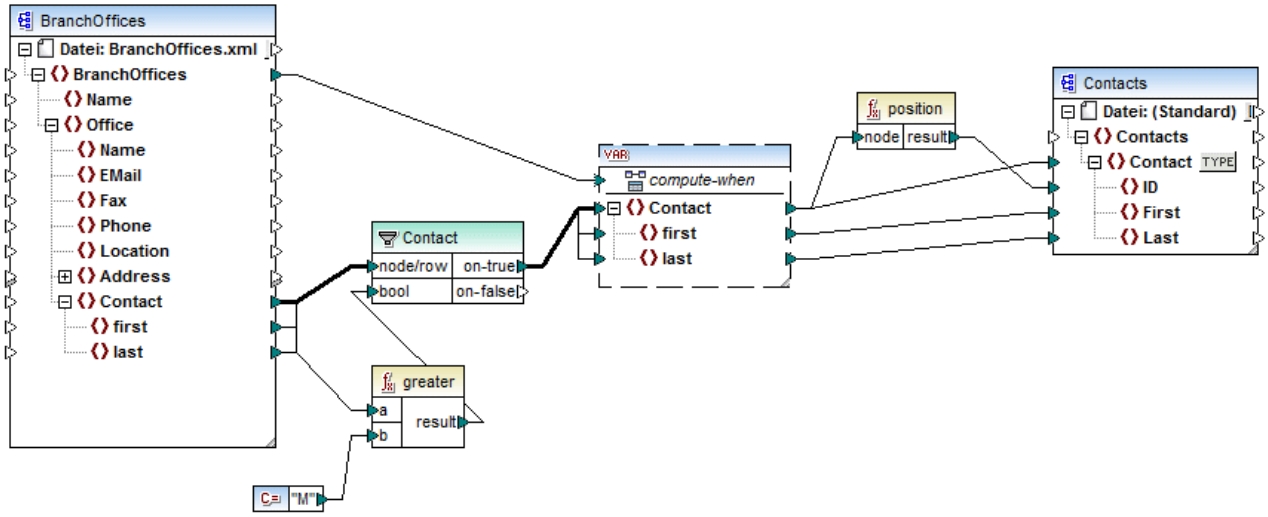
Unten sehen Sie das Ergebnis, das vom oben gezeigten Mapping generiert wird (beachten Sie, dass einige der Datensätze aus Gründen der Übersichtlichkeit entfernt wurden):

```
<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>
```

In einigen Fällen müssen Sie eventuell die Position von Datenelementen nach Anwendung eines Filters 194 ermitteln. Beachten Sie, dass es sich bei der Filterkomponente nicht um eine Sequenz-Funktion handelt. Sie kann nicht *direkt* zusammen mit der **position**-Funktion verwendet werden, um die Position gefilterter

 © 2019-2025 Altova GmbH

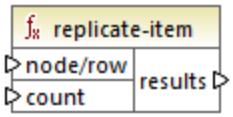
Datenelemente zu finden. Dies kann indirekt durch Hinzufügen einer Variablen 175-Komponente zum Mapping ermittelt werden. Das unten gezeigte Mapping ist eine vereinfachte Version des vorherigen Mappings. Die verwendete Mapping-Design-Datei finden Sie unter dem folgenden Pfad: **<Dokumente>\Altova\MapForce2025\MapForceExamples\PositionInFilteredSequence.mfd**.



Das Ergebnis von Variablen in MapForce sind immer Sequenzen. Daher iteriert die `position`-Funktion im obigen Mapping durch die durch die Variable erstellte Sequenz und gibt die Position in den einzelnen Datenelementen in dieser Sequenz zurück. Dieses Mapping wird unter Beispiel: Filtern und Nummerieren von Nodes 184 ausführlicher erläutert.

5.7.6.9.15 replicate-item

Repliziert alle Datenelement in der Input-Sequenz so oft, wie im Argument **count** definiert. Wenn Sie ein einziges Datenelement mit dem **node/row** Input verbinden, gibt die Funktion *N* Datenelemente zurück, wobei *N* der Wert des Arguments **count** ist. Wenn Sie eine Sequenz von Datenelementen mit dem **node/row** Input verbinden, repliziert die Funktion jedes einzelne Datenelement in der Sequenz **count** Mal, wobei die Datenelemente der Reihe nach verarbeitet werden. Wenn z.B. count **2** ist, so erzeugt die Sequenz `1,2,3` das Ergebnis `1,1,2,2,3,3`. Es kann auch für jedes Datenelement in der Input-Sequenz ein anderer **count**-Wert angegeben werden, wie im Beispiel unten gezeigt.



Sprachen

Built-in, C++, C#, Java, XQuery, XSLT 2.0, XSLT 3.0..

Parameter

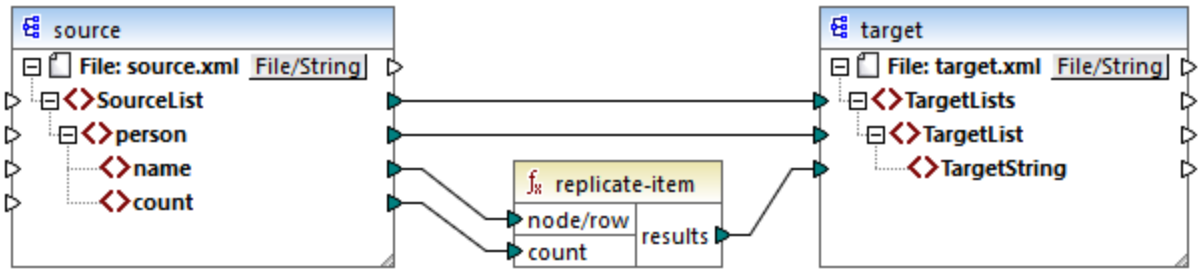
Name	Beschreibung
node/row	Mit diesem Input muss ein Mapping-Datenelement verbunden werden, das eine Sequenz [78] von null oder mehr Werten liefert. So kann damit etwa ein Datenelement aus einer XML-Quelldatei verbunden werden.
count	Definiert, wie oft jedes Datenelement oder jede mit **node/row** verbundene Sequenz repliziert werden soll.

Beispiel

Angenommen, Sie haben eine XML-Quelldatei mit der folgenden Struktur:

```
<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>
```

Mit Hilfe der Funktion `replicate-item` können Sie jeden Personennamen unterschiedlich oft in der Zielkomponente replizieren. Verbinden Sie dazu den Node **count** der einzelnen person-Elemente mit dem **count**-Input der Funktion `replicate-item`:



Das Ergebnis ist das folgende:

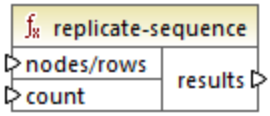
```
<TargetLists>
   <TargetList>
      <TargetString>Michelle</TargetString>
```

 © 2019-2025 Altova GmbH

```
      <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>
```

5.7.6.9.16 replicate-sequence

Repliziert alle Datenelement in der Input-Sequenz so oft, wie im Argument **count** definiert. Wenn z.B. count **2** ist, so erzeugt die Sequenz `1,2,3` das Ergebnis `1,2,3,1,2,3`.



Sprachen

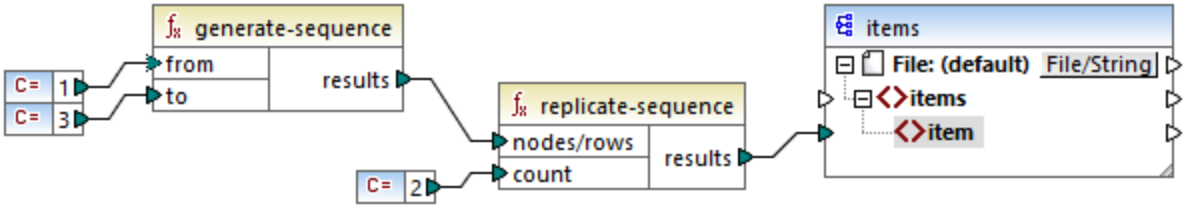
Built-in, C++, C#, Java, XQuery, XSLT 2.0, XSLT 3.0..

Parameter

Name	Beschreibung
node-rows	Mit diesem Input muss ein Mapping-Datenelement verbunden werden, das eine Sequenz [78] von null oder mehr Werten liefert. So kann damit etwa ein Datenelement aus einer XML-Quelldatei verbunden werden.
count	Definiert, wie oft die verbundene Sequenz repliziert werden soll.

Beispiel

Im folgenden Modell-Mapping wird die Sequenz `1,2,3` generiert. Die Sequenz wird von der Funktion `replicate-sequence` verarbeitet und das Ergebnis wird in eine XML-Zieldatei geschrieben.

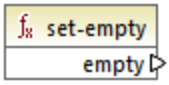


Da das **count**-Argument auf **2** gesetzt wurde, wird die Sequenz zwei Mal repliziert und an die Zielkomponente übergeben. Infolgedessen erhalten wir die folgende Mapping-Ausgabe (ausschließlich der XML- und Schema-Deklaration):

```
<items>
   <item>1</item>
   <item>2</item>
   <item>3</item>
   <item>1</item>
   <item>2</item>
   <item>3</item>
</items>
```

5.7.6.9.17 set-empty

Gibt eine leere Sequenz zurück.

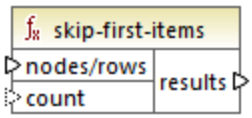


Sprachen

Built-in, C++, C#, Java, XQuery, XSLT 2.0, XSLT 3.0..

5.7.6.9.18 skip-first-items

Überspringt die ersten N Datenelemente der Input-Sequenz, wobei N durch das **count** Argument angegeben wird, und gibt den Rest der Sequenz zurück.



Sprachen

Built-in, C++, C#, Java, XQuery, XSLT 2.0, XSLT 3.0..

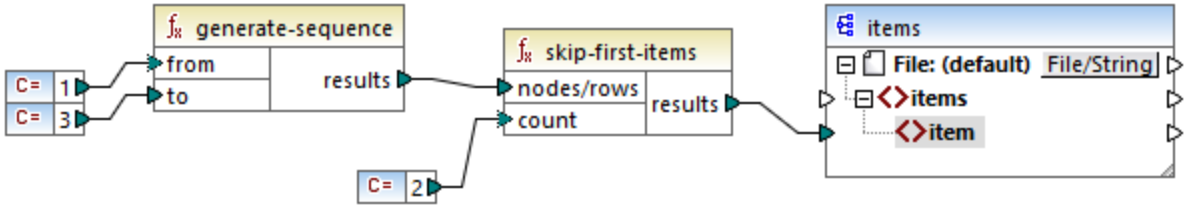
© 2019-2025 Altova GmbH

Parameter

Name	Beschreibung
node-rows	Mit diesem Input muss ein Mapping-Datenelement verbunden werden, das eine Sequenz 78 von null oder mehr Werten liefert. So kann damit etwa ein Datenelement aus einer XML-Quelldatei verbunden werden.
count	Optionales Argument. Definiert die Anzahl der Datenelemente, die übersprungen werden sollen. Der Standardwert ist **1**.

Beispiel

Im folgenden Modell-Mapping wird die Sequenz `1,2,3` generiert. Die Sequenz wird von der Funktion `skip-first-items` verarbeitet und das Ergebnis wird in eine XML-Zieldatei geschrieben.

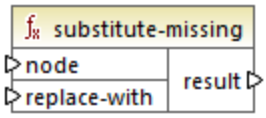


Da das **count**-Argument auf **2** gesetzt wurde, werden die ersten zwei Datenelemente übersprungen und die restlichen Datenelementen an die Zielkomponente übergeben. Infolgedessen erhalten wir die folgende Mapping-Ausgabe (ausschließlich der XML- und Schema-Deklaration):

```
<items>
   <item>3</item>
</items>
```

5.7.6.9.19 substitute-missing

Diese Funktion ist eine praktische Kombination aus der Funktion exists 299 und einer "if-else"-Bedingung 198. Wenn das mit dem **node** Input verbundene Datenelement vorhanden ist, wird dessen Inhalt in die Zielkomponente kopiert. Andernfalls wird der Inhalt des mit dem **replace-with** Input verbundenen Datenelements in die Zielkomponente kopiert.



Sprachen

Built-in, C++, C#, Java, XQuery, XSLT 1.0, XSLT 2.0, XSLT 3.0..

Parameter

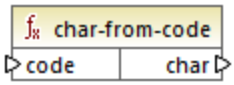
Name	Beschreibung
node	Mit diesem Input muss ein Mapping-Datenelement verbunden werden, das eine Sequenz 78 von null oder mehr Werten liefert. So kann damit etwa ein Datenelement aus einer XML-Quelldatei verbunden werden.
replace-with	Mit diesem Input muss ein Mapping-Datenelement verbunden werden, das den Ersetzungswert bereitstellt.

5.7.6.10 core | string functions (String-Funktionen)

Mit Hilfe von String-Funktionen können Sie String-Daten bearbeiten, um Abschnitte von Strings zu extrahieren, den String auf darin enthaltene Strings zu überprüfen oder Informationen über Strings abzurufen, Strings zu trennen usw.

5.7.6.10.1 char-from-code

Gibt die Zeichendarstellung des als Argument angegebenen Unicode-Dezimalwerts (Code) zurück. **Tipp:** Sie können den Unicode-Dezimalcode eines Zeichens mit Hilfe der Funktion **`code-from-char`** 328 finden.



Sprachen

Built-in, C++, C#, Java, XQuery, XSLT 2.0, XSLT 3.0.

Parameter

Name	Beschreibung
Code	Der Unicode-Wert als Dezimalzahl.

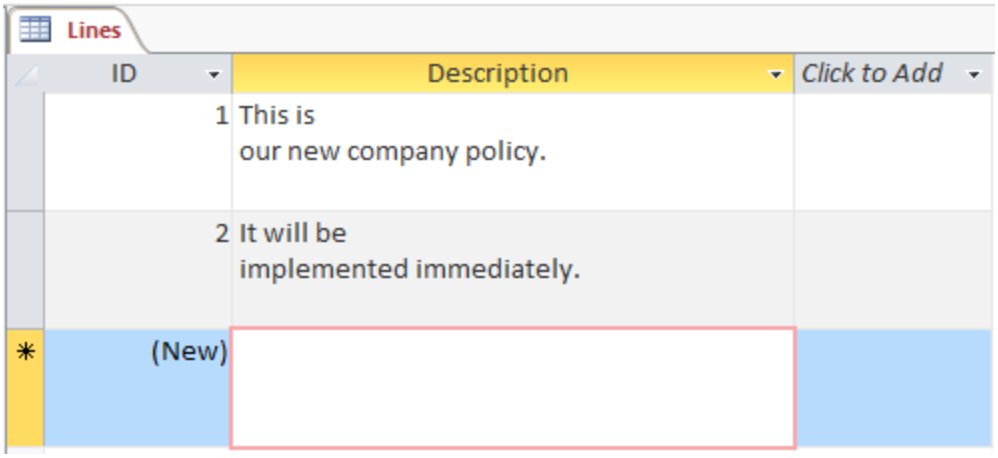
Beispiel 1

Laut den Tabellen auf der Unicode-Website (https://www.unicode.org/charts/) hat das Ausrufezeichen den Hexadezimalwert **`0021`**. Der entsprechende Wert im Dezimalformat ist **33**. Wenn Sie daher **`33`** als Argument für die Funktion **`char-from-code`** bereitstellen, erhalten Sie das Zeichen **`!`** .

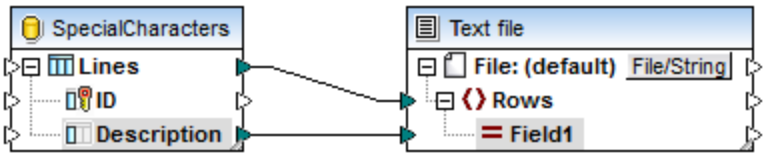
© 2019-2025 Altova GmbH

Beispiel 2 (Professional und Enterprise Edition)

In diesem Beispiel wird gezeigt, wie Sie in einer Datenbank Sonderzeichen durch Leerzeichen ersetzen. Angenommen, Sie haben eine SQLite-Datenbank bestehend aus einer Tabelle "Lines", die zwei Spalten enthält: "ID" und "Description".



Ziel ist es, die einzelnen Beschreibungen in eine CSV-Datei zu extrahieren (eine Beschreibung pro Zeile); ein Mapping, womit Sie dies erreichen, könnte folgendermaßen aussehen:

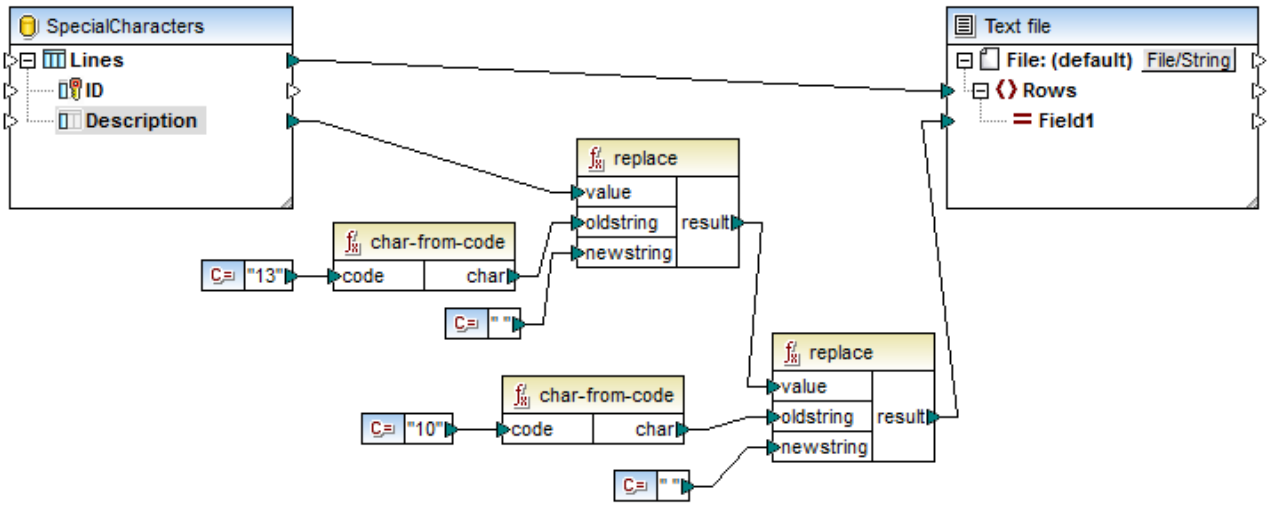


Da jedoch jede "Description"-Zeile in Access mehrere durch CR/LF-Zeichen getrennte Zeichen enthält, enthält auch das Mapping Zeilenumbrüche, was nicht erwünscht ist:

```
1   "This is
2   our new company policy."
3   "It will be
4   implemented immediately."
5
```

Um dieses Problem zu lösen, werden wir die Funktionen **`char-from-code`** und **`replace`** aus der Bibliothek der vordefinierten MapForce-Funktionen zum Mapping hinzufügen. Jede Beschreibung muss verarbeitet werden, sodass die oben genannten Zeichen durch ein Leerzeichen ersetzt werden.

In der Unicode-Tabelle (http://www.unicode.org/charts/) entsprechen die Zeichen LF und CR den Hexadezimalzeichen **hex 0A | dec 10** bzw. **hex 0D | dec 13**. Das Mapping muss daher geändert werden, um die Unicode-Dezimalwerte 13 und 10 in einen String zu konvertieren, damit die Daten mit Hilfe der **`replace`** Funktion weiter bearbeitet werden können.



Bei Anzeige einer Vorschau auf das Mapping sehen Sie jetzt, dass die Zeichen CR/LF in den einzelnen Datenbankfeldern durch Leerzeichen ersetzt wurden.

```
1   This is  our new company policy.
2   It will be  implemented immediately.
3
```

5.7.6.10.2 code-from-char

Gibt den Unicode-Dezimalwert (Code) des als Argument bereitgestellten Zeichens zurück. Wenn der als Argument bereitgestellte String mehrere Zeichen hat, wird der Code des ersten Zeichens zurückgegeben.

Sprachen

Built-in, C++, C#, Java, XQuery, XSLT 2.0, XSLT 3.0..

Parameter

Name	Beschreibung
char	Der Input-String-Wert.

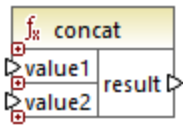
© 2019-2025 Altova GmbH

Beispiel

Wenn der Input **char** das Zeichen **$** (Dollarzeichen) ist, gibt die Funktion **36** zurück (Dies ist der Unicode-Dezimalwert dieses Zeichens).

5.7.6.10.3 concat

Verkettet zwei oder mehr Werte zu einem einzigen Ergebnisstring. Alle Input-Werte werden automatisch in den Typ "string" konvertiert. Standardmäßig hat diese Funktion nur zwei Parameter, Sie können jedoch weitere hinzufügen. Klicken Sie auf **Parameter hinzufügen** (⊞) oder **Parameter löschen** (⊠), um Parameter hinzuzufügen oder zu löschen.



Anmerkung: Alle Inputs für die `concat`-Funktion müssen einen Wert haben. Wenn einer der Inputs keinen Wert hat, wird die Funktion nicht aufgerufen und es tritt ein Fehler auf. Bedenken Sie, dass auch ein leerer String ein gültiger Input-Wert ist; eine leere Sequenz (wie z.B. das Ergebnis der Funktion **`set-empty`**) ist hingegen kein gültiger Wert, weswegen die Funktion als Ergebnis fehlschlägt. Um dies zu vermeiden, können Sie Werte zuerst mit der Funktion substitute-missing 325 verarbeiten und das Ergebnis anschließend als Input für die **`concat`**-Funktion bereitstellen.

Sprachen

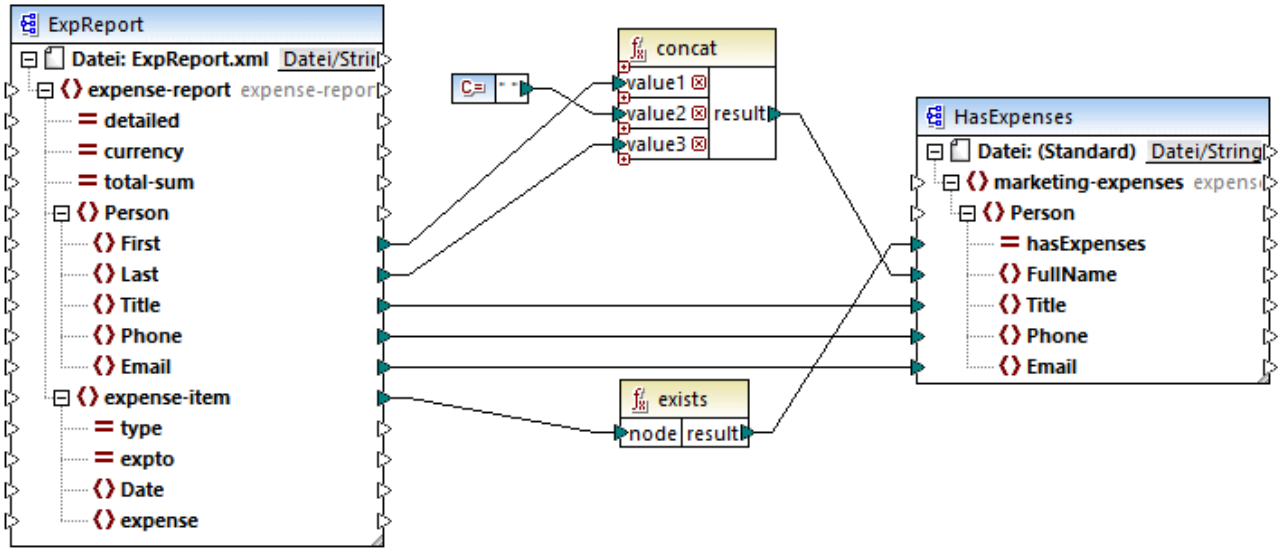
Built-in, C++, C#, Java, XQuery, XSLT 1.0, XSLT 2.0, XSLT 3.0..

Parameter

Name	Beschreibung
value1	Der erste Input-Wert.
value2	Der zweite Input-Wert.
valueN	Der Input-Wert *N*.

Beispiel

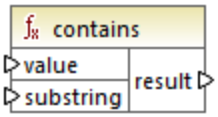
Die **`concat`**-Funktion im unten gezeigten Mapping verbindet den Vornamen, die Konstante " " und den Nachnamen miteinander. Der Ergebniswert wird anschließend in das Zieldatenelement **FullName** geschrieben. Sie finden das Mapping dieser Funktion unter dem folgenden Pfad: **<Dokumente>\Altova\MapForce2025\MapForceExamples\HasMarketingExpenses.mfd**.



HasMarketingExpenses.mfd

5.7.6.10.4 contains

Gibt den Booleschen Wert **true** zurück, wenn das als String-Wert bereitgestellte Argument den als Argument bereitgestellten Substring enthält.



Sprachen

Built-in, C++, C#, Java, XQuery, XSLT 1.0, XSLT 2.0, XSLT 3.0..

Parameter

Name	Beschreibung
value	Der Input-Wert (d.h. der "Heuhaufen").
substring	Der Substring, nach dem gesucht wird (d.h. die "Nadel").

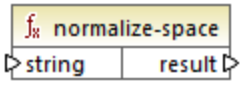
Beispiel

Wenn der Input **value** "category" und der Substring **substring** "cat" ist, gibt die Funktion **true** zurück.

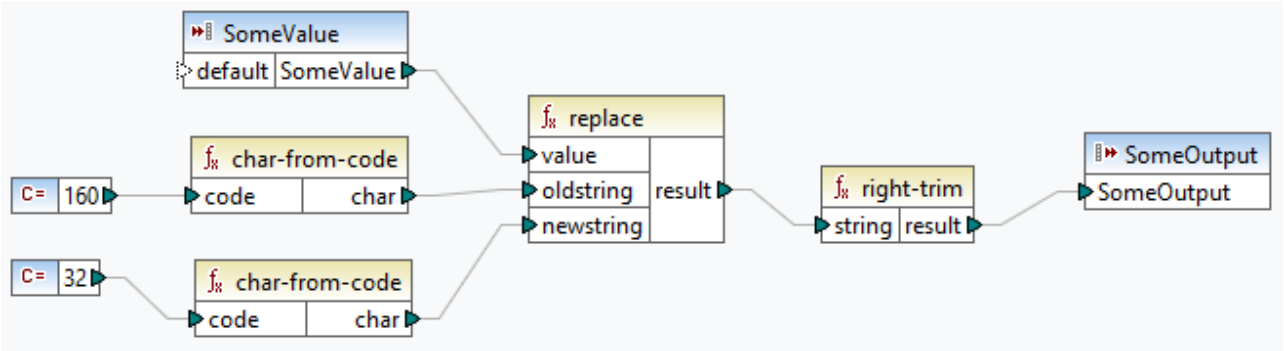
 © 2019-2025 Altova GmbH

5.7.6.10.5 normalize-space

Die Funktion **`normalize-space`** function (*siehe Abbildung unten*) entfernt vorangestellte und nachgestellte Leerzeichen aus einem String und ersetzt interne Whitespace-Zeichen durch ein einziges Whitespace-Zeichen. Zu den Whitespace-Zeichen zählen das Leerzeichen (U+0020), der Tabulator (U+0009), der Wagenrücklauf (U+000D) und der Zeilenvorschub (U+000A). Nähere Informationen zu Whitespaces finden Sie in der XML Recommendation.



Informationen zu geschützten Leerzeichen
Die Funktionen **`left-trim`**, **`right-trim`** und **`normalize-space`** entfernen geschützte Leerzeichen nicht. Eine mögliche Lösung wäre, ein geschütztes Leerzeichen, dessen Dezimaldarstellung 160 ist, durch ein Leerzeichen mit der Dezimaldarstellung 32 zu ersetzen. Im nächsten Schritt wird der gekürzte Wert `Wert` auf das Zieldatenelement gemappt (*siehe Mapping unten*).



Wenn es sich bei Ihrer Quellkomponente um eine Excel-Datei handelt, können Sie überschüssige Leerzeichen in Excel mit Hilfe einer Kombination aus den Funktionen TRIM, CLEAN und SUBSTITUTE entfernen. Nähere Informationen dazu finden Sie unter *Entfernen von Leerzeichen und nicht druckbaren Zeichen aus dem Text*.

Sprachen

Built-in, C++, C#, Java, XQuery, XSLT 1.0, XSLT 2.0, XSLT 3.0.

Parameter

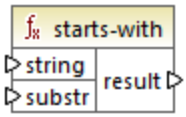
Name	Beschreibung
String	Der zu normalisierende Input-String.

Beispiel

Wenn der Input String `The quick  brown fox` ist, gibt die Funktion `The quick brown fox` zurück.

5.7.6.10.6 starts-with

Gibt den Booleschen Wert **true** zurück, wenn das als String bereitgestellte Argument mit dem als Argument bereitgestellten Substring beginnt; gibt andernfalls **false** zurück.



Sprachen

Built-in, C++, C#, Java, XQuery, XSLT 1.0, XSLT 2.0, XSLT 3.0..

Parameter

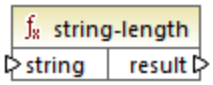
Name	Beschreibung
string	Der Input-String.
substr	Der Substring, auf den der Wert überprüft werden soll.

Beispiel

Wenn der Input **string** `category` und **substr** `cat` ist, gibt die Funktion **true** zurück.

5.7.6.10.7 string-length

Gibt die Anzahl der Zeichen in dem als Argument angegebenen String zurück.



Sprachen

Built-in, C++, C#, Java, XQuery, XSLT 1.0, XSLT 2.0, XSLT 3.0..

Parameter

Name	Beschreibung
string	Der Input-String.

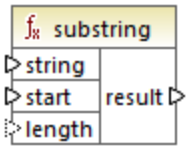
© 2019-2025 Altova GmbH

Beispiel

Wenn der Input-String `car` ist, gibt die Funktion **3** zurück. Wenn der Input-String ein leerer String ist, gibt die Funktion **0** zurück.

5.7.6.10.8 substring

Gibt den Anteil des durch die Parameter **start** und **length** definierten Strings zurück.



Sprachen

Built-in, C++, C#, Java, XQuery, XSLT 1.0, XSLT 2.0, XSLT 3.0..

Parameter

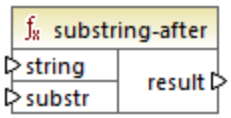
Name	Beschreibung
string	Der Input-String.
start	Definiert die Anfangsposition (Index), ab der der Substring abgerufen werden soll. Der erste Index ist **1**.
length	Optional. Definiert die Anzahl der abzurufenden Zeichen. Wenn der Parameter **length** nicht angegeben wird, ist das Ergebnis ein Fragment ab der Position **start** bis zum Ende des Strings.

Beispiel

Wenn der Input-String `MapForce` ist, "start" **1** und length **3** ist, gibt die Funktion `Map` zurück. Wenn der Input-String `MapForce`, "start" **4** ist und "length" nicht angegeben wird, gibt die Funktion `Force` zurück.

5.7.6.10.9 substring-after

Gibt den Teil des String zurück, der hinter der ersten Instanz von **substr** steht. Wenn **substr** in **string** nicht vorkommt, gibt die Funktion einen leeren String zurück.



Sprachen

Built-in, C++, C#, Java, XQuery, XSLT 1.0, XSLT 2.0, XSLT 3.0..

Parameter

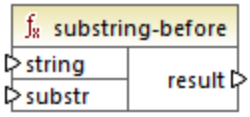
Name	Beschreibung
string	Der Input-String.
substr	Der Substring. Das Ergebnis dieser Funktion sind alle Zeichen nach der ersten Instanz von **substr**.

Beispiel

Wenn der Input-String `MapForce` und **substr** `Map` ist, gibt die Funktion `Force` zurück. Wenn der Input-String **2020/01/04** und `substr` `/` ist, gibt die Funktion `01/04` zurück.

5.7.6.10.10 substring-before

Gibt den Teil des String zurück, der vor der ersten Instanz von **substr** steht. Wenn **substr** in **string** nicht vorkommt, gibt die Funktion einen leeren String zurück.



Sprachen

Built-in, C++, C#, Java, XQuery, XSLT 1.0, XSLT 2.0, XSLT 3.0..

Parameter

Name	Beschreibung
string	Der Input-String.
substr	Der Substring. Das Ergebnis dieser Funktion sind alle Zeichen vor der ersten Instanz von **substr**.

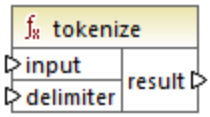
Beispiel

Wenn der Input-String `MapForce` und **substr** `Force` ist, gibt die Funktion `Map` zurück. Wenn der Input-String **2020/01/04** und `substr` `/` ist, gibt die Funktion `2020` zurück.

© 2019-2025 Altova GmbH

5.7.6.10.11 tokenize

Teilt den Input-String mit Hilfe des als Argument angegebenen Trennzeichens in eine Sequenz von Strings auf.



Sprachen

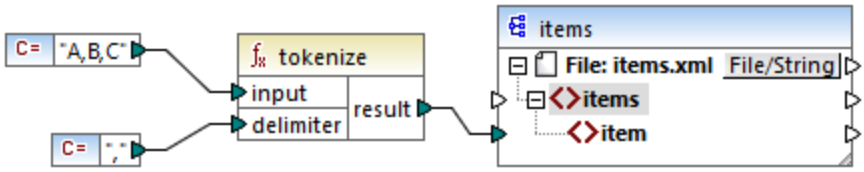
Built-in, C++, C#, Java, XQuery, XSLT 2.0, XSLT 3.0..

Parameter

Name	Beschreibung
input	Der Input-String.
delimiter	Das zu verwendende Trennzeichen.

Beispiel

Wenn der Input-String `A,B,C` und "delimiter" `,` ist, gibt die Funktion eine Sequenz von drei Strings zurück: `A`, `B` und `C`.

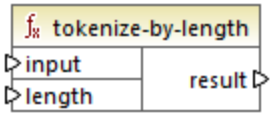


Im oben gezeigten Modell-Mapping ist das Ergebnis der Funktion eine Sequenz von Strings. Entsprechend den allgemeinen Mappingregeln 78 wird für jedes Datenelement in der Quellsequenz ein neues Datenelement **item** in der Zielkomponente erstellt. Die Mapping-Ausgabe sieht daher folgendermaßen aus:

```
<items>
   <item>A</item>
   <item>B</item>
   <item>C</item>
</items>
```

5.7.6.10.12 tokenize-by-length

Teilt den Input-String in eine Sequenz von Strings auf. Die Größe der einzelnen erzeugten Strings wird durch den Parameter **length** definiert.



Sprachen

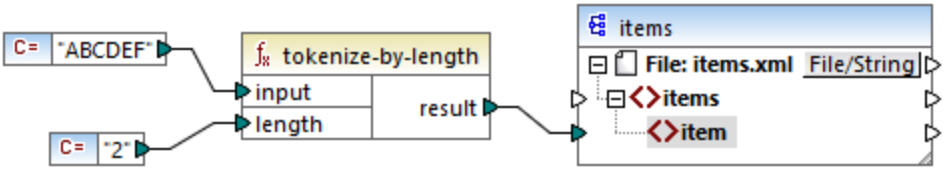
Built-in, C++, C#, Java, XQuery, XSLT 2.0, XSLT 3.0..

Parameter

Name	Beschreibung
input	Der Input-String.
length	Definiert die Länge der einzelnen Strings in der generierten Stringsequenz.

Beispiel

Wenn der Input-String **ABCDEF** und "length " **2** ist, gibt die Funktion eine Sequenz von drei Strings zurück: **AB**, **CD** und **EF**.



Im oben gezeigten Modell-Mapping ist das Ergebnis der Funktion eine Sequenz von Strings. Entsprechend den allgemeinen Mappingregeln 78 wird für jedes Datenelement in der Quellsequenz ein neues Datenelement **item** in der Zielkomponente erstellt. Die Mapping-Ausgabe sieht daher folgendermaßen aus:

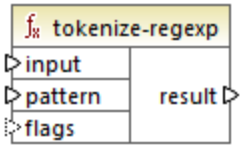
```
<items>
   <item>AB</item>
   <item>CD</item>
   <item>EF</item>
</items>
```

5.7.6.10.13 tokenize-regexp

Teilt den Input-String in eine Sequenz von Strings auf. Als Trennzeichen wird ein beliebiger Substring definiert, der mit dem **pattern** (Muster) der als Argument bereitgestellten Regular Expression übereinstimmt. Die gefundenen Trennzeichen-Strings werden nicht in das durch die Funktion zurückgegebene Ergebnis inkludiert.

Anmerkung: Die komplexen Funktionalitäten der Regular Expression-Syntax können bei der Generierung von C++-, C#- oder Java-Code etwas unterschiedlich sein. Nähere Informationen dazu finden Sie in der regex-Dokumentation zu den einzelnen Sprachen.

 © 2019-2025 Altova GmbH



Sprachen

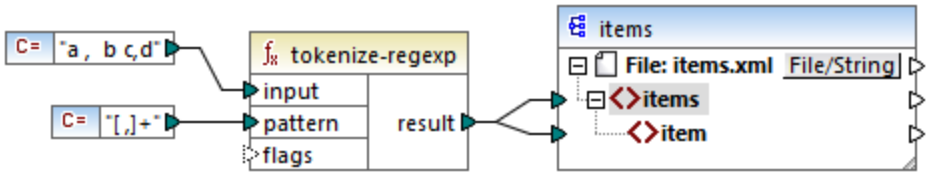
Built-in, C++, C#, Java, XQuery, XSLT 2.0, XSLT 3.0..

Parameter

Name	Beschreibung
input	Der Input-String.
pattern	Gibt ein Muster für eine Regular Expression an. Jeder beliebige Substring, der mit dem Muster übereinstimmt, wird als Trennzeichen verwendet. Nähere Informationen dazu finden Sie unter Regular Expressions 247.
flags	Optionaler Parameter. Stellt die Flags 249 bereit, die für die Regular Expression verwendet werden sollen. So verarbeitet das Mapping etwa den Input aufgrund des Flags "i" ohne Berücksichtigung der Groß- und Kleinschreibung.

Beispiel

Ziel des unten gezeigten Mappings ist es, den String `a ,  b c,d` in eine Stringsequenz aufzuteilen, wobei jedes alphabetische Zeichen ein Element in der Sequenz bildet. Überzählige Whitespace-Zeichen oder Kommas müssen entfernt werden.



Zu diesem Zweck wurde als Parameter für die `tokenize-regexp` Funktion das Ausdrucksmuster `[ ,]+` bereitgestellt. Dieses Muster hat folgende Bedeutung:

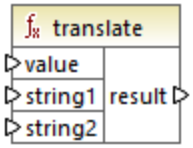
- Es steht für jedes der Zeichen innerhalb der Zeichenklasse `[ ,]`. Daher wird der Input-String überall dort, wo ein Komma oder Leerzeichen steht, aufgeteilt.
- Der Quantifizierer `+` gibt an, dass eine oder mehrere Instanzen des vorangestellten Zeichens übereinstimmen müssen. Ohne diesen Quantifizierer würde für jede Instanz eines Leerzeichens oder Kommas ein separates Datenelement in der Stringsequenz erzeugt, was nicht das Ziel der Aufgabe ist.

Das Ergebnis des Mappings sieht folgendermaßen aus:

```
<items>
   <item>a</item>
   <item>b</item>
   <item>c</item>
   <item>d</item>
</items>
```

5.7.6.10.14 translate

Führt eine Zeichen-für-Zeichen-Ersetzung durch. Sucht in **value** nach in **string1** enthaltenen Zeichen und ersetzt jedes Zeichen durch das Zeichen an derselben Position in **string2**. Wenn es in **string2** keine entsprechenden Zeichen gibt, wird das Zeichen entfernt.



Sprachen

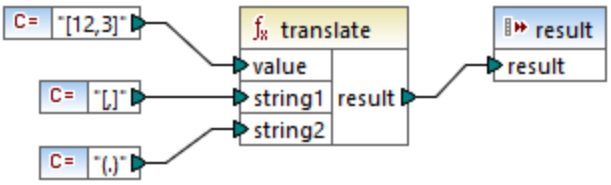
Built-in, C++, C#, Java, XQuery, XSLT 1.0, XSLT 2.0, XSLT 3.0..

Parameter

Name	Beschreibung
value	Der Input-String.
string1	Stellt eine Liste von zu suchenden Zeichen bereit. Die Position der einzelnen Zeichen im String ist von Bedeutung.
string2	Stellt eine Liste von Ersetzungszeichen bereit. Die Position der einzelnen Ersetzungszeichen muss derjenigen in **string1** entsprechen.

Beispiel

Angenommen, Sie möchten den String `[12,3]` in `(12.3)` konvertieren. D.h. die eckigen Klammern müssen durch runde Klammern, Kommas durch einen Punkt ersetzt werden. Sie können zu diesem Zweck die Funktion `translate` folgendermaßen verwenden:



 © 2019-2025 Altova GmbH

Die erste Konstante im oben gezeigten Mapping liefert den zu verarbeitenden Input-String. Die zweite und dritte Konstante liefern als **string1** bzw. **string2**eine Liste von Zeichen.

string1	`[,]`
string2	`(.)`

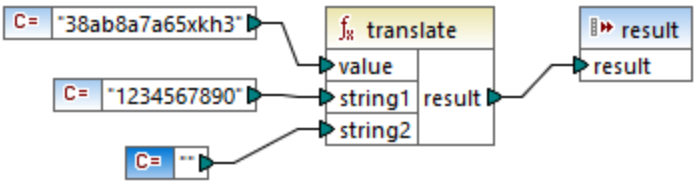
Beachten Sie dass **string1** und **string2** dieselbe Anzahl an Zeichen haben. Für jedes Zeichen in **string1** wird als Ersetzungszeichen das entsprechende Zeichen an derselben Position aus **string2** verwendet. Folglich werden die folgenden Ersetzungen vorgenommen:

- Jedes `[`-Zeichen wird durch ein `(`-Zeichen ersetzt.
- Jedes `,`-Zeichen wird durch ein `.`-Zeichen ersetzt.
- Jedes `]`-Zeichen wird durch ein `)`-Zeichen ersetzt.

Das Ergebnis des Mappings sieht folgendermaßen aus:

```
(12,3)
```

Diese Funktion kann auch dazu verwendet werden, um bestimmte Zeichen selektiv aus einem String zu entfernen. Setzen Sie dazu den Parameter **string1** auf das Zeichen, das entfernt werden soll und **string2** auf einen leeren String. Im unten gezeigten Mapping werden z.B. alle Ziffern aus dem String **`38ab8a7a65xkh3`** entfernt.



Das Ergebnis des Mappings sieht folgendermaßen aus:

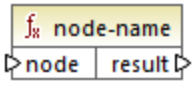
```
abaaxkh
```

5.7.6.11 xpath2 | accessors (Accessor-Funktionen)

Die Funktionen aus der Unterbibliothek **xpath2 | accessors** rufen Informationen über XML-Nodes oder Datenelemente ab. Diese Funktionen stehen zur Verfügung, wenn entweder die Sprache XSLT2 oder XQuery ausgewählt ist.

5.7.6.11.1 base-uri

Die **`base-uri`** Funktion erhält ein Node-Argument als Input und gibt die URI der XML-Ressource, die den Node enthält, zurück. Die Ausgabe hat den Typ `xs:string`.



Sprachen

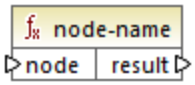
XQuery, XSLT 2.0, XSLT 3.0..

Parameter

Name	Typ	Beschreibung
node	`mf:node`	Der Input-Node.

5.7.6.11.2 node-name

Die **`node-name`**-Funktion erhält einen Node-Namen als Input-Argument und gibt seinen QName zurück. Wenn der QName als String dargestellt wird, hat er die Form `präfix:lokalerName`, wenn der Node ein Präfix hat, oder `lokalerName`, wenn der Node kein Präfix hat. Um die Namespace URI eines Node zu eruieren, verwenden Sie die **`namespace-URI-from-QName`** 296 Funktion.



Sprachen

XQuery, XSLT 2.0, XSLT 3.0..

Parameter

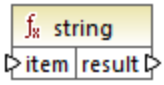
Name	Typ	Beschreibung
node	`mf:node`	Der Input-Node.

5.7.6.11.3 string

Die **`string`** Funktion funktioniert wie der `xs:string` Konstruktor: Sie konvertiert ihr Argument in `xs:string`.

Wenn das Input-Argument ein Wert eines atomaren Typs ist (z.B. `xs:decimal`), wird dieser atomare Wert in einen Wert vom Typ `xs:string` konvertiert. Wenn das Input-Argument ein Node ist, wird der String-Wert des Node extrahiert. (Der String-Wert eines Node ist eine Verkettung der Werte der untergeordneten Nodes des Node.)

 © 2019-2025 Altova GmbH



Sprachen

XQuery, XSLT 2.0, XSLT 3.0..

Parameter

Name	Typ	Beschreibung
item	`mf:item`	Der Input-Wert.

5.7.6.12 xpath2 | anyURI functions (anyURI-Funktionen)

Die Unterbibliothek **xpath2 | anyURI** enthält die `resolve-uri`-Funktion. Diese Funktion steht zur Verfügung, wenn entweder die Sprache XSLT2 oder XQuery ausgewählt ist.

5.7.6.12.1 resolve-uri

Die `resolve-uri` Funktion erhält eine URI als erstes Argument und löst diese anhand der Basis-URI im zweiten Argument auf. Die Ausgabe hat den Datentyp `xs:string`. Beide Inputs werden von der Funktion als Strings behandelt; es wird nicht überprüft, ob die von diesen URIs angegebenen Ressourcen tatsächlich vorhanden sind.

Sprachen

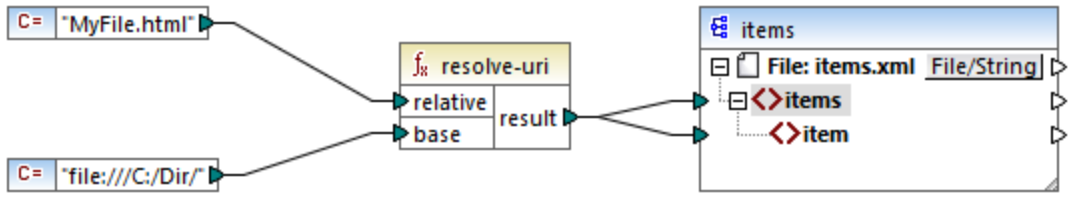
XQuery, XSLT 2.0, XSLT 3.0..

Parameter

Name	Typ	Beschreibung
relative	`xs:string`	Die relative URI, die anhand der Basis aufgelöst werden soll.
base	`xs:string`	Die Basis-URI.

Beispiel

Im Mapping unten liefert das erste Argument die relative URI `MyFile.html` und das zweite Argument die Basis-URI `file:///C:/Dir/`. Die aufgelöste URI ist eine Verkettung aus diesen beiden und lautet `file:///C:/Dir/MyFile.html`.

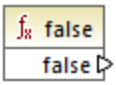


5.7.6.13 xpath2 | boolean functions (Boolesche Funktionen)

Die Booleschen Funktionen `true` und `false` erhalten kein Argument und geben die Booleschen Konstantenwerte **true** und **false** zurück. Sie können verwendet werden, wenn ein Boolescher Konstantenwert benötigt wird.

5.7.6.13.1 false

Gibt den Booleschen Wert **false** zurück.

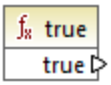


Sprachen

XQuery, XSLT 2.0, XSLT 3.0..

5.7.6.13.2 true

Gibt den Booleschen Wert **true** zurück.



Sprachen

XQuery, XSLT 2.0, XSLT 3.0..

5.7.6.14 xpath2 | constructors (Konstruktoren)

Mit den Funktionen aus der Unterbibliothek "constructors" der XPath 2.0-Bibliothek werden anhand des Input-Texts bestimmte Datentypen konstruiert. In der folgenden Tabelle sind die verfügbaren Konstruktorfunktionen aufgelistet.

 © 2019-2025 Altova GmbH

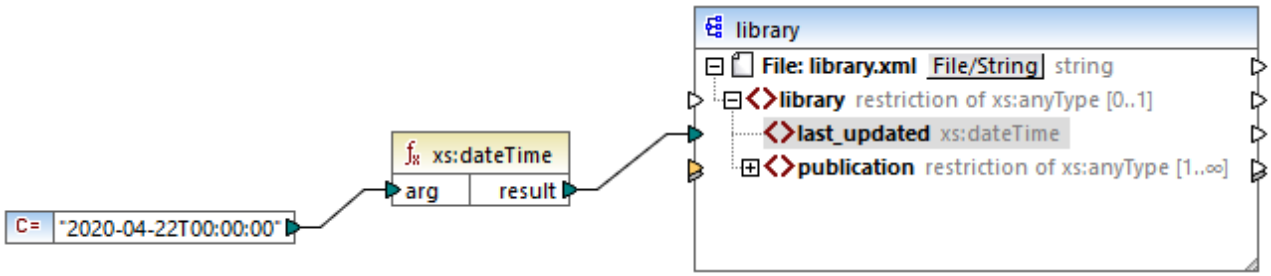
xs:ENTITY	**xs:double**	**xs:nonPositiveInteger**
xs:ID	**xs:duration**	**xs:normalizedString**
xs:IDREF	**xs:float**	**xs:positiveInteger**
xs:NCName	**xs:gDay**	**xs:short**
xs:NMTOKEN	**xs:gMonth**	**xs:string**
xs:Name	**xs:gMonthDay**	**xs:time**
xs:QName	**xs:gYear**	**xs:token**
xs:anyURI	**xs:gYearMonth**	**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**	

Sprachen

XQuery, XSLT 2.0, XSLT 3.0..

Beispiel

Normalerweise muss das lexikalische Format des Input-Texts dem vom zu konstruierenden Datentyp erwarteten entsprechen. Andernfalls ist die Transformation nicht erfolgreich. Um z.B. mit Hilfe der **xs:dateTime**-Konstruktorfunktion einen `xs:dateTime`-Wert zu konstruieren, muss der Input-Text das lexikalische Format des `xs:dateTime`-Datentyps haben. Dieses Format ist **YYYY-MM-DDTHH:mm:ss**.



Im oben gezeigten Mapping wurde als Input-Argument der Funktion eine String-Konstante (**"2020-04-28T00:00:00"**) verwendet. Dieser Input hätte auch aus einem Datenelement im Quelldokument abgerufen werden können. Die **xs:dateTime**-Funktion gibt den Wert **2020-04-28T00:00:00** vom Typ `xs:dateTime` zurück.

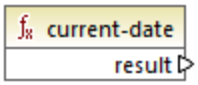
Um zu sehen, welcher Datentyp (einschließlich des Datentyps der Funktionsargumente) von einem Mapping-Datenelement erwartet wird, platzieren Sie den Mauszeiger über den entsprechenden Input- oder Output-Konnektor.

5.7.6.15 xpath2 | context functions (Kontextfunktionen)

Die Kontextfunktionen aus der **xpath2**-Bibliothek stellen verschiedene Informationen über das aktuelle Datum und die Uhrzeit, die vom Prozessor verwendete Standard-Collation, die Größe der aktuellen Sequenz und die Position des aktuellen Node bereit.

5.7.6.15.1 current-date

Gibt das aktuelle Datum (`xs:date`) anhand der Systemuhr zurück.

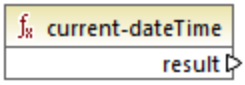


Sprachen

XQuery, XSLT 2.0, XSLT 3.0..

5.7.6.15.2 current-dateTime

Gibt das aktuelle Datum und die aktuelle Uhrzeit (`xs:dateTime`) anhand der Systemuhr zurück.



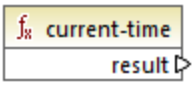
Sprachen

XQuery, XSLT 2.0, XSLT 3.0..

© 2019-2025 Altova GmbH

5.7.6.15.3 current-time

Gibt die aktuelle Uhrzeit (`xs:time`) anhand der Systemuhr zurück.



Sprachen

XQuery, XSLT 2.0, XSLT 3.0..

5.7.6.15.4 default-collation

Die Funktion **`default-collation`** erhält kein Argument und gibt die Standard-Collation zurück, d.h. die Collation, die verwendet wird, wenn keine Collation für eine Funktion, für die eine solche definiert werden kann, festgelegt ist.

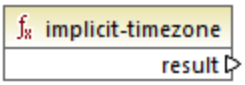
Vergleiche wie für die Funktionen **`max-string`** und **`min-string`** basieren auf der Standard-Collation.

Sprachen

XQuery, XSLT 2.0, XSLT 3.0..

5.7.6.15.5 implicit-timezone

Gibt den Wert der Eigenschaft "implizite Zeitzone" aus dem Auswertungskontext zurück.

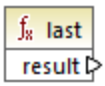


Sprachen

XQuery, XSLT 2.0, XSLT 3.0..

5.7.6.15.6 last

Gibt die Anzahl der Datenelemente innerhalb der gerade verarbeiteten Sequenz von Datenelementen zurück. Beachten Sie, dass die Sequenz der Datenelemente durch den aktuellen Mapping-Kontext 79 bestimmt wird, wie im Beispiel unten beschrieben.



Sprachen

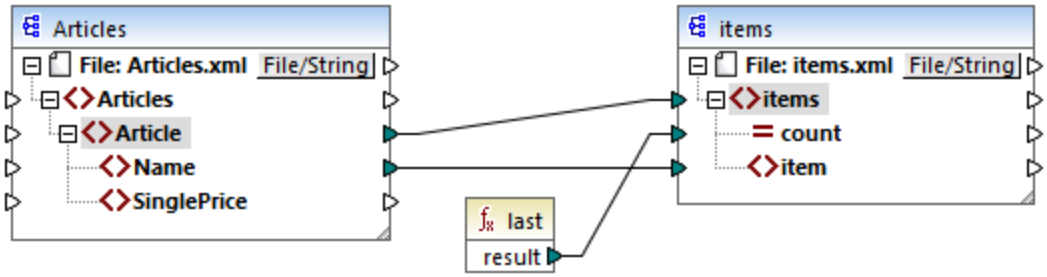
XQuery, XSLT 2.0, XSLT 3.0..

Beispiel

Angenommen, Sie haben die folgende XML-Quelldatei:

```
<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>
```

Es sollen Daten in eine XML-Datei mit einem anderen Schema kopiert werden. Außerdem muss die Anzahl aller Datenelemente in der XML-Zieldatei gespeichert werden. Dies lässt sich mit einem Mapping wie dem unten gezeigten, ermitteln:



Im Beispiel unten gibt die Funktion `last` die Position des letzten Node im aktuellen Parent-Kontext zurück und befüllt das Attribut **count** mit dem Wert **3**.

 © 2019-2025 Altova GmbH

```
<items count="3">
   <item>T-Shirt</item>
   <item>Pants</item>
   <item>Jacket</item>
</items>
```

Beachten Sie, dass der Wert **3** die Position des letzen Datenelements (also die Anzahl aller Datenelemente) in dem durch die Verbindung zwischen **Article** und **items** erstellten Mapping-Kontext ist. Wäre diese Verbindung nicht vorhanden, würden dennoch Datenelemente in die Zielkomponente kopiert werden, doch würde die Funktion **`last`** fälschlicherweise den Wert **1** zurückgeben, da sie keinen Parent-Kontext 83 hätte, über den sie iterieren kann. (Genauer gesagt, würde die Funktion den impliziten zwischen den Root-Elementen der beiden Komponenten erstellten Standardkontext verwenden, der eine Sequenz von 1- und nicht, wie erwartet 3 - Datenelementen erzeugt).

Im Allgemeinen empfiehlt es sich, anstelle der **`last`**-Funktion, die **`count`** 255-Funktion aus der **core**-Bibliothek zu verwenden, da die count-Funktion ein **parent-context**-Argument hat, mit Hilfe dessen Sie den Mapping-Kontext explizit ändern können.

5.7.6.16 xpath2 | durations, date and time functions (Zeitdauer-, Datums- und Uhrzeitfunktionen)

Mit Hilfe der Zeitdauer-, Datums- und Uhrzeitfunktionen aus der **xpath2**-Bibliothek können Sie die Zeitzone in Datums- und Uhrzeitwerten anpassen, bestimmte Komponenten aus Datums-, Uhrzeit- und Zeitdauerwerten extrahieren und Datums- und Uhrzeitwerte subtrahieren.

Anpassen der Zeitzone

Zum Anpassen der Zeitzone in Datums- und Uhrzeitwerten stehen die folgenden Funktionen zur Verfügung:

- **`adjust-date-to-timezone`**
- **`adjust-date-to-timezone`** (mit Zeitzonenargument)
- **`adjust-dateTime-to-timezone`**
- **`adjust-dateTime-to-timezone`** (mit Zeitzonenargument)
- **`adjust-time-to-timezone`**
- **`adjust-time-to-timezone`** (mit Zeitzonenargument)

Jede dieser Funktionen erhält einen `xs:date`, `xs:time` oder `xs:dateTime`-Wert als erstes Argument und passt die Input-Daten je nach dem Wert des zweiten Arguments (falls eines vorhanden ist) durch Hinzufügen, Entfernen oder Ändern der Zeitzonen-Komponente an.

Wenn das erste Argument keine Zeitzone enthält, (z.B. das Datum `2009-01` oder die Uhrzeit `14:00:00`), ergeben sich die folgenden Möglichkeiten:

- Wenn das **Zeitzonenargument** vorhanden ist, enthält das Ergebnis die im zweiten Argument angegebene Zeitzone. Die Zeitzone im zweiten Argument wird hinzugefügt.
- Wenn das **Zeitzonenargument** fehlt, enthält das Ergebnis die implizite Zeitzone, d.h. die Zeitzone des Systems. Die Zeitzone des Systems wird hinzugefügt.
- Wenn das **Zeitzonenargument** leer ist, enthält das Ergebnis keine Zeitzone.

Wenn das erste Argument eine Zeitzone enthält, (z.B. das Datum `01.01.2020+01:00` oder die Uhrzeit `14:00:00+01:00`), ergeben sich die folgenden Möglichkeiten:

- Wenn das **Zeitzonenargument** vorhanden ist, enthält das Ergebnis die im zweiten Argument angegebene Zeitzone. Die ursprüngliche Zeitzone wird durch die im zweiten Argument definierte Zeitzone ersetzt.
- Wenn das **Zeitzonenargument** fehlt, enthält das Ergebnis die implizite Zeitzone, d.h. die Zeitzone des Systems. Die ursprüngliche Zeitzone wird durch die Zeitzone des Systems ersetzt.
- Wenn das **Zeitzonenargument** leer ist, enthält das Ergebnis keine Zeitzone.

Extrahieren von Komponenten von Datums- und Uhrzeitwerten

Um aus Datums- und Uhrzeitwerten numerische Werte wie Stunden, Minuten, Monate, usw. zu extrahieren, stehen die folgenden Funktionen zur Verfügung:

- **`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`**

Jede dieser Funktionen extrahiert eine bestimmte Komponente aus `xs:date`-, `xs:time`-, `xs:dateTime`- und `xs:duration`-Werten. Das Ergebnis ist entweder ein `xs:integer`- oder `xs:decimal`-Typ.

Extrahieren von Zeitdauerkomponenten

Um aus einer Zeitdauer Uhrzeitkomponenten zu extrahieren, stehen die folgenden Funktionen zur Verfügung:

- **`days-from-duration`**
- **`hours-from-duration`**
- **`minutes-from-duration`**
- **`months-from-duration`**
- **`seconds-from-duration`**
- **`years-from-duration`**

Die Zeitdauer muss entweder als `xs:yearMonthDuration` (zum Extrahieren von Jahren und Monaten) oder als `xs:dayTimeDuration` (zum Extrahieren von Tagen, Stunden, Minuten und Sekunden) definiert werden. Alle Funktionen geben ein Ergebnis vom Typ `xs:integer` zurück; ausgenommen davon ist die Funktion **`seconds-from-duration`**, die ein Ergebnis vom Typ `xs:decimal` zurückgibt.

© 2019-2025 Altova GmbH

Subtrahieren von Datums- und Uhrzeitwerten

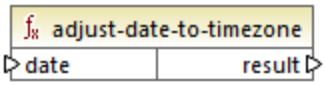
Zum Subtrahieren von Datums- und Uhrzeitwerten stehen die folgenden Funktionen zur Verfügung:

- **subtract-dateTimes**
- **subtract-dates**
- **subtract-times**

Mit Hilfe der Subtraktionsfunktionen können Sie einen Uhrzeitwert von einem anderen subtrahieren und einen Zeitdauerwert errechnen.

5.7.6.16.1 adjust-date-to-timezone

Passt einen `xs:date`-Wert an die implizite Zeitzone im Auswertungskontext (die Zeitzzone des Systems) an.



Sprachen

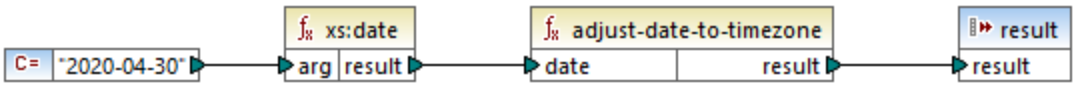
XQuery, XSLT 2.0, XSLT 3.0..

Parameter

Name	Typ	Beschreibung
date	**xs:date**	Der Input-Wert vom Typ `xs:date`.

Beispiel

Im folgenden Mapping wird anhand eines String ein `xs:date`-Wert konstruiert und als Argument für die **adjust-date-to-timezone** bereitgestellt.

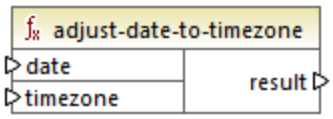


XSLT 2.0-Mapping

Wenn das Mapping auf einem Rechner läuft, dessen Systemzeitzone +02:00 ist, passt die Funktion den Datumswert an, um die Zeitzone des Systems zu inkludieren. Die Mapping-Ausgabe ist daher **2020-04-30+02:00**.

5.7.6.16.2 adjust-date-to-timezone

Passt einen `xs:date`-Wert an eine bestimmte Zeitzone oder gar keine Zeitzone an. Wenn das Argument **timezone** eine leere Sequenz ist, gibt die Funktion einen `xs:date`-Wert ohne eine Zeitzone zurück. Andernfalls gibt sie einen `xs:date`-Wert mit einer Zeitzone zurück.



Sprachen

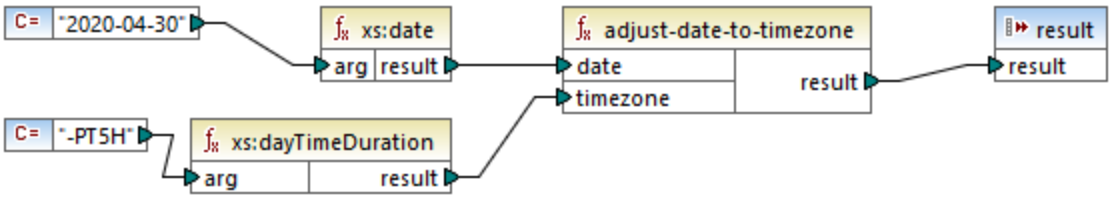
XQuery, XSLT 2.0, XSLT 3.0..

Parameter

Name	Typ	Beschreibung
date	**`xs:date`**	Der Input-Wert vom Typ `xs:date`.
timezone	**`xs:dayTimeDuration`**	Die als `xs:dayTimeDuration`-Wert ausgedrückte Zeitzone. Der Wert kann negativ sein. So kann etwa ein Zeitzonenwert von -5 Stunden als **`-PT5H`** ausgedrückt werden.

Beispiel

Im folgenden Mapping werden beide Parameter für die Funktion **`adjust-date-to-timezone`** mit Hilfe der entsprechenden XPath 2 Konstruktorfunktionen 342 anhand von Strings konstruiert. Ziel des Mappings ist eine Anpassung der Zeitzone an -5 Stunden. Diese Zeitzone kann als **`-PT5H`** ausgedrückt werden.



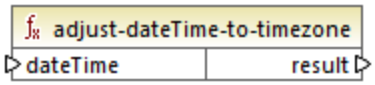
XSLT 2.0-Mapping

Die Funktion passt den Datumswert an die als Argument bereitgestellte Zeitzone an. Die Mapping-Ausgabe ist daher **`2020-04-30-05:00`**.

 © 2019-2025 Altova GmbH

5.7.6.16.3 adjust-dateTime-to-timezone

Passt einen `xs:dateTime`-Wert an die implizite Zeitzone im Auswertungskontext (die Zeitzzone des Systems) an.



Sprachen

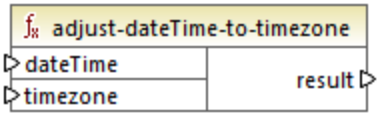
XQuery, XSLT 2.0, XSLT 3.0..

Parameter

Name	Typ	Beschreibung
dateTime	`xs:dateTime`	Der Input-Wert vom Typ `xs:dateTime`.

5.7.6.16.4 adjust-dateTime-to-timezone

Passt einen `xs:dateTime`-Wert an eine bestimmte Zeitzone oder gar keine Zeitzone an. Wenn das Argument **timezone** eine leere Sequenz ist, gibt die Funktion einen `xs:dateTime`-Wert ohne eine Zeitzone zurück. Andernfalls gibt sie einen `xs:dateTime`-Wert mit einer Zeitzone zurück.



Sprachen

XQuery, XSLT 2.0, XSLT 3.0..

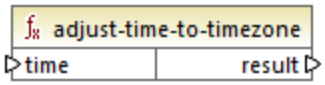
Parameter

Name	Typ	Beschreibung
dateTime	`xs:dateTime`	Der Input-Wert vom Typ `xs:dateTime`.
timezone	`xs:dayTimeDuration`	Die als `xs:dayTimeDuration`-Wert ausgedrückte Zeitzone. Der Wert kann negativ sein. So kann etwa ein Zeitzonenwert von -5

Name	Typ	Beschreibung
		Stunden als `-PT5H` ausgedrückt werden.

5.7.6.16.5 adjust-time-to-timezone

Passt einen `xs:Time`-Wert an die implizite Zeitzone im Auswertungskontext (die Zeitzzone des Systems) an.



Sprachen

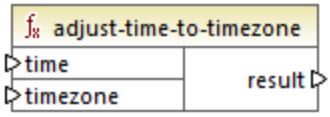
XQuery, XSLT 2.0, XSLT 3.0..

Parameter

Name	Typ	Beschreibung
time	`xs:time`	Der Input-Wert vom Typ `xs:time`.

5.7.6.16.6 adjust-time-to-timezone

Passt einen `xs:time`-Wert an eine bestimmte Zeitzone oder gar keine Zeitzone an. Wenn das Argument **timezone** eine leere Sequenz ist, gibt die Funktion einen `xs:time`-Wert ohne eine Zeitzone zurück. Andernfalls gibt sie einen `xs:time`-Wert mit einer Zeitzone zurück.



Sprachen

XQuery, XSLT 2.0, XSLT 3.0..

Parameter

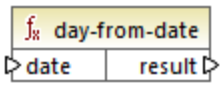
Name	Typ	Beschreibung
time	`xs:time`	Der Input-Wert vom Typ `xs:time`.
timezone	`xs:dayTimeDuration`	Die als `xs:dayTimeDuration`-Wert ausgedrückte Zeitzone. Der

© 2019-2025 Altova GmbH

Name	Typ	Beschreibung
		Wert kann negativ sein. So kann etwa ein Zeitzonenwert von -5 Stunden als `-PT5H` ausgedrückt werden.

5.7.6.16.7 day-from-date

Gibt einen `xs:integer`-Wert für den Tagesteil des als Argument bereitgestellten `xs:date`-Werts zurück.



Sprachen

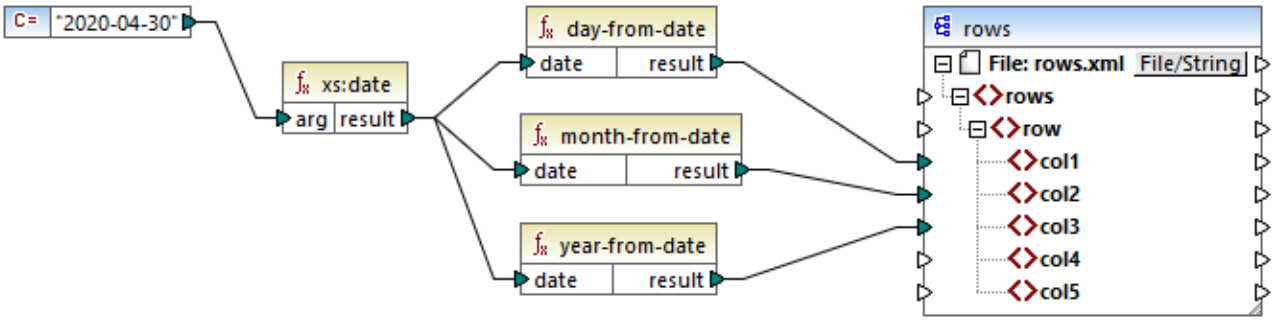
XQuery, XSLT 2.0, XSLT 3.0..

Parameter

Name	Typ	Beschreibung
date	`xs:date`	Der Input-Wert vom Typ `xs:date`.

Beispiel

Im folgenden Mapping wird ein String mit Hilfe der **`xs:date`**-Konstruktorfunktion in einen `xs:date`-Wert konvertiert. Die Funktionen **`day-from-date`**, **`month-from-date`** und **`year-from-date`** extrahieren jeweils den entsprechenden Teil des Datums und schreiben ihn in ein separates Datenelement in der XML-Zieldatei.



XQuery 1.0-Mapping

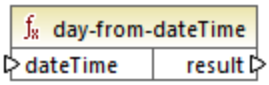
Das Ergebnis des Mappings sieht folgendermaßen aus:

```
<rows>
   <row>
```

```
      <col1>30</col1>
      <col2>4</col2>
      <col3>2020</col3>
   </row>
</rows>
```

5.7.6.16.8 day-from-dateTime

Gibt einen `xs:integer`-Wert für den Tagesteil des als Argument bereitgestellten `xs:dateTime`-Werts zurück.



Sprachen

XQuery, XSLT 2.0, XSLT 3.0..

Parameter

Name	Typ	Beschreibung
dateTime	`xs:dateTime`	Der Input-Wert vom Typ `xs:dateTime`.

5.7.6.16.9 days-from-duration

Gibt einen `xs:integer`-Wert für die "Tage"-Komponente der kanonischen Darstellung des als Argument bereitgestellten Zeitdauerwerts zurück.

Sprachen

XQuery, XSLT 2.0, XSLT 3.0..

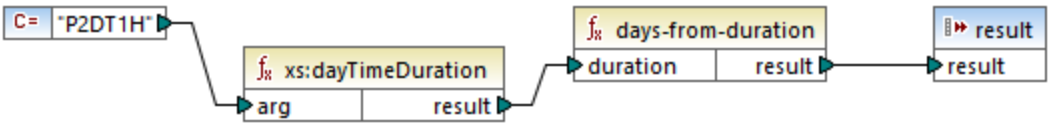
Parameter

Name	Typ	Beschreibung
duration	`xs:duration`	Der Input-Wert vom Typ `xs:duration`.

Beispiel

Im unten gezeigten Mapping wird der `xs:dayTimeDuration`-Wert anhand von **`P2DT1H`** (2 Tage und 1 Stunde) konstruiert und als Input für die Funktion **`days-from-duration`** bereitgestellt. Das Ergebnis ist **2**.

© 2019-2025 Altova GmbH



XSLT 2.0-Mapping

Anmerkung: Wenn die Zeitdauer **`P1DT24H`** (1 Tag und 24 Stunden) ist, gibt die Funktion **2** und nicht **1** zurück. Der Grund dafür ist, dass die kanonische Darstellung von **`P1DT24H`** tatsächlich **`P2D`** (2 Tage) ist.

5.7.6.16.10 hours-from-dateTime

Gibt einen `xs:integer`-Wert für den Stundenteil des als Argument bereitgestellten `xs:dateTime`-Werts zurück.

Sprachen

XQuery, XSLT 2.0, XSLT 3.0..

Parameter

Name	Typ	Beschreibung
dateTime	**`xs:dateTime`**	Der Input-Wert vom Typ `xs:dateTime`.

5.7.6.16.11 hours-from-duration

Gibt einen `xs:integer`-Wert für die "Stunden"-Komponente der kanonischen Darstellung des als Argument bereitgestellten Zeitdauerwerts zurück.

Sprachen

XQuery, XSLT 2.0, XSLT 3.0..

Parameter

Name	Typ	Beschreibung
duration	**`xs:duration`**	Der Input-Wert vom Typ `xs:duration`.

Beispiel

Wenn die Zeitdauer **`PT1H60M`** (1 Stunde und 60 Minuten) ist, gibt die Funktion **2** und nicht **1** zurück. Der Grund dafür ist, dass die kanonische Darstellung von **`PT1H60M`** tatsächlich **`PT2H`** (2 Stunden) ist.

5.7.6.16.12 hours-from-time

Gibt einen `xs:integer`-Wert für den Stundenteil des als Argument bereitgestellten `xs:time`-Werts zurück.

Sprachen

XQuery, XSLT 2.0, XSLT 3.0..

Parameter

Name	Typ	Beschreibung
time	`xs:time`	Der Input-Wert vom Typ `xs:time`.

5.7.6.16.13 minutes-from-dateTime

Gibt einen `xs:integer`-Wert für den Minutenteil des als Argument bereitgestellten `xs:dateTime`-Werts zurück.

Sprachen

XQuery, XSLT 2.0, XSLT 3.0..

Parameter

Name	Typ	Beschreibung
dateTime	`xs:dateTime`	Der Input-Wert vom Typ `xs:dateTime`.

5.7.6.16.14 minutes-from-duration

Gibt einen `xs:integer`-Wert für die "Minuten"-Komponente der kanonischen Darstellung des als Argument bereitgestellten Zeitdauerwerts zurück.

Sprachen

XQuery, XSLT 2.0, XSLT 3.0..

Parameter

Name	Typ	Beschreibung
duration	`xs:duration`	Der Input-Wert vom Typ `xs:duration`.

© 2019-2025 Altova GmbH

Beispiel

Wenn die Zeitdauer **PT1M60S** (1 Minute und 60 Sekunden) ist, gibt die Funktion **2** und nicht **1** zurück. Der Grund dafür ist, dass die kanonische Darstellung von **PT1M60S** tatsächlich **PT2M** (2 Minuten) ist.

5.7.6.16.15 minutes-from-time

Gibt einen `xs:integer`-Wert für den Minutenteil des als Argument bereitgestellten `xs:time`-Werts zurück.

Sprachen

XQuery, XSLT 2.0, XSLT 3.0..

Parameter

Name	Typ	Beschreibung
time	**xs:time**	Der Input-Wert vom Typ `xs:time`.

5.7.6.16.16 month-from-date

Gibt einen `xs:integer`-Wert für den Monatsteil des als Argument bereitgestellten `xs:date`-Werts zurück.

Sprachen

XQuery, XSLT 2.0, XSLT 3.0..

Parameter

Name	Typ	Beschreibung
date	**xs:date**	Der Input-Wert vom Typ `xs:date`.

5.7.6.16.17 month-from-dateTime

Gibt einen `xs:integer`-Wert für den Monatsteil des als Argument bereitgestellten `xs:dateTime`-Werts zurück.

Sprachen

XQuery, XSLT 2.0, XSLT 3.0..

Parameter

Name	Typ	Beschreibung
dateTime	`xs:dateTime`	Der Input-Wert vom Typ `xs:dateTime`.

5.7.6.16.18 months-from-duration

Gibt einen `xs:integer`-Wert für die Monatskomponente der kanonischen Darstellung des als Argument bereitgestellten Zeitdauerwerts zurück.

Sprachen

XQuery, XSLT 2.0, XSLT 3.0..

Parameter

Name	Typ	Beschreibung
duration	`xs:duration`	Der Input-Wert vom Typ `xs:duration`.

5.7.6.16.19 seconds-from-dateTime

Gibt einen xs:integer-Wert zurück, der für die Sekundenkomponente im lokalisierten Wert von dateTime steht.

Sprachen

XQuery, XSLT 2.0, XSLT 3.0..

Parameter

Name	Typ	Beschreibung
dateTime	`xs:dateTime`	

5.7.6.16.20 seconds-from-duration

Gibt einen `xs:integer`-Wert für die Sekundenkomponente der kanonischen Darstellung des als Argument bereitgestellten Zeitdauerwerts zurück.

Sprachen

XQuery, XSLT 2.0, XSLT 3.0..

© 2019-2025 Altova GmbH

Parameter

Name	Typ	Beschreibung
duration	`xs:duration`	Der Input-Wert vom Typ `xs:duration`.

5.7.6.16.21 seconds-from-time

Gibt einen `xs:integer`-Wert für den Sekundenteil des als Argument bereitgestellten `xs:time`-Werts zurück.

Sprachen

XQuery, XSLT 2.0, XSLT 3.0..

Parameter

Name	Typ	Beschreibung
time	`xs:time`	Der Input-Wert vom Typ `xs:time`.

5.7.6.16.22 subtract-dateTimes

Gibt den `xs:dayTimeDuration`-Wert zurück, der der Differenz zwischen dem normalisierten Wert von **dateTime1** und dem normalisierten Wert von **dateTime2** entspricht, zurück.

Sprachen

XQuery, XSLT 2.0, XSLT 3.0..

Parameter

Name	Typ	Beschreibung
dateTime1	`xs:dateTime`	Der erste Input-Wert.
dateTime2	`xs:dateTime`	Der zweite Input-Wert.

5.7.6.16.23 subtract-dates

Gibt den `xs:dayTimeDuration`-Wert, der der Differenz zwischen dem normalisierten Wert von **date1** und dem normalisierten Wert von **date2** entspricht, zurück.

Sprachen

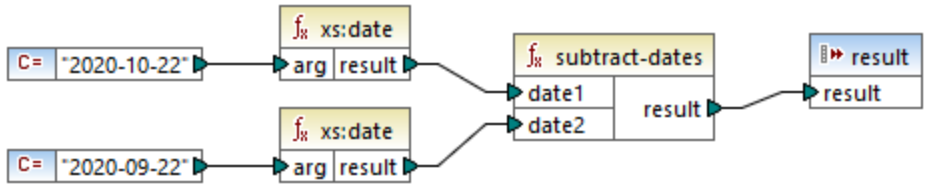
XQuery, XSLT 2.0, XSLT 3.0..

Parameter

Name	Typ	Beschreibung
date1	`xs:date`	Der erste Input-Wert.
date2	`xs:date`	Der zweite Input-Wert.

Beispiel

Im unten gezeigten Mapping werden zwei Datumswerte voneinander subtrahiert (2020-10-22 minus 2020-09-22). Das Ergebnis ist der Wert **`P30D`** vom Typ `xs:dayTimeDuration`, der eine Zeitdauer von 30 Tagen darstellt.



5.7.6.16.24 subtract-times

Gibt den `xs:dayTimeDuration`-Wert, der der Differenz zwischen dem normalisierten Wert von **time1** und dem normalisierten Wert von **time2** entspricht, zurück.

Sprachen

XQuery, XSLT 2.0, XSLT 3.0..

Parameter

Name	Typ	Beschreibung
time1	`xs:time`	Der erste Input-Wert.
time2	`xs:time`	Der zweite Input-Wert.

5.7.6.16.25 timezone-from-date

Gibt die Zeitzonenkomponente des als Argument bereitgestellten Datums zurück. Das Ergebnis ist ein `xs:dayTimeDuration`-Wert, der die Abweichung von UTC angibt; der Wert kann von +14:00 bis einschließlich -14:00 Stunden reichen.

Sprachen

XQuery, XSLT 2.0, XSLT 3.0..

© 2019-2025 Altova GmbH

Parameter

Name	Typ	Beschreibung
date	`xs:date`	Der Input-Wert vom Typ `xs:date`.

5.7.6.16.26 timezone-from-dateTime

Gibt die Zeitzonenkomponente des als Argument bereitgestellten `xs:dateTime`-Werts zurück. Das Ergebnis ist ein `xs:dayTimeDuration`-Wert, der die Abweichung von UTC angibt; der Wert kann von +14:00 bis einschließlich -14:00 Stunden reichen.

Sprachen

XQuery, XSLT 2.0, XSLT 3.0..

Parameter

Name	Typ	Beschreibung
dateTime	`xs:dateTime`	Der Input-Wert vom Typ `xs:dateTime`.

5.7.6.16.27 timezone-from-time

Gibt die Zeitzonenkomponente des als Argument bereitgestellten `xs:time`-Werts zurück. Das Ergebnis ist ein `xs:dayTimeDuration`-Wert, der die Abweichung von UTC angibt; der Wert kann von +14:00 bis einschließlich -14:00 Stunden reichen.

Sprachen

XQuery, XSLT 2.0, XSLT 3.0..

Parameter

Name	Typ	Beschreibung
time	`xs:time`	Der Input-Wert vom Typ `xs:time`.

5.7.6.16.28 year-from-date

Gibt einen `xs:integer`-Wert für den Jahresteil des als Argument bereitgestellten `xs:date`-Werts zurück.

Sprachen

XQuery, XSLT 2.0, XSLT 3.0..

Parameter

Name	Typ	Beschreibung
date	**`xs:date`**	Der Input-Wert vom Typ `xs:date`.

5.7.6.16.29 year-from-dateTime

Gibt einen `xs:integer`-Wert für den Jahresteil des als Argument bereitgestellten `xs:dateTime`-Werts zurück.

Sprachen

XQuery, XSLT 2.0, XSLT 3.0..

Parameter

Name	Typ	Beschreibung
dateTime	**`xs:dateTime`**	Der Input-Wert vom Typ `xs:dateTime`.

5.7.6.16.30 years-from-duration

Gibt einen `xs:integer`-Wert für die Jahreskomponente in der kanonisch-lexikalischen Darstellung des als Argument bereitgestellten Zeitdauerwerts zurück.

Sprachen

XQuery, XSLT 2.0, XSLT 3.0..

Parameter

Name	Typ	Beschreibung
duration	**`xs:duration`**	Der Input-Wert vom Typ `xs:duration`.

© 2019-2025 Altova GmbH

5.7.6.17 xpath2 | node functions (Node-Funktionen)

Die Node-Funktionen aus der **xpath2**-Bibliothek liefern Informationen über Nodes (Datenelemente) in einer Mapping-Komponente.

Die **lang** Funktion erhält ein String-Argument, das einen Sprachcode definiert (wie z.B. "en"). Die Funktion gibt je nachdem, ob der Kontext-Node ein **xml:lang** Attribut mit einem Wert hat, der mit dem Argument der Funktion übereinstimmt, entweder **true** oder `false` zurück.

Die Funktionen **local-name**, **name** und **namespace-uri** geben den lokalen Namen, den Namen bzw. die Namespace URI des Input-Node zurück. So ist z.B. beim Node **altova:Products** der lokale Name **Products**, der Name **altova:Products** und die Namespace URI die URI des Namespace, an den das Präfix altova: gebunden ist (siehe Beispiel zur Funktion **local-name** 365). Jede dieser drei Funktionen hat zwei Varianten:

- ohne Argument: In diesem Fall wird die Funktion auf den Kontext-Node (ein Beispiel für einen Kontext-Node finden Sie im Beispiel oben zur **lang** 363 Funktion) angewendet.
- mit einem Argument, das ein Node sein muss: Die Funktion wird auf den verbundenen Node angewendet.

Die **number**-Funktion erhält einen Node als Input, zerlegt den Node in seine Bestandteile (d.h. extrahiert seinen Inhalt), konvertiert den Wert in eine Dezimalzahl und gibt den konvertierten Wert zurück. Die **number**-Funktion hat zwei Varianten:

- ohne Argument: In diesem Fall wird die Funktion auf den Kontext-Node (ein Beispiel für einen Kontext-Node finden Sie im Beispiel oben zur **lang** 363 Funktion) angewendet.
- mit einem Argument, das ein Node sein muss: Die Funktion wird auf den verbundenen Node angewendet.

5.7.6.17.1 lang

Gibt **true** zurück, wenn der Kontext-Node ein `xml:lang`-Attribut mit einem Wert hat, der entweder genau mit dem Argument **testlang** übereinstimmt oder eine Untergruppe davon ist. Andernfalls gibt die Funktion **false** zurück.

Sprachen

XQuery, XSLT 2.0, XSLT 3.0..

Parameter

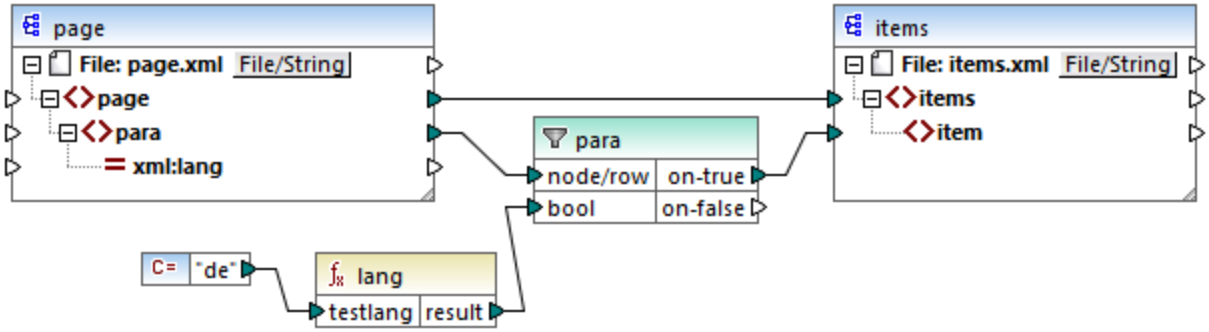
Name	Typ	Beschreibung
testlang	`xs:string`	Der zu überprüfende Sprachencode, z.B. "en".

Beispiel

Die folgende XML-Datei enthält **para**-Elemente mit verschiedenen Werten für das 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>
```

Im unten gezeigten Mapping werden mit Hilfe der Funktion **`lang`** unabhängig von der Landesvariante nur die deutschen Absätze gefiltert.



XSLT 2.0-Mapping

Im obigen Mapping wird für jedes **para**-Element in der Quellkomponente auf Basis von Bedingungen ein **item**Element in der Zielkomponente erstellt. Die Bedingung wird durch einen Filter bereitgestellt, der nur die Nodes an die Zielkomponente übergibt, für die die Funktion **`lang`** **true**, zurückgibt. D.h. nur die Nodes, deren `xml:lang`-Attribut auf "de" (oder eine Untergruppe von "de") gesetzt ist, erfüllen die Filterbedingung. Die Mapping-Ausgabe sieht daher folgendermaßen aus:

```
<items>
   <item>Grüss Gott!</item>
   <item>Guten Tag!</item>
   <item>Grüezi!</item>
</items>
```

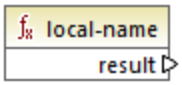
Beachten Sie, dass die **`lang`**-Funktion aufgrund der Parent-Verbindung zwischen **para** und **item** im Kontext der einzelnen **para**-Elemente ausgeführt wird, siehe auch Der Mapping-Kontext 79.

5.7.6.17.2 local-name

Gibt den lokalen Teil des Namens des Kontext-Node als `xs:string` zurück. Dies ist eine parameterlose Variante der **`local-name`**-Funktion, bei der der Kontext-Node durch die Verbindungen in Ihrem Mapping

 © 2019-2025 Altova GmbH

festgelegt wird. Um einen Node explizit zu definieren, verwenden Sie die Funktion **local-name** 365, die einen Input-Node als Parameter erhält.

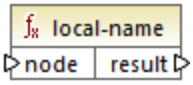


Sprachen

XQuery, XSLT 2.0, XSLT 3.0..

5.7.6.17.3 local-name

Gibt den lokalen Teil des Namens des **Node** als `xs:string` zurück.



Sprachen

XQuery, XSLT 2.0, XSLT 3.0..

Parameter

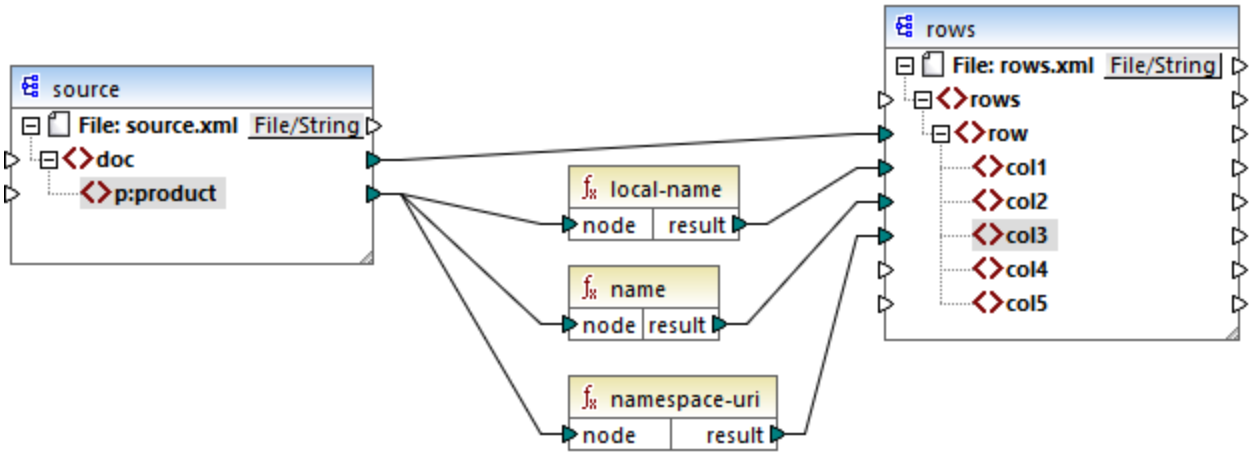
Name	Typ	Beschreibung
node	**node()**	Der Input-Node.

Beispiel

In der folgenden XML-Datei ist der Name des Elements `p:product` ein qualifizierter Name (QName) mit einem Präfix. Das Präfix "p" ist auf den Namespace "http://mycompany.com" gemappt.

```
<?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>
```

Im folgenden Mapping werden der lokale Name, der Name und die Namespace URI des Node extrahiert und diese Werte werden in eine Zieldatei geschrieben:



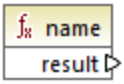
XSLT 2.0-Mapping

Weiter unten sehen Sie das Ergebnis des Mappings. In den einzelnen **col**-Datenelementen wird das Ergebnis der Funktion `local-name`, `name` bzw. `namespace-uri` aufgelistet.

```
<rows>
   <row>
      <col1>product</col1>
      <col2>p:product</col2>
      <col3>http://mycompany.com</col3>
   </row>
</rows>
```

5.7.6.17.4 name

Gibt den Namen des Kontext-Node zurück. Dies ist eine parameterlose Variante der `name`-Funktion, bei der der Kontext-Node durch die Verbindungen in Ihrem Mapping festgelegt wird. Um einen Node explizit zu definieren, verwenden Sie die Funktion `name` 367, die einen Input-Node als Parameter erhält.



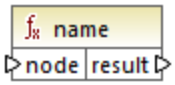
Sprachen

XQuery, XSLT 2.0, XSLT 3.0..

 © 2019-2025 Altova GmbH

5.7.6.17.5 name

Gibt den Namen eines Node zurück.



Sprachen

XQuery, XSLT 2.0, XSLT 3.0..

Parameter

Name	Typ	Beschreibung
node	`node()`	Der Input-Node.

Beispiel

Siehe das Beispiel zur Funktion `local-name` 365.

5.7.6.17.6 namespace-uri

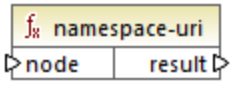
Gibt die Namespace URI des QName des Kontext-Node als `xs:string` zurück. Dies ist eine parameterlose Variante der `namespace-uri`-Funktion, bei der der Kontext-Node durch die Verbindungen in Ihrem Mapping festgelegt wird. Um einen Node explizit zu definieren, verwenden Sie die Funktion `namespace-uri` 367, die einen Input-Node als Parameter erhält.

Sprachen

XQuery, XSLT 2.0, XSLT 3.0..

5.7.6.17.7 namespace-uri

Gibt die Namespace URI des QName von **node** als `xs:string` zurück.



Sprachen

XQuery, XSLT 2.0, XSLT 3.0..

Parameter

Name	Typ	Beschreibung
node	`node()`	Der Input-Node.

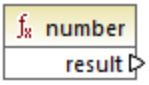
Beispiel

Siehe das Beispiel zur Funktion `local-name` 365.

5.7.6.17.8 number

Gibt den in einen `xs:double`-Typ konvertierten Wert des Kontext-Node zurück. Dies ist eine parameterlose Variante der `number`-Funktion, bei der der Kontext-Node durch die Verbindungen in Ihrem Mapping festgelegt wird. Um einen Node explizit zu definieren, verwenden Sie die Funktion `number` 368, die einen Input-Node als Parameter erhält.

Die einzigen Typen, die in Zahlen konvertiert werden können, sind Boolesche Werte, numerische Strings und andere numerische Typen. Nicht numerische Input-Werte (wie z.B. ein nicht numerischer String) führen zum Resultat NaN (Not a Number).

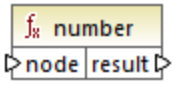


Sprachen

XQuery, XSLT 2.0, XSLT 3.0..

5.7.6.17.9 number

Gibt den in einen `xs:double`-Typ konvertierten Wert von **node** zurück. Die einzigen Typen, die in Zahlen konvertiert werden können, sind Boolesche Werte, numerische Strings und andere numerische Typen. Nicht numerische Input-Werte (wie z.B. ein nicht numerischer String) führen zum Resultat NaN (Not a Number).



Sprachen

XQuery, XSLT 2.0, XSLT 3.0..

© 2019-2025 Altova GmbH

Parameter

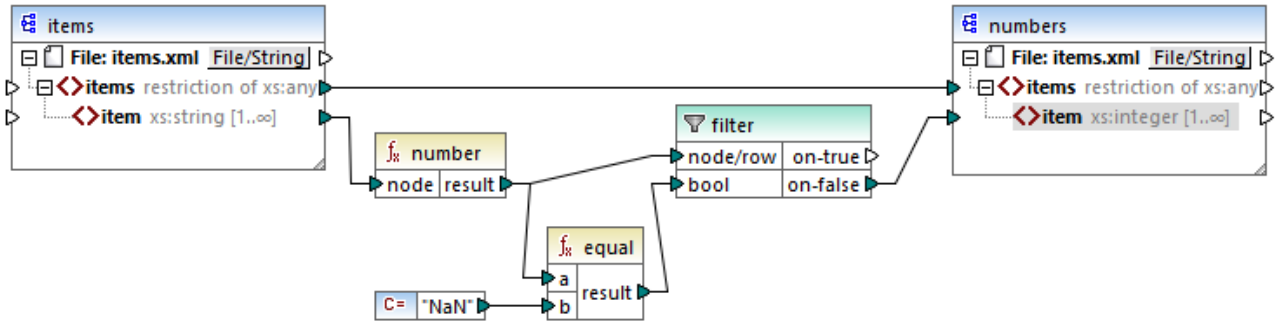
Name	Typ	Beschreibung
node	`mf:atomic`	Der Input-Node.

Beispiel

Die folgende XML-Datei enthält Datenelemente vom Typ `string`:

```
<items>
   <item>1</item>
   <item>2</item>
   <item>Jingle Bells</item>
</items>
```

Im unten gezeigten Mapping wird versucht, alle diese Strings in numerische Werte zu konvertieren und diese in eine XML-Zieldatei zu schreiben. Beachten Sie, dass der Datentyp von **item** in der XML-Zielkomponente `xs:integer` ist, während das Quelldatenelement **item** den Datentyp `xs:string` hat. Wenn die Konvertierung nicht erfolgreich war, muss das Datenelement übersprungen werden und darf nicht in die Zieldatei kopiert werden.



XSLT 2.0-Mapping

Um das gewünschte Mapping-Resultat zu erhalten, wurde ein Filter verwendet. Mit der Funktion **`equal`** wird überprüft, ob das Ergebnis der Konvertierung "NaN" ist. Wenn dies false ist, weist dies darauf hin, dass die Konvertierung erfolgreich war. Daher wird das Datenelement in die Zielkomponente kopiert. Des Ergebnis des Mappings sieht folgendermaßen aus:

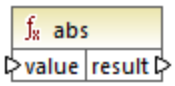
```
<items>
   <item>1</item>
   <item>2</item>
</items>
```

5.7.6.18 xpath2 | numeric functions

Zu den numerischen Funktionen der **xpath2**-Bibliothek gehören die Funktionen **abs** und **round-half-to-even**.

5.7.6.18.1 abs

Gibt den absoluten Wert des Arguments zurück. Wenn das Input-Argument z.B. **-2** oder **2** ist, gibt die Funktion **2** zurück.



Sprachen

XQuery, XSLT 2.0, XSLT 3.0..

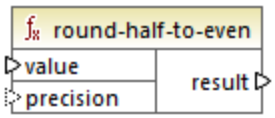
Parameter

Name	Typ	Beschreibung
value	`xs:decimal`	Der Input-Wert.

5.7.6.18.2 round-half-to-even

Die **round-half-to-even** Funktion rundet die bereitgestellte Zahl (das erste Argument) auf den Präzisionsgrad (Anzahl der Dezimalstellen) auf bzw. ab, der im optionalen zweiten Argument definiert ist. Wenn z.B. das erste Argument **2,141567** und das zweite Argument **3** ist, dann wird das erste Argument (die Zahl) auf drei Dezimalstellen gerundet, d.h. das Ergebnis ist **2,142**. Wenn kein Präzisionsgrad (zweites Argument) angegeben ist, wird die Zahl auf null Dezimalstellen, also eine Ganzzahl gerundet.

"even" im Funktionsnamen bezieht sich auf die Rundung auf eine gerade Zahl, wenn eine Ziffer in einer Zahl sich genau in der Mitte zwischen zwei Werten befindet. `round-half-to-even(3,475, 2)` ergäbe z.B. **3,48**.



Sprachen

XQuery, XSLT 2.0, XSLT 3.0..

© 2019-2025 Altova GmbH

Parameter

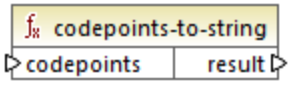
Name	Typ	Beschreibung
value	`xs:decimal`	Obligatorisches Argument, das den zu rundenden Input-Wert bereitstellt.
precision	`xs:integer`	Optionales Argument, das die Anzahl der Dezimalstellen angibt, auf die gerundet werden soll. Der Standardwert ist **0**.

5.7.6.19 xpath2 | string functions (String-Funktionen)

Mit Hilfe der String-Funktionen der **xpath2**-Bibliothek können Sie Strings verarbeiten (dazu gehören der Vergleich von Strings, die Konvertierung von Strings in Groß- und Kleinbuchstaben, die Extraktion von Substrings aus Strings und andere).

5.7.6.19.1 codepoints-to-string

Erstellt anhand einer Sequenz von Unicode-Codepoints einen String. Diese Funktoin ist das Gegenteil der Funktion `string-to-codepoints` 380.



Sprachen

XQuery, XSLT 2.0, XSLT 3.0..

Parameter

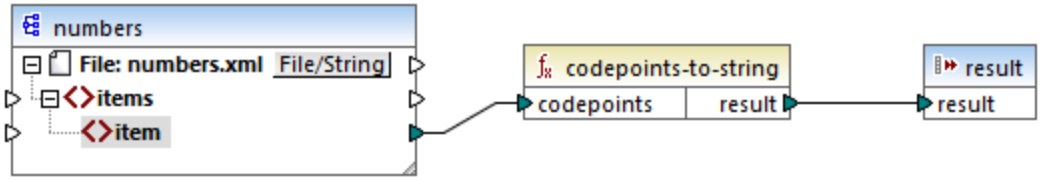
Name	Typ	Beschreibung
codepoints	`ZeroOrMore xs:integer`	Dieser Input muss mit einer Sequenz von Datenelementen vom Typ Ganzzahl verbunden werden, wobei jede Ganzzahl einen Unicode-Codepoint definiert.

Beispiel

Die folgende XML-Datei enthält mehrere **item**-Elemente, in denen jeweils Unicode-Codepoint-Werte gespeichert sind.

```
<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>
```

Im unten gezeigten Mapping wird die Sequenz von Datenelementen für die Funktion **codepoint-to-string** bereitgestellt.



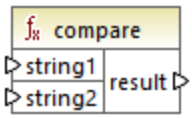
XSLT 2.0-Mapping

Die Mapping-Ausgabe ist **MapForce**.

5.7.6.19.2 compare

Die **compare**-Funktion erhält zwei Strings als Argumente und vergleicht diese alphabetisch und überprüft, ob diese identisch sind. Wenn **string1** im Alphabet vor **string2** (z.B. bei zwei Strings A und B) vorkommt, dann gibt die Funktion **-1** zurück. Wenn die beiden Strings gleich sind (z.B. A und A), gibt die Funktion **0** zurück. Wenn **string1** im Alphabet nach **string2** (z.B. bei zwei Strings B und A) vorkommt, dann gibt die Funktion **1** zurück.

In dieser Variante der Funktion wird die Standard-Collation, also Unicode verwendet. Es gibt eine weitere Variante[373] dieser Funktion, bei der Sie die Collation als Argument angeben können.



Sprachen

XQuery, XSLT 2.0, XSLT 3.0..

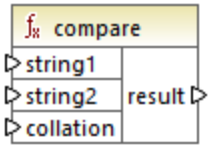
© 2019-2025 Altova GmbH

Parameter

Name	Typ	Beschreibung
string1	`xs:string`	Der erste Input-String.
string2	`xs:string`	Der zweite Input-String.

5.7.6.19.3 compare

Die `compare`-Funktion erhält zwei Strings als Argumente und vergleicht diese unter Verwendung der als Argument bereitgestellten Collation alphabetisch und überprüft, ob diese identisch sind. Wenn **string1** im Alphabet vor **string2** (z.B. bei zwei Strings A und B) vorkommt, dann gibt die Funktion **-1** zurück. Wenn die beiden Strings gleich sind (z.B. A und A), gibt die Funktion **0** zurück. Wenn **string1** im Alphabet nach **string2** (z.B. bei zwei Strings B und A) vorkommt, dann gibt die Funktion **1** zurück.



Sprachen

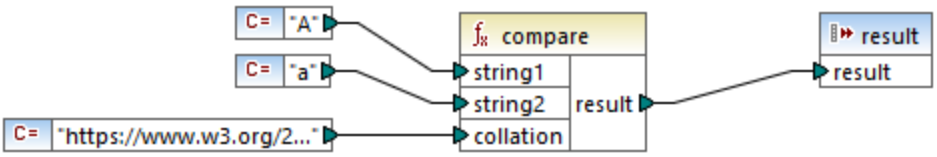
XQuery, XSLT 2.0, XSLT 3.0..

Parameter

Name	Typ	Beschreibung
string1	`xs:string`	Der erste Input-String.
string2	`xs:string`	Der zweite Input-String.
collation	`xs:string`	Definiert, welche Collation für den String-Vergleich verwendet werden soll. Dieser Input kann aus der Ausgabe der Funktion **`default-collation`** 345 stammen oder kann eine Collation wie z.B. `http://www.w3.org/2005/xpath-functions/collation/html-ascii-case-insensitive` sein.

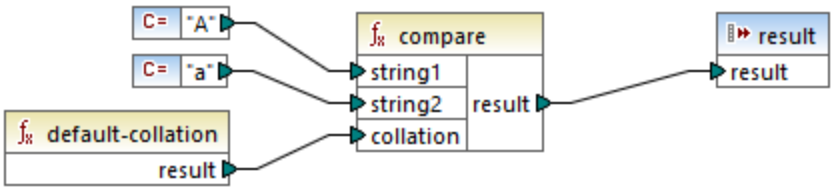
Beispiel

Im folgenden Mapping werden die Strings "A" und "a" mit Hilfe der durch eine Konstante bereitgestellten Collation `http://www.w3.org/2005/xpath-functions/collation/html-ascii-case-insensitive`, in der die Groß- und Kleinschreibung keine Rolle spielt, verglichen.



XSLT 2.0 Mapping

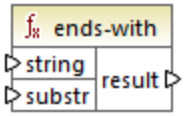
Das Ergebnis des obigen Mappings ist **0**, was bedeutet, dass beide Strings als identisch betrachtet werden. Wenn Sie die Collation jedoch durch die durch die Funktion **`default-collation`** bereitgestellte Collation ersetzen, wird stattdessen die Standard-Unicode Codepoint Collation verwendet und das Ergebnis des Mappings ist **-1** ("A" kommt im Alphabet vor "a").



5.7.6.19.4 ends-with

Gibt **true** zurück, wenn **string** mit **substr** endet; gibt andernfalls **false** zurück. Der Rückgabewert hat den Typ `xs:boolean`.

In dieser Variante der Funktion wird die Standard-Collation, also Unicode verwendet. Es gibt eine weitere Variante[375] dieser Funktion, bei der Sie die Collation als Argument angeben können.



Sprachen

XQuery, XSLT 2.0, XSLT 3.0..

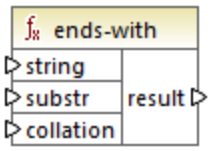
© 2019-2025 Altova GmbH

Parameter

Name	Typ	Beschreibung
string	`xs:string`	Der Input-String (d.h. der "Heuhaufen").
substr	`xs:string`	Der Substring (d.h. die "Nadel").

5.7.6.19.5 ends-with

Gibt **true** zurück, wenn **string** mit **substr** endet; gibt andernfalls **false** zurück. Der Rückgabewert hat den Typ `xs:boolean`.



Sprachen

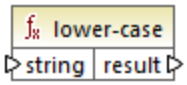
XQuery, XSLT 2.0, XSLT 3.0..

Parameter

Name	Typ	Beschreibung
string	`xs:string`	Der Input-String (d.h. der "Heuhaufen").
substr	`xs:string`	Der Substring (d.h. die "Nadel").
collation	`xs:string`	Definiert, welche Collation für den String-Vergleich verwendet werden soll. Dieser Input kann aus der Ausgabe der Funktion **`default-collation`** 345 stammen oder kann eine Collation wie z.B. `http://www.w3.org/2005/xpath-functions/collation/html-ascii-case-insensitive` sein.

5.7.6.19.6 lower-case

Gibt den Wert von **string** nach Konvertierung der einzelnen Zeichen in Kleinbuchstaben zurück.



Sprachen

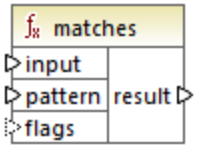
XQuery, XSLT 2.0, XSLT 3.0..

Parameter

Name	Typ	Beschreibung
string	`xs:string`	Der Input-Wert.

5.7.6.19.7 matches

Die `matches` Funktion überprüft, ob ein bereitgestellter String (das erste Argument) einer Regular Expression (dem zweiten Argument) entspricht. Die Syntax der Regular Expression muss die Syntax sein, die für das `pattern` Facet von XML-Schema definiert wurde. Die Funktion gibt **true** zurück, wenn der String der Regular Expression entspricht, und andernfalls **false**.



Sprachen

XQuery, XSLT 2.0, XSLT 3.0..

Parameter

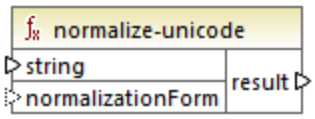
Name	Typ	Beschreibung
input	`xs:string`	Der Input-String.
pattern	`xs:string`	Die Regular Expression, der der String entsprechen muss, siehe Regular Expressions 247.
flags	`xs:string`	Optionales Argument, das die Übereinstimmung beeinflusst. In diesem Argument kann jede beliebige Kombination der folgenden Flags angegeben werden: `i`, `m`, `s`, `x`. Es können mehrere Flags

© 2019-2025 Altova GmbH

Name	Typ	Beschreibung
		verwendet werden, z.B. `imx`. Wenn kein Flag verwendet wird, werden die Standardwerte aller vier Flags verwendet. Es gibt die folgenden vier Flags: `i` Modus "Groß/Kleinschreibung wird nicht berücksichtigt" verwenden. Die Standardeinstellung ist "Groß-/Kleinschreibung berücksichtigen". `m` Mehrzeiligen Modus verwenden. In diesem Modus wird der Input-String als mehrzeilig betrachtet, wobei jede Zeile durch ein newline-Zeichen (`x0a`) getrennt wird. Die Metazeichen **^** und **$** kennzeichnen den Beginn und das Ende der einzelnen Zeilen. Die Standardeinstellung ist der String-Modus, in dem der String mit den Metazeichen **^** und **$** beginnt und endet. `s` dot-all Modus verwenden. Die Standardeinstellung ist der not-dot-all Modus, in dem das Metazeichen **.** für alle Zeichen mit Ausnahme des newline-Zeichens (`x0a`) steht. Im dot-all Modus steht der Punkt auch für das newline-Zeichen. `x` Whitespaces ignorieren. Standardmäßig werden Whitespace-Zeichen nicht ignoriert.

5.7.6.19.8 normalize-unicode

Gibt den Wert von **string** zurück, der entsprechend den Regeln des definierten Normalisierungsprotokolls (zweites Argument) normalisiert ist. Nähere Informationen zur Unicode-Normalisierung siehe §2.2 von https://www.w3.org/TR/charmod-norm/.



Sprachen

XQuery, XSLT 2.0, XSLT 3.0..

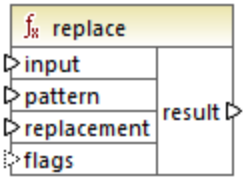
Parameter

Name	Typ	Beschreibung
string	`xs:string`	Der zu normalisierende Stringwert.

Name	Typ	Beschreibung
normalizationForm	`xs:string`	Optionales Argument, das das Normalisierungsprotokoll angibt. Das Standardprotokoll ist Unicode Normalization Form C (NFC). Es werden die Normalisierungsprotokolle NFC, NFD, NFKC und NFKD unterstützt.

5.7.6.19.9 replace

Diese Funktion erhält einen Input-String, eine Regular Expression und einen Ersetzungsstring als Argumente. Sie ersetzt alle Übereinstimmungen mit der Regular Expression im Input-String durch den Ersetzungsstring. Wenn zwei einander überlappende Strings im Input-String mit der Regular Expression übereinstimmen, wird nur die erste Entsprechung ersetzt.



Sprachen

XQuery, XSLT 2.0, XSLT 3.0..

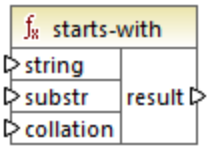
Parameter

Name	Typ	Beschreibung
input	`xs:string`	Der Input-String.
pattern	`xs:string`	Die Regular Expression, der der String entsprechen muss, siehe Regular Expressions 247.
replacement	`xs:string`	Der Ersetzungsstring.
flags	`xs:string`	Optionales Argument, das die Übereinstimmung beeinflusst. Dieses Argument wird auf dieselbe Art, wie das Argument **flags** in der Funktion `matches` 376 verwendet.

© 2019-2025 Altova GmbH

5.7.6.19.10 starts-with

Gibt **true** zurück, wenn **string** mit **substr** beginnt; gibt andernfalls **false** zurück. Der Rückgabewert hat den Typ `xs:boolean`. Der String-Vergleich erfolgt gemäß der angegebenen Collation.



Sprachen

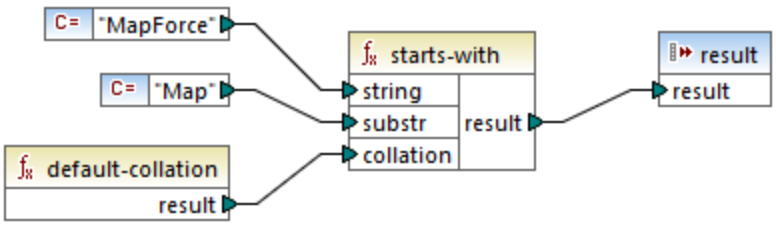
XQuery, XSLT 2.0, XSLT 3.0..

Parameter

Name	Typ	Beschreibung
string	`xs:string`	Der Input-String (d.h. der "Heuhaufen").
substr	`xs:string`	Der Substring (d.h. die "Nadel").
collation	`xs:string`	Definiert, welche Collation für den String-Vergleich verwendet werden soll. Dieser Input kann aus der Ausgabe der Funktion **default-collation** 345 stammen oder kann eine Collation wie z.B. `http://www.w3.org/2005/xpath-functions/collation/html-ascii-case-insensitive` sein.

Beispiel

Im folgenden Mapping wird der Wert **true** zurückgegeben , da der Input-String "MapForce" mit dem Substring "Map" beginnt, vorausgesetzt, es wird die Standard-Unicode Collation verwendet.



5.7.6.19.11 string-to-codepoints

Gibt die Sequenz von Unicode Codepoints (Ganzzahlwerte), die den als Argument bereitgestellten String darstellen, zurück. Diese Funktoin ist das Gegenteil der Funktion **`codepoints-to-string`** 371.

Sprachen

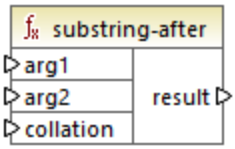
XQuery, XSLT 2.0, XSLT 3.0..

Parameter

Name	Typ	Beschreibung
input	`xs:string`	Der Input-String

5.7.6.19.12 substring-after

Gibt den Teil des String **arg1** zurück, der nach dem String **arg2** kommt.



Sprachen

XQuery, XSLT 2.0, XSLT 3.0..

Parameter

Name	Typ	Beschreibung
arg1	`xs:string`	Der Input-String (d.h. der "Heuhaufen").
arg2	`xs:string`	Der Substring (d.h. die "Nadel").
collation	`xs:string`	Definiert, welche Collation für den String-Vergleich verwendet werden soll. Dieser Input kann aus der Ausgabe der Funktion **`default-`**

 © 2019-2025 Altova GmbH

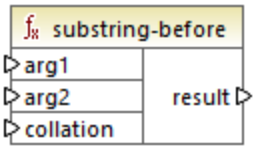
Name	Typ	Beschreibung
		`collation`[345] stammen oder kann eine Collation wie z.B. `http://www.w3.org/2005/xpath-functions/collation/html-ascii-case-insensitive` sein.

Beispiel

Wenn **arg1** "MapForce" ist, **arg2** "Map" und **collation** die `default-collation`[345] ist, gibt die Funktion "Force" zurück.

5.7.6.19.13 substring-before

Gibt den Teil des String **arg1** zurück, der vor dem String **arg2** kommt.



Sprachen

XQuery, XSLT 2.0, XSLT 3.0..

Parameter

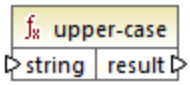
Name	Typ	Beschreibung
arg1	`xs:string`	Der Input-String (d.h. der "Heuhaufen").
arg2	`xs:string`	Der Substring (d.h. die "Nadel").
collation	`xs:string`	Definiert, welche Collation für den String-Vergleich verwendet werden soll. Dieser Input kann aus der Ausgabe der Funktion `default-collation`[345] stammen oder kann eine Collation wie z.B. `http://www.w3.org/2005/xpath-functions/collation/html-ascii-case-insensitive` sein.

Beispiel

Wenn **arg1** "MapForce" ist, **arg2** "Force" und **collation** die `default-collation` 345 ist, gibt die Funktion "Map" zurück.

5.7.6.19.14 upper-case

Gibt den Wert von **string** nach Konvertierung der einzelnen Zeichen in Großbuchstaben zurück.



Sprachen

XQuery, XSLT 2.0, XSLT 3.0..

Parameter

Name	Typ	Beschreibung
string	`xs:string`	Der Input-String.

5.7.6.20 xpath3 | external information functions

Mit Hilfe der Funktionen für externe Informationen der **xpath3**-Bibliothek können Sie Informationen über die XSLT-Ausführungsumgebung aufrufen oder Daten aus externen Ressourcen abrufen.

5.7.6.20.1 available-environment-variables

Gibt eine Liste von Umgebungsvariablennamen zurück, die sich für die Übergabe an die `environment-variable`-Funktion in Form einer (möglicherweise leeren) Stringsequenz eignen.

Sprachen

XSLT 3.0.

© 2019-2025 Altova GmbH

5.7.6.20.2 environment-variable

Gibt den Wert einer Systemumgebungsvariablen, falls vorhanden, zurück. Der Rückgabetyp ist `xs:string`.

Sprachen

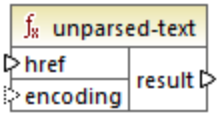
XSLT 3.0.

Parameter

Name	Typ	Beschreibung
name	`xs:string`	Der Name der Umgebungsvariablen.

5.7.6.20.3 unparsed-text

Liest eine externe Ressource (z.B. eine Datei) und gibt eine String-Darstellung der Ressource zurück.



Sprachen

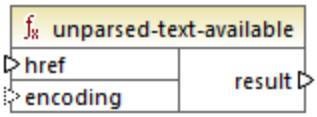
XSLT 3.0.

Parameter

Name	Typ	Beschreibung
href	`xs:string`	Ein String in der Form einer URI-Referenz.
encoding	`xs:string`	Optionales Argument. Definiert den Namen der Kodierung, z.B. "UTF-8", "UTF-16". Wenn die Kodierung nicht ermittelt werden kann, wird automatisch UTF-8 angenommen.

5.7.6.20.4 unparsed-text-available

Stellt fest, ob ein Aufruf von **`unparsed-text`** mit bestimmten Argumenten erfolgreich wäre. Der Rückgabetyp ist `xs: boolean`.



Sprachen

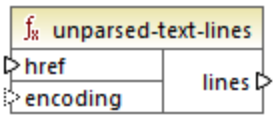
XSLT 3.0.

Parameter

Name	Typ	Beschreibung
href	`xs:string`	Ein String in der Form einer URI-Referenz.
encoding	`xs:string`	Optionales Argument. Definiert den Namen der Kodierung, z.B. "UTF-8", "UTF-16". Wenn die Kodierung nicht ermittelt werden kann, wird automatisch UTF-8 angenommen.

5.7.6.20.5 unparsed-text-lines

Liest eine externe Ressource (z.B. eine Datei) und gibt ihren Inhalt als String-Sequenz zurück, und zwar eine für jede Textzeile in der String-Darstellung der Ressource.



Sprachen

XSLT 3.0.

© 2019-2025 Altova GmbH

Parameter

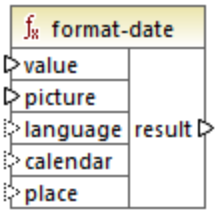
Name	Typ	Beschreibung
href	`xs:string`	Ein String in der Form einer URI-Referenz.
encoding	`xs:string`	Optionales Argument. Definiert den Namen der Kodierung, z.B. "UTF-8", "UTF-16". Wenn die Kodierung nicht ermittelt werden kann, wird automatisch UTF-8 angenommen.

5.7.6.21 xpath3 | formatting functions

Die Formatierungsfunktionen der **xpath3**-Bibliothek dienen zum Formatieren von Datums-, Uhrzeit und Ganzzahlwerten.

5.7.6.21.1 format-date

Gibt einen String zurück, der einen für die Anzeige formatierten `xs:date`-Wert enthält.



Sprachen

XSLT 3.0.

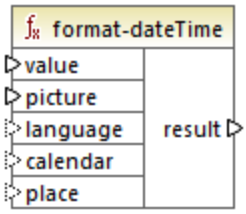
Parameter

Name	Typ	Beschreibung
value	`xs:date`	Der zu formatierende `xs:date`-Input-Wert. Obligatorischer Parameter.
picture	`xs:string`	Obligatorischer Parameter.

Name	Typ	Beschreibung
		Siehe Abschnitt 9.8.4.1 der "XPath and XQuery Functions and Operators 3.1" W3C Recommendation (https://www.w3.org/TR/xpath-functions-31).
language	`xs:string`	Optionaler Parameter. Siehe Abschnitt 9.8.4.8 der "XPath and XQuery Functions and Operators 3.1" W3C Recommendation (https://www.w3.org/TR/xpath-functions-31).
calendar	`xs:string`	Wie oben.
place	`xs:string`	Wie oben.

5.7.6.21.2 format-dateTime

Gibt einen String zurück, der einen für die Anzeige formatierten `xs:dateTime`-Wert enthält.



Sprachen

XSLT 3.0.

Parameter

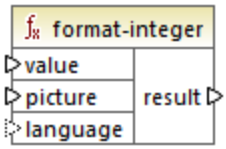
Name	Typ	Beschreibung
value	`xs:dateTime`	Der zu formatierende `xs:dateTime`-Input-Wert.
picture	`xs:string`	Obligatorischer Parameter. Siehe Abschnitt 9.8.4.1 der "XPath and XQuery Functions and

 © 2019-2025 Altova GmbH

Name	Typ	Beschreibung
		Operators 3.1" W3C Recommendation (https://www.w3.org/TR/xpath-functions-31).
language	`xs:string`	Optionaler Parameter. Siehe Abschnitt 9.8.4.8 der "XPath and XQuery Functions and Operators 3.1" W3C Recommendation (https://www.w3.org/TR/xpath-functions-31).
calendar	`xs:string`	Wie oben.
place	`xs:string`	Wie oben.

5.7.6.21.3 format-integer

Formatiert eine Ganzzahl anhand der Konventionen einer angegebenen natürlichen Sprache (falls angegeben) gemäß einem angegebenen picture-String.



Sprachen

XSLT 3.0.

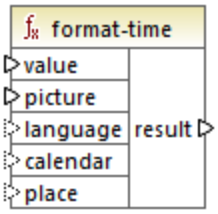
Parameter

Name	Typ	Beschreibung
value	`xs:integer`	Der zu formatierende Integer-Input-Wert.
picture	`xs:string`	Obligatorischer Parameter. Siehe Abschnitt 4.6.1 der "XPath and XQuery Functions and Operators 3.1" W3C Recommendation

Name	Typ	Beschreibung
		(https://www.w3.org/TR/xpath-functions-31).
language	`xs:string`	Optionaler Parameter. Definiert die natürliche Sprache, gemäß der der Wert formatiert werden soll. Falls angegeben, muss es sich bei diesem Wert entweder um eine leeren String oder einen beliebigen für das `xml:lang`-Attribut gemäß der "Extensible Markup Language (XML) 1.0 W3C Recommendation (https://www.w3.org/TR/xml) zulässigen Wert handeln.

5.7.6.21.4 format-time

Gibt einen String zurück, der einen für die Anzeige formatierten `xs:time`-Wert enthält.



Sprachen

XSLT 3.0.

Parameter

Name	Typ	Beschreibung
value	`xs:time`	Der zu formatierende `xs:time`-Input-Wert.
picture	`xs:string`	Obligatorischer Parameter. Siehe Abschnitt 9.8.4.1 der "XPath and XQuery Functions and Operators 3.1" W3C Recommendation

 © 2019-2025 Altova GmbH

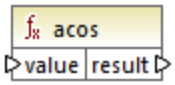
Name	Typ	Beschreibung
		(https://www.w3.org/TR/xpath-functions-31).
language	`xs:string`	Optionaler Parameter. Siehe Abschnitt 9.8.4.8 der "XPath and XQuery Functions and Operators 3.1" W3C Recommendation (https://www.w3.org/TR/xpath-functions-31).
calendar	`xs:string`	Wie oben.
place	`xs:string`	Wie oben.

5.7.6.22 xpath3 | math functions

Mit Hilfe der math-Funktionen der **xpath3**-Bibliothek können Sie trigonometrische und andere mathematische Berechnungen durchführen.

5.7.6.22.1 acos

Gibt den Arkuscosinus eines Winkels im Bereich von **0** bis **pi** zurück.



Sprachen

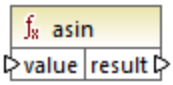
XSLT 3.0.

Parameter

Name	Typ	Beschreibung
value	`xs:double`	Der Input-Wert.

5.7.6.22.2 asin

Gibt den ArkusSinus eines Winkels im Bereich von **-pi/2** bis **pi/2** zurück.



Sprachen

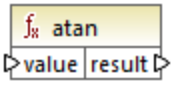
XSLT 3.0.

Parameter

Name	Typ	Beschreibung
value	`xs:double`	Der Input-Wert.

5.7.6.22.3 atan

Gibt den Arkustangens eines Winkels im Bereich von **-pi/2** bis **pi/2** zurück.



Sprachen

XSLT 3.0.

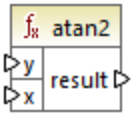
Parameter

Name	Typ	Beschreibung
value	`xs:double`	Der Input-Wert.

5.7.6.22.4 atan2

Gibt den Winkel in Bogenmaß zwischen einem Punkt auf einer Fläche mit den Koordinaten (x,y) und der positiven X-Achse zurück.

© 2019-2025 Altova GmbH



Sprachen

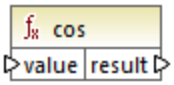
XSLT 3.0.

Parameter

Name	Typ	Beschreibung
y	`xs:double`	Die x-Koordinate.
x	`xs:double`	The y-Koordinate.

5.7.6.22.5 cos

Gibt den trigonometrischen Cosinus des durch value angegebenen Winkels zurück. Die Werteinheit ist radians.



Sprachen

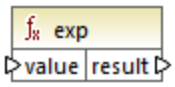
XSLT 3.0.

Parameter

Name	Typ	Beschreibung
value	`xs:double`	Der Input-Wert.

5.7.6.22.6 exp

Gibt die Eulersche Zahl *e* hoch value zurück.



Sprachen

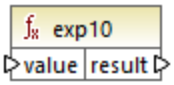
XSLT 3.0.

Parameter

Name	Typ	Beschreibung
value	`xs:double`	Der Input-Wert.

5.7.6.22.7 exp10

Gibt 10 hoch value zurück.



Sprachen

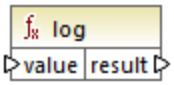
XSLT 3.0.

Parameter

Name	Typ	Beschreibung
value	`xs:double`	Der Input-Wert.

5.7.6.22.8 log

Gibt den natürlichen Logarithmus (Basis e) eines Werts zurück.



Sprachen

XSLT 3.0.

Parameter

Name	Typ	Beschreibung
value	`xs:double`	Der Input-Wert.

© 2019-2025 Altova GmbH

5.7.6.22.9 log10

Gibt den dekadischen Logarithmus (Basis 10) eines Werts zurück.

Sprachen

XSLT 3.0.

Parameter

Name	Typ	Beschreibung
value	`xs:double`	Der Input-Wert.

5.7.6.22.10 pi

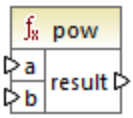
Gibt einen Näherungswert an die mathematische Konstante **Pi** zurück.

Sprachen

XSLT 3.0.

5.7.6.22.11 pow

Gibt den Wert von **a** hoch **b** zurück.



Sprachen

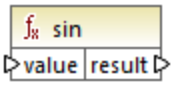
XSLT 3.0.

Parameter

Name	Typ	Beschreibung
a	`xs:double`	Der Input-Wert **a**.
b	`xs:double`	Der Input-Wert **b**.

5.7.6.22.12 sin

Gibt den trigonometrischen Sinus des durch value angegebenen Winkels zurück. Die Werteinheit ist Radians.



Sprachen

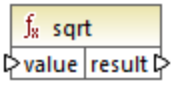
XSLT 3.0.

Parameter

Name	Typ	Beschreibung
value	`xs:double`	Der Input-Wert.

5.7.6.22.13 sqrt

Gibt die nicht negative Quadratwurzel des Arguments zurück.



Sprachen

XSLT 3.0.

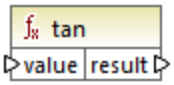
Parameter

Name	Typ	Beschreibung
value	`xs:double`	Der Input-Wert.

© 2019-2025 Altova GmbH

5.7.6.22.14 tan

Gibt den trigonometrischen Tangens des durch value angegebenen Winkels zurück. Die Werteinheit ist Radians.



Sprachen

XSLT 3.0.

Parameter

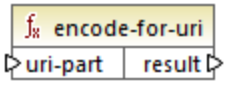
Name	Typ	Beschreibung
value	`xs:double`	Der Input-Wert.

5.7.6.23 xpath3 | URI functions

Die URI-Funktionen in der **xpath3**-Bibliothek dienen zum Kodieren, mit Escape-Zeichen Versehen und Konvertieren von Werte, die in URIs zur Verwendung kommen sollen.

5.7.6.23.1 encode-for-uri

Kodiert reservierte Zeichen in einem String, die im Pfadsegment einer URI verwendet werden sollen. Nähere Informationen zu dieser Funktion finden Sie im Abschnitt 6.2 der "XPath and XQuery Functions and Operators 3.1" W3C Recommendation (https://www.w3.org/TR/xpath-functions-31).



Sprachen

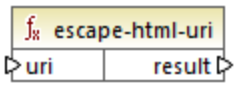
XSLT 3.0.

Parameter

Name	Typ	Beschreibung
uri-part	`xs:string`	Der zu kodierende URI-Input-Wert.

5.7.6.23.2 escape-html-uri

Versieht eine URI auf dieselbe Art, auf die auch HTML-Benutzer-Agenten Attributwerte behandeln, die wahrscheinlich URIs enthalten, mit Escape-Zeichen. Nähere Informationen zu dieser Funktion finden Sie im Abschnitt 6.4 der "XPath and XQuery Functions and Operators 3.1" W3C Recommendation (https://www.w3.org/TR/xpath-functions-31).



Sprachen

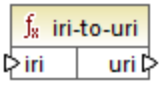
XSLT 3.0.

Parameter

Name	Typ	Beschreibung
uri	`xs:string`	Der mit Escape-Zeichen zu versehende URI-Input-Wert.

5.7.6.23.3 iri-to-uri

Konvertiert einen String, der einen IRI (Internationalized Resource Identifier) enthält in eine URI (Uniform Resource Identifier). Nähere Informationen zu dieser Funktion finden Sie im Abschnitt 6.3 der "XPath and XQuery Functions and Operators 3.1" W3C Recommendation (https://www.w3.org/TR/xpath-functions-31).



Sprachen

XSLT 3.0.

Parameter

Name	Typ	Beschreibung
iri	`xs:string`	Der IRI-Input-Wert.

© 2019-2025 Altova GmbH

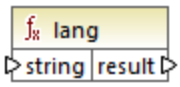
5.7.6.24 xslt | xpath functions

Die Funktionen in dieser Untergruppe sind XPath 1.0-Funktionen, die Informationen über Mapping-Datenelemente (oder Nodes) zurückgeben. Die meisten dieser Funktionen erhalten einen Node als Argument und geben Informationen über diesen Node zurück. Die Funktionen **last** und **position** werden im aktuellen Mapping-Kontext 79, der durch die Verbindungen in Ihrem Mapping bestimmt wird, ausgeführt.

Anmerkung: Weitere XPath 1.0-Funktionen finden Sie in der **core**-Bibliothek.

5.7.6.24.1 lang

Gibt **true** zurück, wenn der Kontext-Node ein `xml:lang`-Attribut mit einem Wert hat, der entweder genau mit dem Argument **string** übereinstimmt oder eine Untergruppe davon ist. Andernfalls gibt die Funktion **false** zurück.



Sprachen

XSLT 1.0.

Parameter

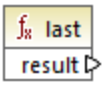
Name	Typ	Beschreibung
string	`xs:string`	Der zu überprüfende Sprachencode, z.B. "en".

Beispiel

Siehe das Beispiel unter der Funktion **lang** 363 der **xpath2**-Bibliothek.

5.7.6.24.2 last

Gibt die Position des letzten Node in der Liste der verarbeiteten Nodes zurück.



Sprachen

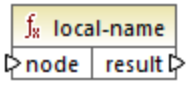
XSLT 1.0.

Beispiel

Siehe das Beispiel unter der Funktion `last` 346 der **xpath2**-Bibliothek.

5.7.6.24.3 local-name

Gibt den lokalen Teil des Namens des als Argument bereitgestellten Node zurück.



Sprachen

XSLT 1.0, XSLT 2.0, XSLT 3.0.

Parameter

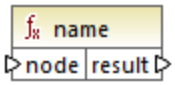
Name	Typ	Beschreibung
node	`node()`	Der Input-Node.

Beispiel

Siehe das Beispiel unter der Funktion `local-name` 365 der **xpath2**-Bibliothek.

5.7.6.24.4 name

Gibt den Namen des als Argument bereitgestellten Node zurück.



Sprachen

XSLT 1.0, XSLT 2.0, XSLT 3.0.

Parameter

Name	Typ	Beschreibung
node	`node()`	Der Input-Node.

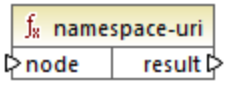
© 2019-2025 Altova GmbH

Beispiel

Siehe das Beispiel unter der Funktion `local-name` 365 der **xpath2**-Bibliothek.

5.7.6.24.5 namespace-uri

Gibt die Namespace URI des als Argument bereitgestellten Node zurück.



Sprachen

XSLT 1.0, XSLT 2.0, XSLT 3.0.

Parameter

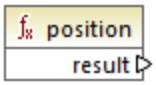
Name	Typ	Beschreibung
node	`node()`	Der Input-Node.

Beispiel

Siehe das Beispiel unter der Funktion `local-name` 365 der **xpath2**-Bibliothek.

5.7.6.24.6 position

Gibt die Position des aktuellen Node im gerade verarbeiteten Nodeset zurück.



Sprachen

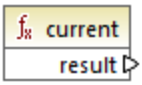
XSLT 1.0.

5.7.6.25 xslt | xslt function (XSLT-Funktionen)

Bei den Funktionen in dieser Gruppe handelt es sich um diverse XSLT 1.0-Funktionen.

5.7.6.25.1 current

Die **current**-Funktion erhält kein Argument und gibt den aktuellen Node zurück.

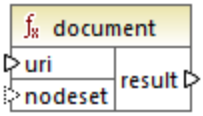


Sprachen

XSLT 1.0.

5.7.6.25.2 document

Ruft Nodes aus einem externen XML-Dokument auf. Das Ergebnis wird in einen Node im Ausgabedokument ausgegeben.



Sprachen

XSLT 1.0.

Parameter

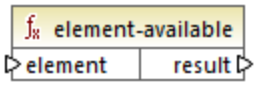
Name	Typ	Beschreibung
uri	`xs:string`	Obligatorisch. Definiert den Pfad zum XML-Dokument. Das XML-Dokument muss gültig und parsebar sein.
nodeset	`node()`	Optional. Definiert einen Node, anhand dessen Basis-URI die URI aufgelöst wird, die als erstes Argument bereitgestellt wird, wenn diese URI relativ ist.

5.7.6.25.3 element-available

Die **element-available** Funktion überprüft, ob ein Element, das als das einzige String-Argument der Funktion bereitgestellt wird, vom XSLT-Prozessor unterstützt wird. Der Argument-String wird als QName ausgewertet. Daher müssen XSLT-Elemente ein `xsl:`-Präfix und XML-Schema-Elemente ein `xs:`-Präfix haben, da dies die

 © 2019-2025 Altova GmbH

Präfixe sind, die im zugrunde liegenden XSLT-Dokument, das für das Mapping erstellt wird, deklariert sind. Die Funktion gibt einen Booleschen Wert zurück.



Sprachen

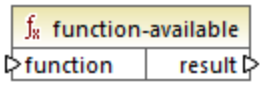
XSLT 1.0.

Parameter

Name	Typ	Beschreibung
element	`xs:string`	Der Elementname.

5.7.6.25.4 function-available

Die **`function-available`**-Funktion ähnelt der **`element-available`**-Funktion und überprüft, ob der als Argument der Funktion bereitgestellte Funktionsname vom XSLT-Prozessor unterstützt wird. Der Input-String wird als QName ausgewertet. Die Funktion gibt einen Booleschen Wert zurück.



Sprachen

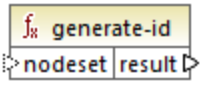
XSLT 1.0.

Parameter

Name	Typ	Beschreibung
function	`xs:string`	Der Funktionsname.

5.7.6.25.5 generate-id

Die **`generate-id`**-Funktion generiert einen eindeutigen String, der den ersten Node des Nodeset anhand des optionalen Input-Arguments identifiziert. Wenn kein Argument bereitgestellt wird, wird die ID am Kontext-Node generiert. Das Ergebnis kann an jeden Node im Ausgabedokument gerichtet werden.



Sprachen

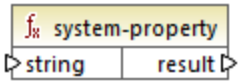
XSLT 1.0, XSLT 2.0, XSLT 3.0.

Parameter

Name	Typ	Beschreibung
nodeset	`node()`	Optionales Argument, das den Input-Node bereitstellt.

5.7.6.25.6 system-property

Die `system-property`-Funktion gibt die Eigenschaften des XSLT-Prozessors (des Systems) zurück. Drei Systemeigenschaften, alle im XSLT-Namespace, sind bei XSLT-Prozessoren obligatorisch, nämlich `xsl:version`, `xsl:vendor` und `xsl:vendor-url`. Der Input-String wird als QName ausgewertet und muss daher das Präfix `xsl:` haben, da dies das Präfix ist, das im zugrunde liegenden XSLT-Stylesheet mit dem XSLT-Namespace verknüpft ist.



Sprachen

XSLT 1.0, XSLT 2.0, XSLT 3.0.

Parameter

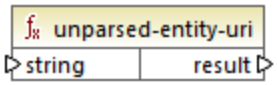
Name	Typ	Beschreibung
string	`xs:string`	Definiert den Eigenschaftsnamen. Dieser kann einer der folgenden sein: `xsl:version`, `xsl:vendor`, `xsl:vendor-url`.

5.7.6.25.7 unparsed-entity-uri

Wenn Sie eine DTD verwenden, können Sie darin eine ungeparste Entity deklarieren. Diese ungeparste Entity, z.B. ein Bild, hat eine URI, die den Pfad zur Entity angibt. Der Input-String der Funktion muss dem Namen der

 © 2019-2025 Altova GmbH

in der DTD deklarierten ungeparsten Entity entsprechen, dann gibt die Funktion die URI der ungeparsten Entity zurück. Diese URI kann dann an einen Node im Ausgabedokument, z.B. an einen **href** Node gerichtet werden.



Sprachen

XSLT 1.0.

Parameter

Name	Typ	Beschreibung
string	`xs:string`	Der Name der ungeparsten Entity, deren URI abgerufen werden soll.

6 Komplexe Mappings

Altova Website: Datenintegrationstool

In diesem Abschnitt sind komplexe Mapping-Szenarien beschrieben. Er enthält die folgenden Kapitel:

- Mappen von Node-Namen 405
- Stapel-Verarbeitung von Dateien 422

© 2019-2025 Altova GmbH

6.1 Mappen von Node-Namen

Wenn Sie ein Mapping mit MapForce erstellen, ist das Ziel in den meisten Fällen, *Werte* aus einer Quelldatei zu lesen und *Werte* in eine Zieldatei zu schreiben. In manchen Fällen benötigen Sie jedoch nicht nur Zugriff auf die Node-*Werte* aus der Quelldatei, sondern auch auf die Node-Namen. Sie könnten z.B. ein Mapping benötigen, in dem die Element- oder Attributnamen (nicht die Werte) aus einer XML-Quelldatei gelesen und in Element- oder Attributwerte (nicht Namen) in einer XML-Zieldatei konvertiert werden.

Betrachten Sie das folgende Beispiel: Sie haben eine XML-Datei, die eine Liste von Produkten enthält. Jedes Produkt hat das folgende Format:

```
<product>
   <id>1</id>
   <color>red</color>
   <size>10</size>
</product>
```

Ihr Ziel ist es, Informationen über jedes Produkt in Namen-Wert-Paare zu konvertieren, z.B.:

```
<product>
   <attribute name="id" value="1" />
   <attribute name="color" value="red" />
   <attribute name="size" value="10" />
</product>
```

In diesem Szenario benötigen Sie im Mapping Zugriff auf den Node-Namen. Beim *dynamischen* Zugriff auf Node-Namen können Sie Datenkonvertierungen wie die oben beschriebene durchführen.

Anmerkung: Sie können die Transformation auch mit Hilfe der Funktionen `node-name` 292 und `static-node-name` 294 aus der core-Bibliothek durchführen. In diesem Fall müssen Sie jedoch genau wissen, welche Element-Namen in der Quelldatei vorkommen und Sie müssen jedes einzelne Element manuell mit der Zielkomponente verbinden. Diese Funktionen genügen oft außerdem nicht, wenn Sie z.B. Nodes nach Namen filtern oder gruppieren müssen oder den Datentyp des Node im Mapping bearbeiten müssen.

Ein dynamischer Zugriff auf Node-Namen ist nicht nur beim Lesen von Node-Namen, sondern auch beim Schreiben von Node-Namen möglich. In einem Standard-Mapping sind die Namen von Attributen oder Elementen in einer Zieldatei immer noch vor Ausführung des Mappings bekannt; sie stammen aus dem Schema, das der Komponente zugrunde liegt. Bei dynamischen Node-Namen können Sie hingegen neue Attribute oder Elemente erstellen, deren Namen vor Ausführung des Mappings nicht bekannt sind. Der Name des Attributs oder Elements kommt in diesem Fall aus dem Mapping selbst, nämlich aus jeder beliebigen von MapForce unterstützten Quelldatei.

> Damit ein dynamischer Zugriff auf die Child-Elemente oder -Attribute eines Node möglich ist, muss der Node tatsächlich Child-Elemente oder -Attribute haben und darf nicht der XML-Node sein.

Dynamische Node-Namen werden unterstützt, wenn Sie von oder auf die folgenden Komponententypen mappen:

- XML
- CSV/FLF*

* MapForce Professional oder Enterprise Edition erforderlich.

Dynamische Node-Namen werden unterstützt in jeder der folgenden Mapping-Sprachen unterstützt: Built-In*, XSLT2, XSLT 3.0, XQuery*, C#*, C++*, Java*.

* MapForce Professional oder Enterprise Edition erforderlich.

Informationen zur Funktionsweise von dynamischen Node-Namen finden Sie unter Zugriff auf Node-Namen 406. Ein Schritt-für-Schritt-Beispiel für ein solches Mapping finden Sie unter Beispiel: Mappen von Elementnamen auf Attributwerte 418.

6.1.1 Zugriff auf Node-Namen

Wenn ein Node in einer XML-Komponente Child-Nodes hat, können sowohl der Name als auch der Wert jedes einzelnen Child-Node direkt im Mapping abgerufen werden. Diese Methode wird als "dynamische Node-Namen" bezeichnet. "Dynamisch" bezieht sich darauf, dass die Verarbeitung "on-the-fly" zur Mapping-Laufzeit und nicht auf Basis statischer noch vor Ausführung des Mappings bekannter Schemainformationen erfolgt. In diesem Kapitel wird näher erläutert, wie Sie dynamischen Zugriff auf Node-Namen erhalten und was Sie damit erreichen können.

Wenn Daten aus einer Quelldatei ausgelesen werden, bedeutet "dynamische Node-Namen", dass Folgendes möglich ist:

- Abrufen einer Liste aller Child-Nodes (oder Attribute) eines Node als Sequenz. Eine "Sequenz" in MapForce ist eine Liste von null oder mehr Datenelementen, die mit einer Zielkomponente verbunden werden können, sodass in der Zielkomponente dieselbe Anzahl an Datenelementen wie in der Quelldatei erstellt werden kann. Wenn z.B. ein Node fünf Attribute in der Quelldatei hat, so könnten in der Zieldatei fünf neue Elemente, eines für jedes Attribut, erstellt werden.
- Lesen nicht nur der Child-Node-Werte (wie bei einem Standard-Mapping), sondern auch der Namen dieser Nodes.

Wenn Daten in eine Zieldatei geschrieben werden sollen, bedeutet "dynamische Node-Namen", dass Folgendes möglich ist:

- Erstellung neuer Nodes anhand von Namen aus dem Mapping (so genannter "dynamischer" Namen) anstelle von Namen, die von den Komponenteneinstellungen bereitgestellt werden (so genannte "statische" Namen).

Zur Veranschaulichung dynamischer Node-Namen wird in diesem Kapitel das folgende XML-Schema verwendet: **<Dokumente>\Altova\MapForce2025\MapForceExamples\Products.xsd**. Das Beispielinstanzdokument zu diesem Schema ist **Products.xml**. Um Schema und Instanzdatei zum Mapping-Bereich hinzuzufügen, wählen Sie den Befehl **Einfügen | XML-Schema/Datei** und navigieren Sie zum Ordner **<Dokumente>\Altova\MapForce2025\MapForceExamples\Products.xml**. Wenn Sie aufgefordert werden, ein Root-Element auszuwählen, klicken Sie auf `products`.

Um dynamische Node-Namen für den Node `product` zu aktivieren, klicken Sie mit der rechten Maustaste darauf und wählen Sie einen der folgenden Kontextmenübefehle:

 © 2019-2025 Altova GmbH

- **Attribute mit dynamischem Namen anzeigen,** wenn Sie Zugriff auf die Attribute des Node benötigen
- **Child-Elemente mit dynamischem Namen anzeigen**, wenn Sie Zugriff auf die Child-Elemente des Node benötigen

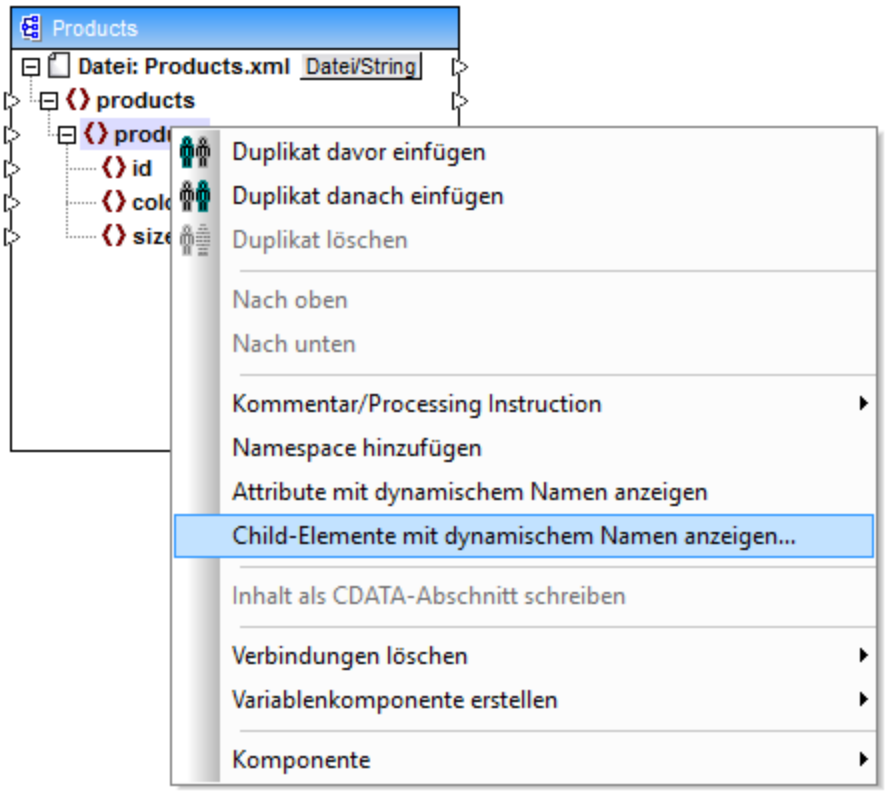


Abb. 1 Aktivieren dynamischer Node-Namen (für Child-Elemente)

Anmerkung: Die oben beschriebenen Befehle stehen nur für Nodes zur Verfügung, die Child-Nodes haben. Außerdem stehen die Befehle auch nicht für Root-Nodes zur Verfügung.

Wenn Sie bei einem Node in den dynamischen Modus wechseln, wird ein Dialogfeld wie das unten gezeigte, angezeigt. Wählen Sie für das Beispiel in diesem Kapitel die unten gezeigten Optionen aus; eine nähere Beschreibung zu diesen Optionen finden Sie unter Zugriff auf Nodes eines bestimmten Typs 414.

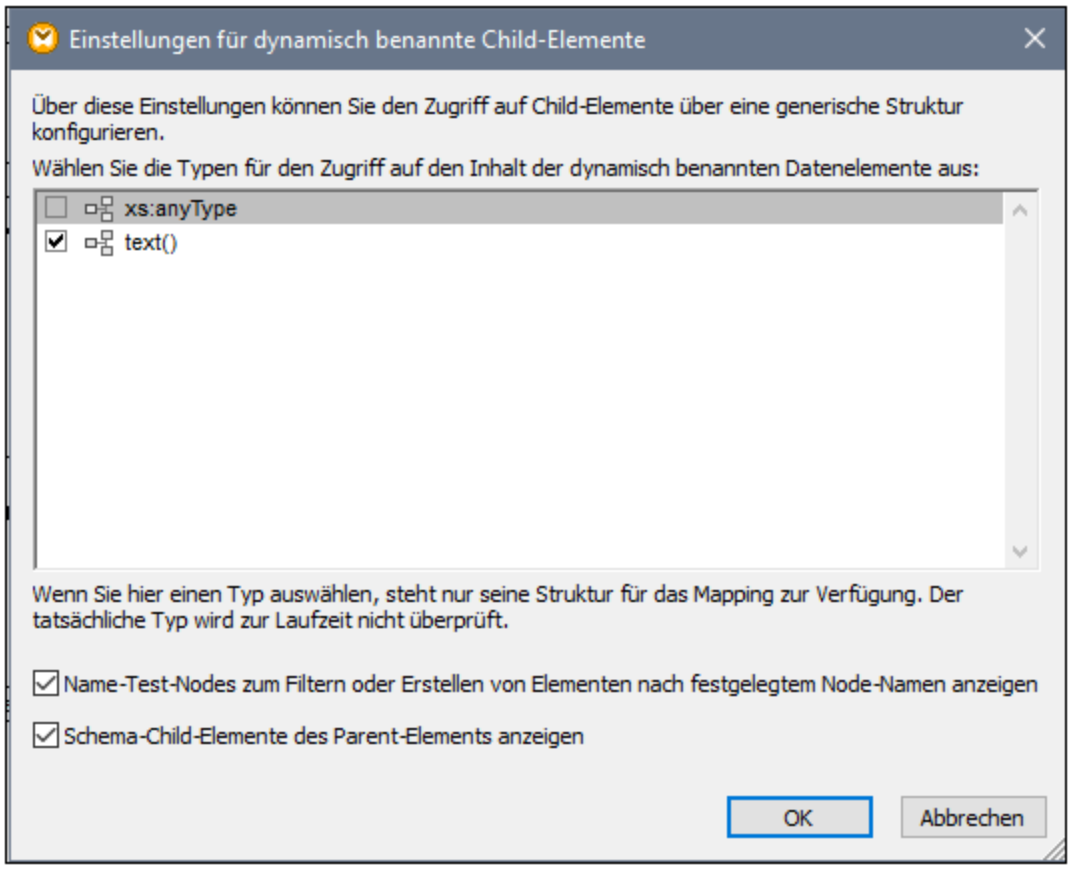


Abb. 2 Dialogfeld "Einstellungen für dynamisch benannte Child-Elemente"

In Abb. 3 sehen Sie, wie die Komponente aussieht, wenn dynamische Node-Namen für den `product` Node aktiviert sind. Beachten Sie, wie sehr sich das Aussehen der Komponente jetzt geändert hat.

 © 2019-2025 Altova GmbH

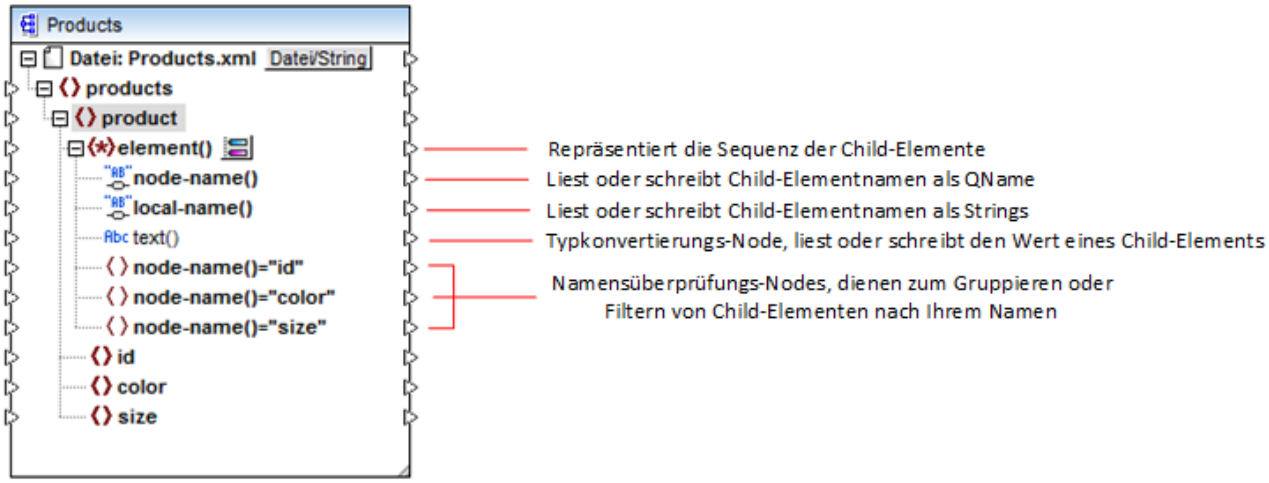


Abb.3 Aktivieren dynamischer Node-Namen (für Elemente)

Um die Komponente wieder zurück in den Standardmodus zu schalten, klicken Sie mit der rechten Maustaste auf den Node `product` und deaktivieren Sie die Option **Child-Elemente mit dynamischem Namen anzeigen** im Kontextmenü.

In der Abbildung unten sehen Sie, wie dieselbe Komponente aussieht, wenn der dynamische Zugriff auf Attribute eines Node aktiviert ist. Die Komponente wurde durch Rechtsklick auf das Element `product` und Auswahl des Kontextmenübefehls **Attribute mit dynamischem Namen anzeigen** definiert.

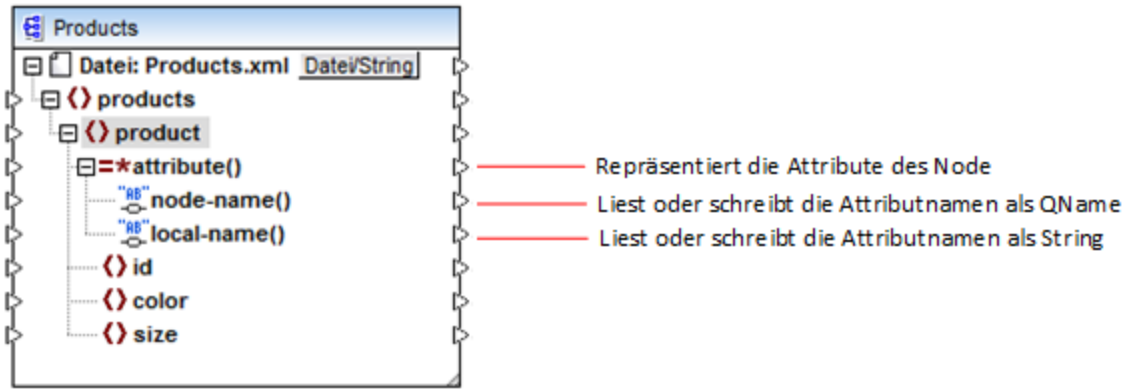


Abb. 4 Dynamische Node-Namen (für Attribute) sind aktiviert

Um die Komponente wieder zurück in den Standardmodus zu schalten, klicken Sie mit der rechten Maustaste auf den Node `product` und deaktivieren Sie die Option **Attribute mit dynamischem Namen anzeigen** im Kontextmenü.

Wie Sie in Abbildung 3 und 4 sehen, ändert sich das Aussehen der Komponente, wenn ein Node (in diesem Fall `product`) in den Modus "dynamischer Node-Name" wechselt. In diesem Modus ist nun Folgendes möglich:

- Lesen oder Schreiben einer Liste alle Child-Elemente oder -Attribute eines Node. Diese werden vom Datenelement `element()` bzw. `attribute()` bereitgestellt.

- Lesen oder Schreiben der Namen der einzelnen Child-Elemente oder -Attribute. Der Name wird von den Datenelementen `node-name()` und `local-name()` bereitgestellt.
- Lesen oder Schreiben des Werts von einzelnen Child-Elementen (bei Elementen) als spezifischer Datentyp. Dieser Wert wird vom Typkonvertierungs-Node (in diesem Fall dem Datenelement `text()`) bereitgestellt. Beachten Sie, dass nur Elemente Typkonvertierungs-Nodes haben. Attribute werden immer als "String"-Typ behandelt.
- Gruppieren oder Filtern generischer Child-Elemente nach Namen.

Im Folgenden finden Sie eine Beschreibung der Node-Typen, mit denen Sie im Modus "dynamischer Node-Name" arbeiten können.

element()

Dieser Node weist in einer Quellkomponente ein anderes Verhalten als in einer Zielkomponente auf. Er stellt in der Quellkomponente die Child-Elemente des Node als Sequenz bereit. In Abb.3 stellt `element()` eine Liste (Sequenz) aller Child-Elemente von `product` bereit. Die anhand des folgenden XML-Fragments erstellte Sequenz würde z.B. drei Datenelemente enthalten (da `product` drei Child-Elemente hat):

```
<product>
   <id>1</id>
   <color>red</color>
   <size>10</size>
</product>
```

Beachten Sie, dass der tatsächliche Name und Typ der einzelnen Datenelemente in der Sequenz vom Node `node-name()` bzw. dem Typkonvertierungs-Node bereitgestellt wird (Beschreibung siehe unten). Um dies zu veranschaulichen, stellen Sie sich vor, Sie müssen Daten folgendermaßen aus einer XML-Quelldatei in eine XML-Zieldatei transformieren:

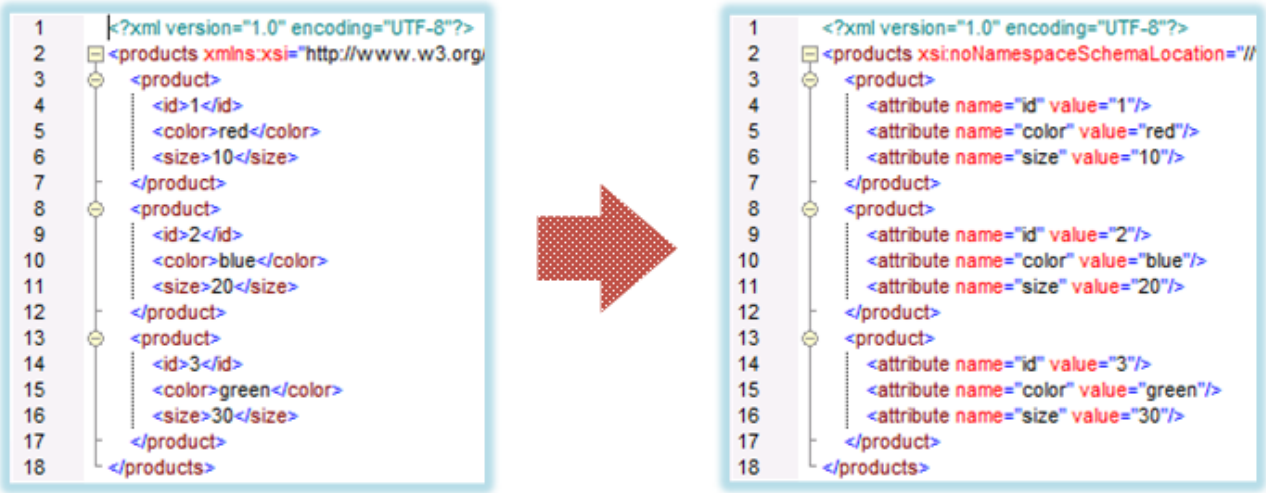


Abb. 6 Mappen von XML-Elementnamen auf Attributwerte (Aufgabe)

Das Mapping, mit dem Sie dies erreichen, sieht folgendermaßen aus:

 © 2019-2025 Altova GmbH

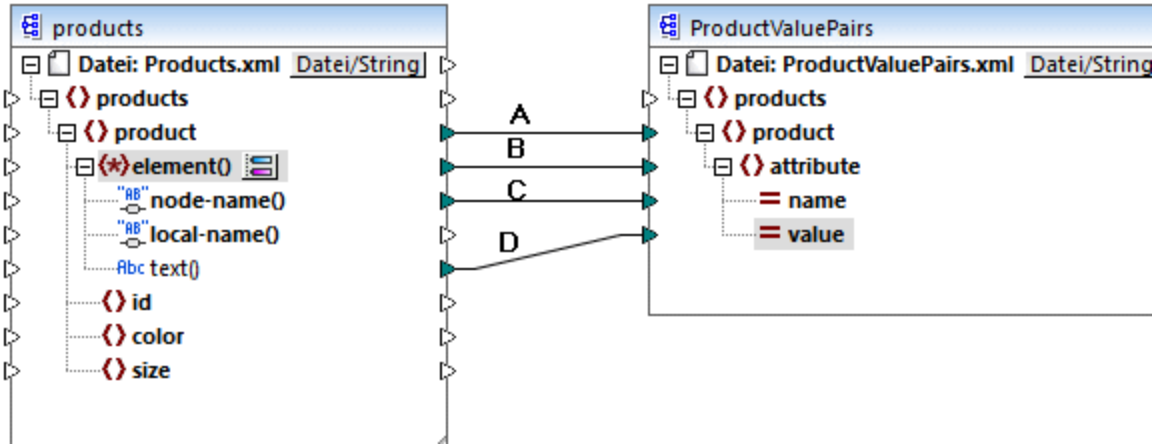


Abb. 7 Mappen von XML-Elementnamen auf Attributwerte (in MapForce)

Die Rolle von `element()` ist es hier, die Sequenz von Child-Elementen von `product`, bereitzustellen, während `node-name()` und `text()` den tatsächlichen Namen und Wert der einzelnen Datenelemente in der Sequenz bereitstellen. Zu diesem Mapping gibt es ein Tutorial-Beispiel, das unter Beispiel: Mappen von Elementnamen auf Attributwerte 418 näher beschrieben ist.

`element()` selbst erstellt nichts in der Zielkomponente. Dies stellt eine Ausnahme der Grundregel "Erstelle für jedes Datenelement in der Quellkomponente ein Datenelement in der Zielkomponente" dar. Die eigentlichen Elemente werden (unter Verwendung des Werts von `node-name()`) durch die Typkonvertierungs- und (unter Verwendung der eigenen Namen) durch die Namensüberprüfungs-Nodes erstellt.

attribute()

Wie Sie in Abb. 4 sehen, ermöglicht dieses Datenelement zur Mapping-Laufzeit den Zugriff auf alle Attribute des Node. Es stellt in einer Quellkomponente die Attribute des damit verbundenen Quell-Node als Sequenz bereit. So würde die Sequenz im folgenden XML-Fragment etwas zwei Datenelemente enthalten (da `product` zwei Attribute hat):

```
<product id="1" color="red" />
```

Beachten Sie, dass der `attribute()`-Node nur den Wert der einzelnen Attribute in der Sequenz, und zwar immer als String-Typ, bereitstellt. Der Name der einzelnen Attribute wird vom Node `node-name()` bereitgestellt.

In einer Zielkomponente wird über diesen Node eine verbundene Sequenz verarbeitet und für jedes Datenelement in der Sequenz wird ein Attributwert erstellt. Der Attributname wird vom Node `node-name()` bereitgestellt. Angenommen, Sie möchten Daten aus einer XML-Quelldatei folgendermaßen in eine XML-Zieldatei transformieren:

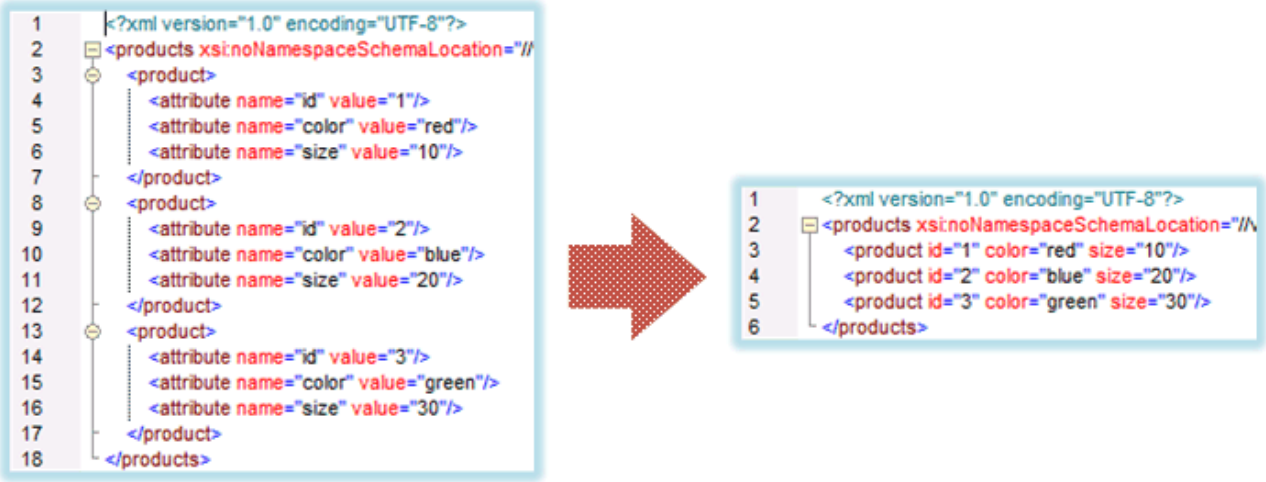


Abb. 8 Mappen von Attributwerten auf Attributnamen (Aufgabe)

Das Mapping, mit dem Sie dies erreichen, sieht folgendermaßen aus:

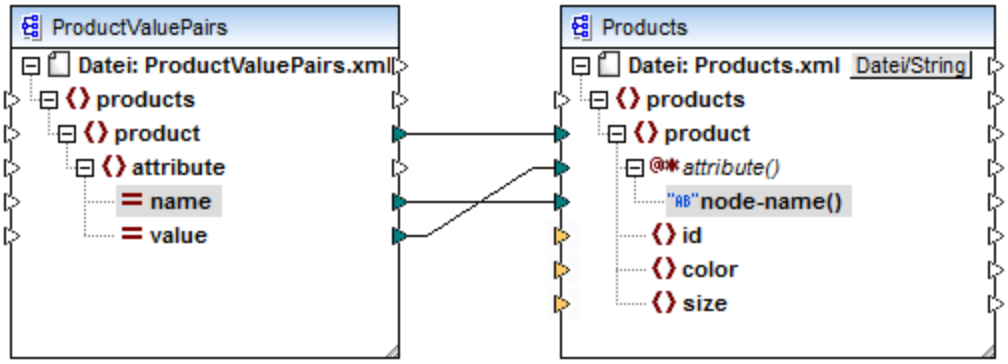


Abb. 9 Mappen von Attributwerten auf Attributnamen (in MapForce)

Anmerkung: Diese Transformation lässt sich auch ohne Aktivierung des dynamischen Zugriffs auf die Attribute eines Node bewerkstelligen. Hier wird nur gezeigt, wie `attribute()` in einer Zielkomponente funktioniert.

Wenn Sie dieses Mapping nachstellen möchten, verwenden Sie dazu dieselben XML-Komponenten wie im Mapping **ConvertProducts.mfd** aus dem Ordner **<Dokumente>\Altova\MapForce2025\MapForceExamples**. Der einzige Unterschied besteht darin, dass die Zieldatei nun als Quelldatei und die Quelldatei als Zieldatei verwendet wird. Sie benötigen als Input-Daten für die Quellkomponente eine XML-Instanz, die tatsächlich Attribute-Werte enthält, wie z.B.:

```
<?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>
```

Beachten Sie, dass die Namespace- und die Schema-Deklaration im obigen Codefragment aus Gründen der Einfachheit weggelassen wurden.

node-name()

In einer Quellkomponente stellt `node-name()` die Namen der einzelnen Child-Elemente von `element()` bzw. die Namen der einzelnen Child-Attribute von `attribute()` bereit. Standardmäßig hat der bereitgestellte Name den Typ `xs:QName`. Um den Namen als String zu erhalten, verwenden Sie den Node `local-name()` (siehe Abb. 3).

In einer Zielkomponente schreibt `node-name()` die Namen der einzelnen in `element()` oder `attribute()` enthaltenen Elemente bzw. Attribute.

local-name()

Dieser Node funktioniert auf dieselbe Art wie `node-name()` mit dem Unterschied, dass der Typ ist `xs:string` anstelle von `xs:QName`.

Typkonvertierungs-Node

Der Typkonvertierungs-Node in einer Quellkomponente stellt den Wert der einzelnen in `element()` enthaltenen Child-Elemente bereit. Der Name und die Struktur dieses Node hängen von dem im Dialogfeld "Einstellungen für dynamisch benannte Child-Elemente" ausgewählten Typ ab (Abb. 2).

Um den Typ des Node zu ändern, klicken Sie auf die Schaltfläche **Auswahl ändern** ( ) und wählen Sie einen Typ aus der Liste der verfügbaren Typen (darunter auch eine Schema Wildcard (`xs:any`)) aus. Nähere Informationen dazu finden Sie unter Zugriff auf Nodes eines bestimmten Typs 414.

In einer Zielkomponente wird über den Typkonvertierungs-Node der Wert der einzelnen in `element()` enthaltenen Child-Elemente als spezifischer Datentyp geschrieben. Der gewünschte Datentyp kann auch hier über die Schaltfläche **Auswahl ändern** ( ) ausgewählt werden.

Namensüberprüfungs-Nodes

Namensüberprüfiungs-Nodes bieten in einer Quellkomponente eine Möglichkeit, Child-Elemente aus einer Quellinstanz nach Namen zu filtern. Child-Elemente müssen eventuell nach Namen gefiltert werden, um sicherzustellen, dass das Mapping den korrekten Typ verwendet, um auf die Instanzdaten zuzugreifen (siehe Zugriff auf Nodes eines bestimmten Typs 414).

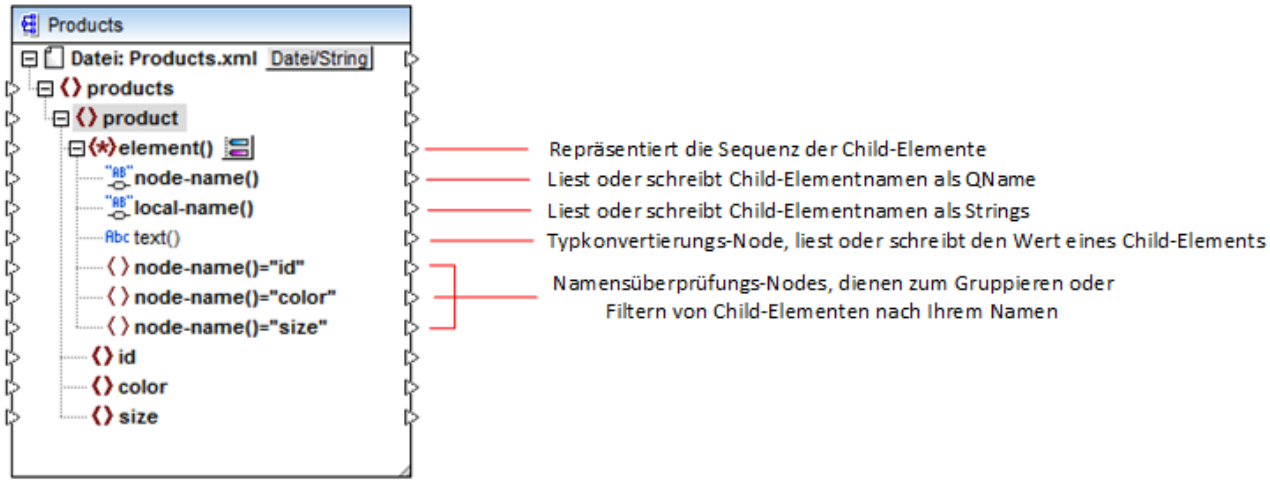
Im Allgemeinen funktionieren Namensüberprüfungs-Nodes beim Lesen und Schreiben von Werten und hierarchisch untergeordneten Strukturen fast wie normale Element-Nodes. Da sich die Mapping-Semantik aber unterscheidet, wenn der dynamische Zugriff aktiviert ist, gibt es einige Einschränkungen. So können Sie etwa die Werte von zwei Namensüberprüfungs-Nodes nicht miteinander verknüpfen.

In der Zielkomponente werden für Namensüberprüfungs-Nodes in der Ausgabe so viele Elemente erstellt, wie Datenelemente in der verbundenen Quellsequenz vorhanden sind. Ihr Name setzt den auf `node-name()` gemappten Wert außer Kraft.

Falls erforderlich, können Sie die Namensüberprüfungs-Nodes aus der Komponente ausblenden. Klicken Sie dazu auf die Schaltfläche **Auswahl ändern** ( ) neben dem `element()`-Node. Deaktivieren Sie anschließend im Dialogfeld "*Einstellungen für dynamisch benannte Child-Elemente*" (Abb. 2) das Kontrollkästchen **Name-Test-Nodes ...anzeigen"**.

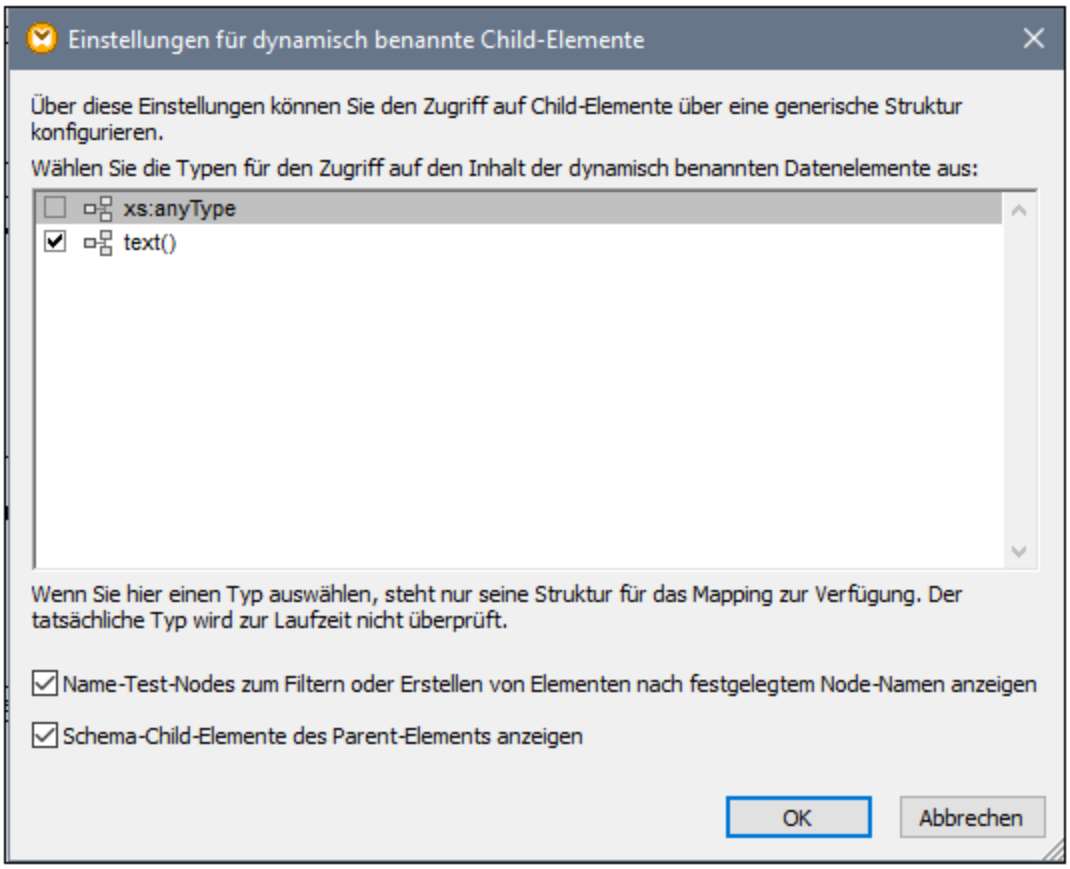
6.1.2 Zugriff auf Nodes eines bestimmten Typs

Wie in vorherigen Abschnitt, Zugriff auf Node-Namen 406, erwähnt, erhalten Sie durch Rechtsklick auf den Node und Auswahl des Kontextmenübefehls **Child-Elemente mit generischem Namen anzeigen** Zugriff auf alle Chld-Elemente eines Node. Auf diese Art stehen die Namen der einzelnen Child-Elemente über den Node `node-name()` zur Mapping-Laufzeit Verfügung, während der Wert des Node über einen speziellen Typkonvertierungsnode (im Bild unten der Node `text()`) zur Verfügung steht.



Beachten Sie, dass der Datentyp der einzelnen Child-Elemente vor der Mapping-Laufzeit nicht bekannt ist. Jedes Child-Element kann außerdem einen anderen Datentyp haben. So kann z.B. ein `product`-Node in der XML-Instanz ein Child-Element `id` vom Typ `xs:integer` und ein Child-Element `size` vom Typ `xs:string` haben. Damit Sie Zugriff auf den Node-Inhalt eines bestimmten Typs haben, wird jedes Mal, wenn Sie den dynamischen Zugriff auf die Child-Elemente eines Node aktivieren, das unten gezeigte Dialogfeld geöffnet. Sie können dieses Dialogfeld auch später jederzeit durch Klick auf die Schaltfläche **Auswahl ändern** ( ) neben einem `element(-Node` aufrufen.

 © 2019-2025 Altova GmbH



Dialogfeld "Einstellungen für dynamisch benannte Child-Elemente"

Es gibt verschiedene Möglichkeiten, um zur Laufzeit Zugriff auf den Inhalt der einzelnen Child-Elemente zu erhalten:

1. Aufruf des Inhalts als String. Aktivieren Sie dazu das Kontrollkästchen **text()** im oben gezeigten Dialogfeld. In diesem Fall wird beim Schließend des Dialogfelds in der Komponente ein `text()`-Node erstellt. Diese Option eignet sich, wenn der Inhalt einen simpleType hat (`xs:int`, `xs:string`, usw.). Eine Beschreibung dazu finden Sie unter Beispiel: Mappen von Elementnamen auf Attributwerte 418. Beachten Sie, dass ein **text()**-Node nur angezeigt wird, wenn ein Child Node des aktuellen Node Text enthalten kann.
2. Aufruf des Inhalts als bestimmter complexType, der laut Schema zulässig ist. Wenn laut Schemadefinition global definierte benutzerdefinierte complexTypes für den ausgewählten Node zulässig sind, stehen diese ebenfalls im obigen Dialogfeld zur Verfügung und Sie können das Kontrollkästchen daneben aktivieren. In der Abbildung oben sind im Schema keine complexTypes definiert, daher stehen keine zur Auswahl zur Verfügung.
3. Aufruf des Inhalts als any-Typ. Dies kann in komplexen Mapping-Szenarien hilfreich sein (siehe Zugriff auf tiefer verschachtelte Strukturen") weiter unten. Aktivieren Sie dazu das Kontrollkästchen neben **xs:anyType**.

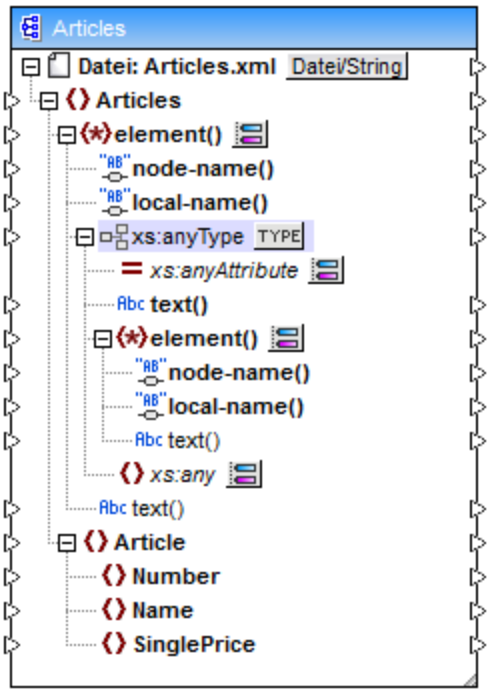
Beachten Sie, dass MapForce (über den Typkonvertierungs-Node) keinerlei Informationen darüber hat, was für einen Datentyp der Instanz-Node zur Mapping-Laufzeit tatsächlich haben wird. Ihr Mapping muss daher

den Node-Inhalt mit dem richtigen Typ aufrufen. Wenn Sie z.B. erwarten, dass der Node einer XML-Quellinstanz Child-Nodes verschiedener complexType-Arten haben kann, gehen Sie folgendermaßen vor:
a) Setzen Sie den Typkonvertierungs-Node auf den complexTyp, mit dem er übereinstimmen muss (siehe Punkt 2 in der Liste oben).
b) Fügen Sie einen Filter hinzu, damit nur der gewünschte complexType aus der Instanz ausgelesen wird. Nähere Informationen zu Filtern finden Sie unter Filter und Bedingungen 194.

Zugriff auf tiefer verschachtelte Strukturen

Es ist möglich, Zugriff auf Nodes, die sich auf einer Ebene noch unterhalb der unmittelbaren Child-Nodes eines Node befinden, zu erhalten. Dies eignet sich für komplexe Mapping-Szenarien. Bei einfachen Mappings wie z.B. im Beispiel: Mappen von Elementnamen auf Attributwerte 418 ist diese Methode nicht notwendig, da im Mapping nur die unmittelbaren Child-Nodes eines XML-Node aufgerufen werden. Wenn Sie jedoch dynamischen Zugriff auf tiefer gelegene Strukturen wie z.B. "Enkel" und "Urenkel" usw. benötigen, so gehen Sie vor, wie unten gezeigt.

1. Erstellen Sie ein neues Mapping.
2. Klicken Sie im Menü "Einfügen" auf **XML-Schema/Datei einfügen** und navigieren Sie zur XML-Instanzdatei (in diesem Beispiel die Datei **Articles.xml** aus dem Ordner **<Dokumente>\Altova\MapForce2025\MapForceExamples**).
3. Klicken Sie mit der rechten Maustaste auf den Node `Articles` und wählen Sie den Kontextmenübefehl **Child-Elemente mit dynamischem Namen anzeigen**.
4. Wählen Sie im Dialogfeld "Einstellungen für dynamisch benannte Child-Elemente" **xs:anyType** aus.
5. Klicken Sie mit der rechten Maustaste auf den Node **xs:anyType** und wählen Sie erneut den Kontextmenübefehl **Child-Elemente mit dynamischem Namen anzeigen**.
6. Wählen Sie im Dialogfeld "Einstellungen für dynamisch benannte Child-Elemente" **text()** aus.



Beachten Sie, dass die Komponente oben zwei `element()` Nodes enthält. Der zweite `element()`-Node ermöglicht dynamischen Zugriff auf die Enkel des Node `<Articles>` in der Instanz **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

Um z.B. die Enkelelementnamen (`Number`, `Name`, `SinglePrice`) abzurufen, würden Sie eine Verbindung vom Node `local-name()` unterhalb des zweiten `element()` Node zu einem Zieldatenelement ziehen. Um die Enkelelementwerte (`1`, `T-Shirt`, `25`) abzurufen, würden Sie eine Verbindung vom Node `text()` ziehen.

Zwar gilt dies nicht für dieses Beispiel, doch können Sie in realen Einsatzszenarien auch für jeden nachfolgenden `xs:anyType` Node den dynamischen Zugriff auf Node-Namen aktivieren, um noch tiefer gelegene Ebenen zu erreichen.

Beachten Sie bitte Folgendes:

- Über die Schaltfläche TYPE können Sie jeden abgeleiteten Typ aus dem aktuellen Schema auswählen und in einem separaten Node anzeigen. Dies ist nur dann nützlich, wenn Sie von oder auf abgeleitete Schematypen mappen müssen (siehe Abgeleitete XML-Schema-Typen 136).
- Über die neben einem `element ()` Node gelegene Schaltfläche **Auswahl ändern** ( ) rufen Sie das in diesem Kapitel beschriebene Dialogfeld "Einstellungen für dynamisch benannte Child-Elemente" auf.
- Über die Schaltfläche **Auswahl ändern** ( ) neben dem `xs:anyAttribute`-Attribut können Sie jedes beliebige global im Schema definierte Attribut auswählen. Auf die gleiche Art können Sie über die Schaltfläche **Auswahl ändern** ( ) neben dem `xs:any`-Element jedes beliebige global im Schema definierte Element auswählen. Dies funktioniert auf dieselbe Art und Weise wie das Mappen von oder auf Schema-Wildcards (siehe auch Wildcards - xs:any / xs:anyAttribute 142). Vergewissern Sie sich

bei Verwendung dieser Option, dass das ausgewählte Attribut oder Element gemäß dem Schema auf dieser Ebene auch wirklich zulässig ist.

6.1.3 Beispiel: Mappen von Elementnamen auf Attributwerte

In diesem Beispiel wird gezeigt, wie Sie Elementnamen aus einem XML-Dokument auf Attributwerte in einem XML-Zieldokument mappen. Das Mapping zu diesem Beispiel finden Sie im folgenden Ordner: **<Dokumente>\Altova\MapForce2025\MapForceExamples\ConvertProducts.mfd**.

Betrachten wir zur Erläuterung des Beispiels folgendes Szenario: Angenommen Sie haben eine XML-Datei, die eine Liste von Produkten enthält. Jedes Produkt hat das folgende Format:

```
<product>
   <id>1</id>
   <color>red</color>
   <size>10</size>
</product>
```

Unser Ziel ist es, Informationen zu jedem Produkt in ein Namen-Wert-Paar zu konvertieren, z.B.:

```
<product>
   <attribute name="id" value="1" />
   <attribute name="color" value="red" />
   <attribute name="size" value="10" />
</product>
```

Um ein Mapping wie das oben beschriebene mit möglichst wenig Aufwand durchzuführen, wird in diesem Beispiel eine MapForce-Funktion namens "dynamischer Zugriff auf Node-Namen" verwendet. "Dynamisch" bedeutet, dass die Node-Namen (und nicht nur die Werte) während der Ausführung des Mappings gelesen und als Werte geschrieben werden. Das Mapping dazu wird in wenigen Schritten folgendermaßen erstellt:

Schritt 1: Hinzufügen der XML-Quellkomponente zum Mapping

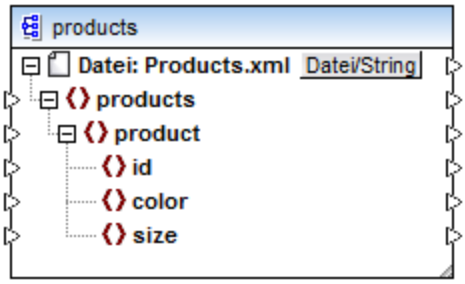
- Klicken Sie im Menü **Einfügen** auf **XML-Schema/Datei** und navigieren Sie zur folgenden Datei: **<Dokumente>\Altova\MapForce2025\MapForceExamples\Products.xml**. Diese XML-Datei verweist auf das Schema **Products.xsd** im selben Ordner.

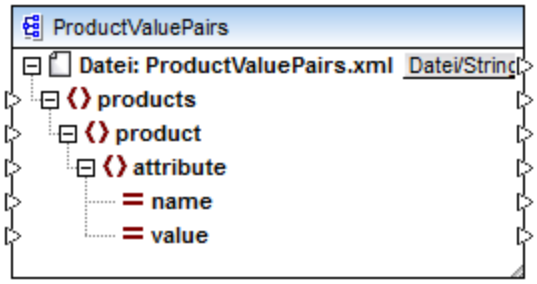
Schritt 2: Fügen Sie die XML-Zielkomponente zum Mapping hinzu

- Klicken Sie im Menü **Einfügen** auf **XML-Schema/Datei** und navigieren Sie zur folgenden Datei: **<Dokumente>\Altova\MapForce2025\MapForceExamples\ProductValuePairs.xsd**. Wenn Sie aufgefordert werden, eine Instanzdatei anzugeben, klicken Sie auf **Überspringen**. Wenn Sie aufgefordert werden, ein Root-Element auszuwählen, wählen Sie `products` als Root-Element.

Das Mapping sollte zu diesem Zeitpunkt folgendermaßen aussehen:

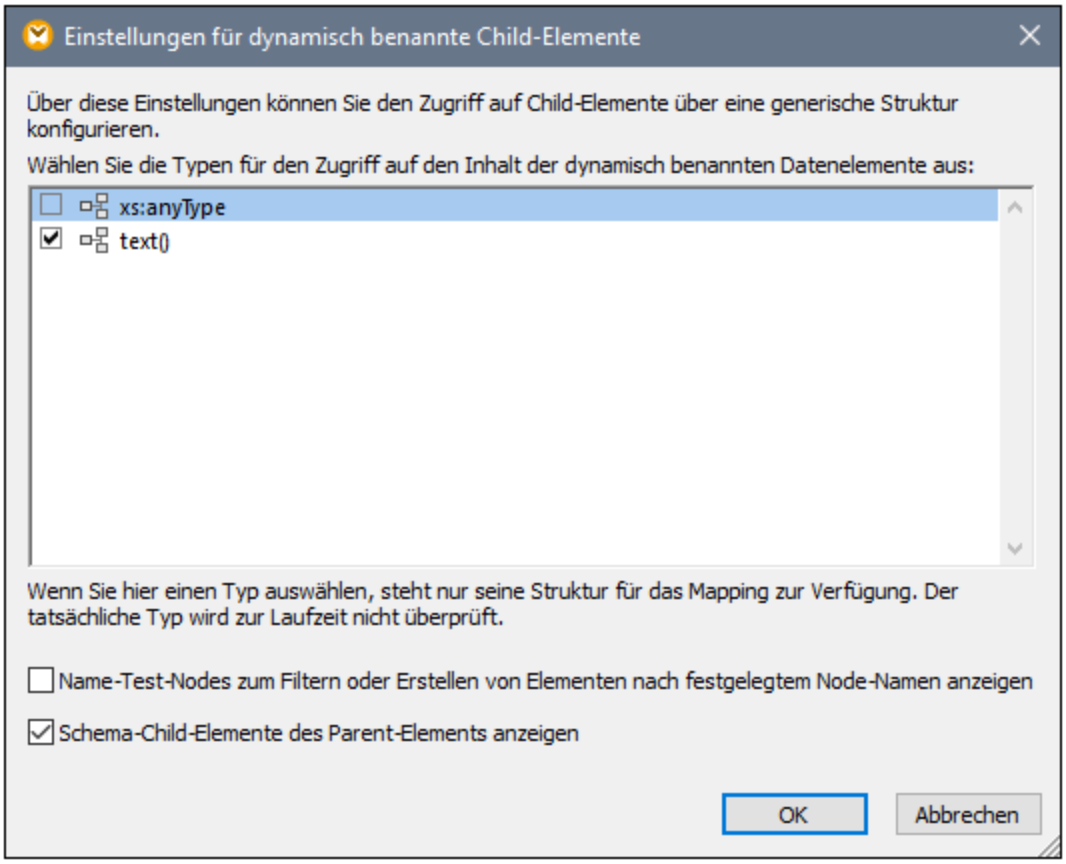
 © 2019-2025 Altova GmbH



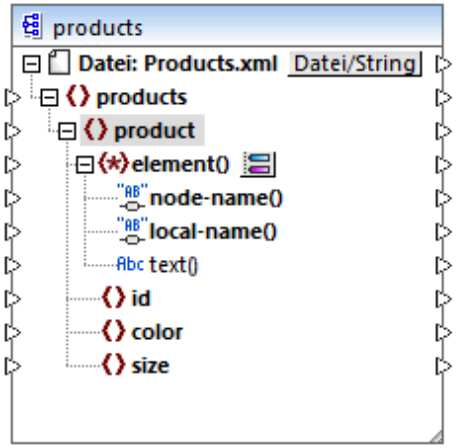


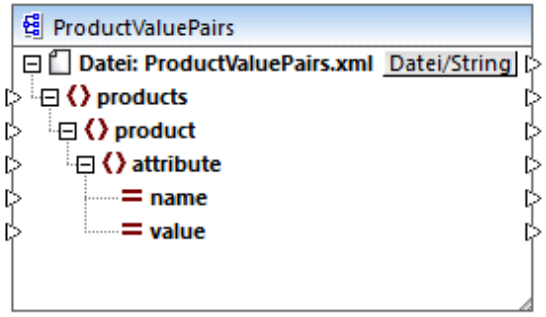
Schritt 3: Aktivieren Sie den dynamischen Zugriff auf Child-Nodes

1. Klicken Sie mit der rechten Maustaste in der Quellkomponente auf den Node `product` und wählen Sie im Kontextmenü den Befehl **Child-Elemente mit dynamischem Namen anzeigen**.
2. Wählen Sie im Dialogfeld, das daraufhin geöffnet wird, **text()** als Typ aus. Belassen Sie andere Optionen unverändert.



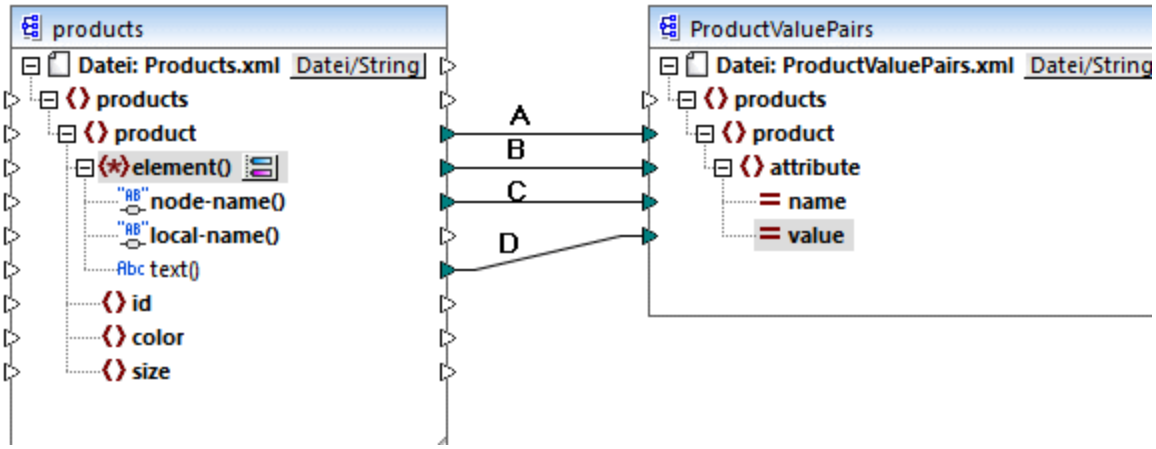
Beachten Sie, dass nun ein `text()`-Node zur Quellkomponente hinzugefügt wurde. Dieser Node stellt den Inhalt der einzelnen Child-Elemente für das Mapping bereit (in diesem Fall den Wert von "id", "color" und "size").





Schritt 4: Ziehen Sie die Mapping-Verbindungen

Ziehen Sie schließlich die Mapping-Verbindungen A, B, C, D, wie unten gezeigt. Doppelklicken Sie optional auf die einzelnen Verbindungen und geben Sie von oben nach unten die Buchstaben "A", "B", "C" und "D" in die einzelnen Beschreibungsfelder ein.



ConvertProducts.mfd

Im oben gezeigten Mapping wird mit Verbindung A für jedes Produkt in der Quellkomponente ein Produkt in der Zielkomponente erstellt. Bis jetzt handelt es sich um eine Standard-Mapping-Verbindung, die die Node-Namen nicht berücksichtigt. In Verbindung B wird jedoch für jedes Child-Element von `product` in der Zielkomponente ein neues Element namens `attribute` erstellt.

Die Verbindung B ist eine entscheidende Verbindung im Mapping, da damit eine Sequenz von Child-Elementen von `product` von der Quellkomponente in die Zielkomponente übertragen wird. Dabei werden nicht die tatsächlichen *Namen* oder *Werte* übertragen. Diese Verbindung ist folgendermaßen zu verstehen: Wenn **element()** in der Quellkomponente N Child-Elemente hat, so werden in der Zielkomponente N Instanzen dieses Datenelements erstellt. In diesem konkreten Beispiel hat `product` in der

 © 2019-2025 Altova GmbH

Quellkomponente drei Child-Elemente (`id`, `color` und `size`). Das bedeutet, dass jedes `product`-Element in der Zieldatei drei Child-Elemente mit dem Namen `attribute` hat.

In diesem Beispiel wird dies zwar nicht gezeigt, doch wird dieselbe Regel auch auf das Mappen von Child-Elemente von **attribute()** angewendet. Wenn das Datenelement **attribute()** in der Quellkomponente N Child-Attribute hat, so werden in der Zielkomponente N Instanzen dieses Datenelements erstellt.

Als Nächstes werden in Verbindung C die tatsächlichen Namen der einzelnen Child-Elemente von `product` in die Zielkomponente kopiert (nämlich "id", "color" und "size").

Schlussendlich werden in Verbindung D die Werte der einzelnen Child-Elemente von product als String-Typ in die Zielkomponente kopiert.

Um eine Vorschau auf die Mapping-Ausgabe zu sehen, klicken Sie auf das Fenster **Ausgabe** und werfen Sie einen Blick auf die generierte XML-Datei. Wie erwartet und beabsichtigt, enthält die Ausgabe mehrere Produkte, deren Daten als Namen-Wert-Paare gespeichert sind.

```
<?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>
```

Generierte Mapping-Ausgabe

6.2 Stapel-Verarbeitung von Dateien

Sie können MapForce so konfigurieren, dass das Programm bei der Ausführung des Mappings mehrere Dateien (z.B. alle Dateien in einem Verzeichnis) verarbeitet. Mit Hilfe dieser Funktion können Sie die folgenden Aufgaben durchführen:

- Bereitstellen einer Liste von Input-Dateien, die vom Mapping verarbeitet werden sollen
- Generieren einer Liste von Dateien anstelle einer einzigen Ausgabedatei als Mapping-Ausgabe
- Generieren einer Mapping-Applikation, in der sowohl die Namen der Input- als auch die der Output-Dateien zur Laufzeit definiert werden
- Konvertieren einer Gruppe von Dateien in ein anderes Format
- Aufteilen einer großen Datei in kleinere Teile
- Zusammenführen mehrerer Dateien in einer großen Datei

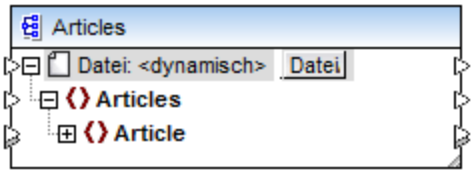
Sie können eine MapForce-Komponente so konfigurieren, dass mehrere Dateien auf eine der folgenden Arten verarbeitet werden:

- Bereitstellung des Pfads zur/zu den gewünschten Input- oder Output-Datei(en) mit Hilfe von Platzhalterzeichen anstelle eines festgelegten Dateinamens in den Komponenteneinstellungen (siehe Ändern der Komponenteneinstellungen 38). Sie können im Dialogfeld "Komponenteneinstellungen" die Platzhalterzeichen * und ? verwenden, sodass MapForce bei der Ausführung des Mappings den entsprechenden Pfad auflöst.
- Verbinden mit dem Root-Node einer Komponente einer Sequenz, die den Pfad dynamisch bereitstellt (z.B. das Ergebnis der Funktion `replace-fileext`). MapForce liest bei der Ausführung des Mappings alle Input-Dateien dynamisch bzw. generiert alle Output-Dateien dynamisch.

Je nachdem, welches Ergebnis Sie erzielen möchten, können Sie im selben Mapping entweder eine oder beide dieser Methoden verwenden. Es ist jedoch nicht sinnvoll, beide Methoden gleichzeitig für dieselbe Komponente zu verwenden. Um diese Methode für eine bestimmte Komponente auszuwählen, klicken Sie auf die Schaltfläche **Datei** (Datei) oder **Datei/String** (Datei/String) neben dem Root-Node einer Komponente. Über diese Schaltfläche können Sie die folgenden Einstellungen vornehmen:

Dateinamen aus Komponenteneinstellungen verwenden	Wenn die Komponente eine oder mehrere Instanzdateien verarbeiten soll, verarbeitet MapForce bei Auswahl dieser Option den/die im Dialogfeld "Komponenteneinstellungen" definierten Dateinamen. Wenn Sie diese Option auswählen, hat der Root-Node keinen Input-Konnektor, da dieser keine Bedeutung hat. Wenn Sie im Dialogfeld "Komponenteneinstellungen" noch keine Input- oder Output-Datei definiert haben, lautet der

 © 2019-2025 Altova GmbH

	Name des Root-Node **Datei: (Standard)**. Andernfalls wird im Root-Node der Name der Input-Datei, gefolgt von einem Semikolon (;), gefolgt vom Namen der Output-Datei angezeigt. Wenn der Name der Input-Datei mit dem der Output-Datei identisch ist, wird er als Name des Root-Node angezeigt. Beachten Sie, dass Sie entweder diese Option oder die Option *Über das Mapping bereitgestellte dynamische Dateinamen verwenden* verwenden können.
Über das Mapping bereitgestellte dynamische Dateinamen verwenden	Bei Auswahl dieser Option verarbeitet MapForce den/die im Mapping-Bereich definierten Dateinamen, indem das Programm Werte mit dem Root-Node der Komponente verbindet. Bei Auswahl dieser Option erhält der Root-Node einen Input-Konnektor, mit dem Sie Werte verbinden können, die die zu verarbeitenden Dateinamen während der Mapping-Ausführung dynamisch bereitstellen. Wenn Sie auch im Dialogfeld "Komponenteneinstellungen" Dateinamen definiert haben, so werden diese Werte ignoriert. Wenn diese Option ausgewählt ist, wird als Name des Root-Node **Datei: <dynamisch>** angezeigt.  Diese Option kann nicht gleichzeitig mit der Option *Dateinamen aus Komponenteneinstellungen verwenden* verwendet werden.

Für die folgenden Komponenten können mehrere Input- oder Output-Dateien definiert werden:

- XML-Dateien
- Textdateien (CSV-*, FLF-* und FlexText**-Dateien)

- EDI-Dokumente**
- Excel-Arbeitsblätter**
- XBRL-Dokumente**
- JSON-Dateien**
- Protocol Buffer-Dateien**

** MapForce Professional Edition erforderlich*
*** MapForce Enterprise Edition erforderlich*

In der folgenden Tabelle finden Sie Informationen über die Unterstützung für dynamische Input- und Output-Dateien und Platzhalter in MapForce-Sprachen.

Zielsprache	Dynamischer Input-Dateiname	Unterstützung von Platzhaltern für Input-Dateinamen	Dynamischer Output-Dateiname
XSLT 1.0	*	Nicht von XSLT 1.0 unterstützt	Nicht von XSLT 1.0 unterstützt
XSLT 2.0	*	*(1)	*
XSLT 3.0	*	*(1)	*
C++	*	*	*
C#	*	*	*
Java	*	*	*
BUILT-IN	*	*	*

Legende:

*	Unterstützt
(1)	Für XSLT 2.0, XSLT 3.0 und XQuery wird die Funktion `fn:collection` verwendet. In der Implementierung im Altova XSLT 2.0-, XSLT 3.0- und XQuery-Prozessor werden Platzhalter aufgelöst. Andere Prozessoren verhalten sich eventuell anders.

6.2.1 Beispiel: Aufteilen einer XML-Datei in mehrere

In diesem Beispiel wird gezeigt, wie Sie anhand einer einzigen XML-Quelldatei dynamisch mehrere XML-Dateien generieren. Sie finden die Mapping-Beispieldatei dazu unter dem folgenden Pfad: **<Dokumente>\Altova\MapForce2025\MapForceExamples\Tut-ExpReport-dyn.mfd**.

Die XML-Quelldatei (die im selben Ordner wie das Mapping liegt) besteht aus der Spesenabrechnung für eine Person namens Fred Landis und enthält fünf Spesenposten (expense-item) unterschiedlichen Typs. Ziel dieses Beispiels ist es, für jeden der unten aufgelisteten Spesenposten eine separate XML-Datei zu generieren.

 © 2019-2025 Altova GmbH

Person	
First	Fred
Last	Landis
Title	Project Manager
Phone	123-456-78
Email	f.landis@nanonull.com

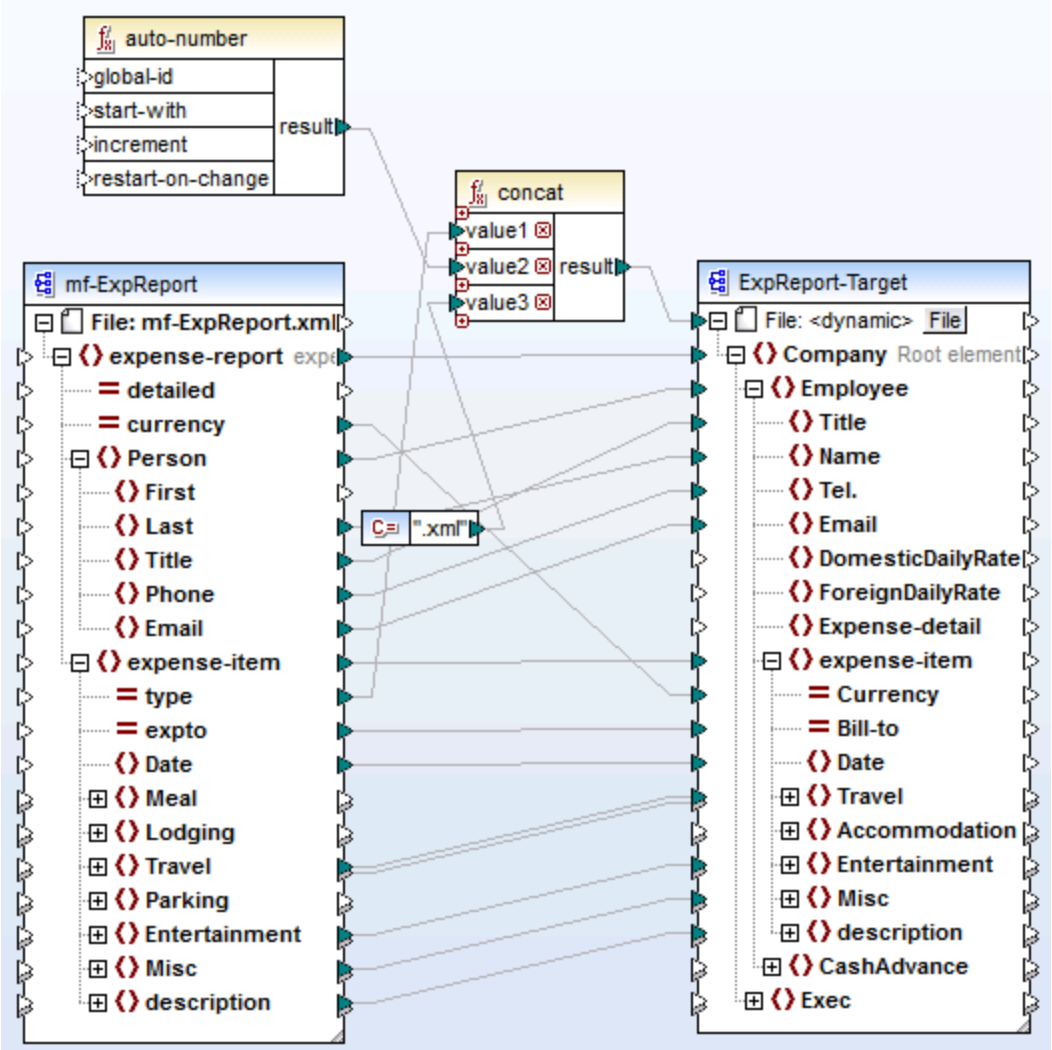
expense-item (5)

	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 (in der XMLSpy Grid-Ansicht)

Da das Attribut "type" den jeweiligen Spesentyp definiert, ist dies das Datenelement, das wir zum Aufteilen der Quelldatei verwenden werden. Gehen Sie dazu folgendermaßen vor:

1. Fügen Sie eine `concat`-Funktion ein (Sie können diese mit der Maus aus der Bibliothek **core | string functions** in den Mapping-Bereich ziehen).
2. Fügen Sie (über das Menü **Einfügen | Konstante**) eine Konstante ein und geben Sie als ihren Wert ".xml" ein.
3. Ziehen Sie die Funktion **`auto-number`** aus der Bibliothek **core | generator functions** mit der Maus in den Mapping-Bereich.
4. Klicken Sie auf die Schaltfläche **Datei** (Datei) oder **Datei/String** (Datei/String) der Zielkomponente und wählen Sie den Befehl **Über das Mapping bereitgestellte dynamische Dateinamen verwenden**.
5. Erstellen Sie die Verbindungen wie oben gezeigt und klicken Sie auf das Register "Ausgabe", um das Ergebnis des Mappings zu sehen.



Tut-ExpReport-dyn.mfd (MapForce Basic Edition)

Beachten Sie, dass die erzeugten Ausgabedateien folgendermaßen dynamisch benannt werden:

- Das Attribut `type` liefert den ersten Teil des Dateinamens, z.B. Travel.
- Die Funktion `auto-number` liefert die fortlaufend nummerierten Dateinummern (z.B. "Travel1", "Travel2", usw.)
- Die Konstante liefert die Dateierweiterung, d.h. .xml, also lautet der Dateiname der ersten Datei Travel1.xml.

 © 2019-2025 Altova GmbH

7 Automatisieren mit Altova-Produkten

Mit MapForce erstellte Mappings können von den folgenden (separate lizenzierten) Altova-Transformationsprozessoren in einer Server-Umgebung (u.a. auch mit Linux und macOS-Servern) und mit der Leistung eines Servers ausgeführt werden:

- *RaptorXML Server*. Dieser Prozessor eignet sich für die Ausführung des Mappings, wenn die Transformationssprache des Mappings XSLT 1.0, XSLT 2.0, XSLT 3.0 oder XQuery ist. Siehe Automatisierung mit RaptorXML Server 428.
- *MapForce Server (oder MapForce Server Advanced Edition).* Dieser Prozessor eignet sich für Mappings, deren Transformationssprache BUILT-IN* ist. Die Sprache BUILT-IN unterstützt die meisten Mapping-Funktionen in MapForce, während MapForce Server (und v.a. die MapForce Server Advanced Edition) die höchste Leistung für die Verarbeitung eines Mappings bietet.

** Für die Transformationssprache BUILT-IN wird MapForce Professional oder Enterprise Edition benötigt.*

Zusätzlich dazu bietet MapForce die Möglichkeit, die Generierung von XSLT-Code über die Befehlszeile zu automatisieren. Nähere Informationen dazu finden Sie unter MapForce-Befehlszeilenschnittstelle 429.

7.1 Automatisierung mit RaptorXML Server

Altova RaptorXML Server (in der Folge als RaptorXML bezeichnet) ist Altovas ultraschneller XML- und XBRL-Prozessor der dritten Generation, der für die neuesten Standards und parallele Rechnerumgebungen optimiert wurde. RaptorXML lässt sich plattformübergreifend einsetzen und ermöglicht dank der Nutzung moderner Multi-Core Computer die ultraschnelle Verarbeitung von XML- und XBRL-Daten.

RaptorXML steht in mehreren Editionen zur Verfügung, die von der Altova Download-Seite heruntergeladen und anschließend installiert werden können (https://www.altova.com/de/download-trial-server.html):

- RaptorXML Server ist ein sehr schneller Prozessor zur Verarbeitung von XML, der XML, XML-Schema, XSLT, XPath, XQuery und mehr.
- RaptorXML+XBRL Server unterstützt alle Funktionen von RaptorXML Server und kann zusätzlich XBRL-Daten verarbeiten und validieren.

Beim Generieren von Code in XSLT generiert MapForce eine Batch-Datei namens **DoTransform.bat**, die in dem bei der Generierung gewählten Ausgabeordner gespeichert wird. Die Batch-Datei ruft RaptorXML Server auf und führt die XSLT-Transformation auf dem Server aus.

Anmerkung: Sie können mit dem Built-in-Ausführungsprozessor eine Vorschau des XSLT-[65] Codes anzeigen.

© 2019-2025 Altova GmbH

7.2 MapForce-Befehlszeilenschnittstelle

Die allgemeine Syntax eines MapForce-Befehls in der Befehlszeile lautet:

```
MapForce.exe <dateiname> [/{ziel} [[<ausgabeverz>] [/optionen]]]
```

Nähere Informationen zu den einzelnen Parametern des Befehls finden Sie weiter unten unter *Parameter*.

Notation
Die Befehlszeilensyntax wird folgendermaßen notiert:

Notation	Beschreibung
Text ohne runde oder geschweifte Klammern	Elemente, die Sie so, wie angezeigt, eingeben müssen
`<Text innerhalb von spitzen Klammern>`	Platzhalter, für die Sie einen Wert angeben müssen
`[Text innerhalb von eckigen Klammern]`	Optionale Elemente
`{Text innerhalb von geschweiften Klammern}`	Eine Gruppe zwingend erforderlicher Elemente; wählen Sie eines aus
Pipe-Zeichen (`|`)	Trennzeichen für einander gegenseitig ausschließende Elemente; wählen Sie eines aus
Auslassungszeichen (`...`)	Elemente, die wiederholt werden können

Parameter
`<dateiname>`
Das Mapping-Design (**`.mfd`**) oder Mapping-Projekt (**`.mfp`**) (*Professional und Enterprise Edition*), anhand dessen Code generiert werden soll.

`/{ziel}`
Definiert die Zielsprache oder Zielumgebung, für die Code generiert werden soll. Es werden die folgenden Codegenerierungsziele unterstützt.

- `/XSLT`
- `/XSLT2`
- `/XSLT3`

`<ausgabeverz>`
Optionaler Parameter, mit dem das Ausgabeverzeichnis definiert wird. Wenn kein Ausgabepfad angegeben wird, wird das aktuelle Arbeitsverzeichnis verwendet. Beachten Sie, dass relative Dateipfade relativ zum aktuellen Arbeitsverzeichnis sind.

`/optionen`

Die `/optionen` schließen einander nicht gegenseitig aus. Sie können eine oder mehrere der folgenden Optionen definieren:

- Die Option `/GLOBALRESOURCEFILE <Dateiname>` kann verwendet werden, wenn im Mapping zum Auflösen von Input- oder Output-Datei- oder -Ordnerpfaden oder -Datenbanken globale Ressourcen verwendet werden. Nähere Informationen dazu finden Sie unter Globale Altova-Ressourcen 432. Die Option `/GLOBALRESOURCEFILE` definiert den Pfad zu einer XML-Datei für globale Ressourcen. Wenn `/GLOBALRESOURCEFILE` definiert ist, muss auch `/GLOBALRESOURCECONFIG` definiert sein.

- Mit der Option `/GLOBALRESOURCECONFIG <Konfig.>` wird der Name der XML-Datei für globale Ressourcen definiert (*siehe auch vorhergehende Option*). Wenn `/GLOBALRESOURCEFILE` definiert ist, muss auch `/GLOBALRESOURCECONFIG` definiert sein.

- Die Option `/LOG <logdateiname>` generiert eine Log-Datei unter dem angegebenen Pfad. `<logdateiname>` kann ein absoluter Pfad sein. Wenn ein vollständiger Pfad angegeben wird, muss das Verzeichnis für die zu generierende Log-Datei vorhanden sein. Wenn Sie nur den Dateinamen angeben, wird die Datei in das aktuelle Verzeichnis der Windows-Befehlszeile geschrieben.

Anmerkungen

- Relative Pfade sind relativ zum Arbeitsverzeichnis, welches das aktuelle Verzeichnis der Applikation ist, die MapForce aufruft. Dies gilt für den Pfad der **`.mfd`**-Datei, das Ausgabeverzeichnis, den Log-Dateinamen und die globale Ressourcendatei.
- Verwenden Sie in der Befehlszeile am Ende nicht den umgekehrten Schrägstrich und das schließende Anführungszeichen (z.B., **`"C:\My directory\"`**). Diese beiden Zeichen werden vom Befehlszeilenparser als Literalzeichen, d.h. als doppeltes Anführungszeichen interpretiert. Es wird empfohlen keine Leerzeichen und Anführungszeichen zu verwenden. Verwenden Sie den doppelten umgekehrten Schrägstrich \\, wenn in der Befehlszeile Leerzeichen vorkommen und Sie die Anführungszeichen (**`"c:\Mein Verzeichnis\\"`**) benötigen.

Beispiele

1) Um MapForce zu starten, und das Mapping `<dateiname>.mfd` zu öffnen, verwenden Sie:

```
MapForce.exe <Dateiname>.mfd
```

2) Um XSLT 2.0-Code zu generieren und auch eine Log-Datei mit dem Namen `<logdateiname>` zu erstellen, verwenden Sie:

```
MapForce.exe <dateiname>.mfd /XSLT2 <ausgabeverz> /LOG <logdateiname>
```

3) Um unter Verwendung der globalen Ressourcenkonfiguration `<grkonfigname>` aus der globalen Ressourcendatei `<grdateiname>` XSLT 2.0-Code zu generieren, verwenden Sie:

```
Mapforce.exe <dateiname>.mfd /XSLT2 <ausgabeverz> /GLOBALRESOURCEFILE
<grdateiname> /GLOBALRESOURCECONFIG <grkonfigname>
```

Beispiele für die Professional und die Enterprise Edition

1) Um eine C#-Applikation für Visual Studio 2022 zu generieren und eine Log-Datei zu erstellen, verwenden Sie:

```
MapForce.exe <dateiname>.mfd /CS:VS2022 <ausgabeverz> /LOG <logdateiname>
```

 © 2019-2025 Altova GmbH

2) Um unter Verwendung der in **Extras | Optionen** definierten Codegenerierungseinstellungen eine C++-Applikation zu generieren und eine Log-Datei zu erstellen, verwenden Sie:

```
MapForce.exe <dateiname>.mfd /CPP <ausgabeverz> /LOG <logdateiname>
```

3) Um eine C++-Applikation für Visual Studio 2022, MSXML, mit statischen Bibliotheken, MFC-Unterstützung und ohne Log-Datei zu generieren, verwenden Sie:

```
MapForce.exe <Dateiname>.mfd /CPP:VS2022,MSXML,LIB,MFC
```

4) Um eine C++-Applikation für Visual Studio 2022, Xerces, mit statischen Bibliotheken, keine MFC-Unterstützung und ohne Log-Datei zu generieren, verwenden Sie:

```
MapForce.exe <dateiname>.mfd /CPP:VS2022,XERCES,DLL,NoMFC <ausgabeverz> /LOG
<logdateiname>
```

5) Um eine Java-Applikation zu generieren und eine Log-Datei zu erstellen, verwenden Sie:

```
MapForce.exe <dateiname>.mfd /JAVA <ausgabeverz> /LOG <logdateiname>
```

6) Um unter Verwendung der Sprache und des Ausgabeverzeichnisses, die/das in den Ordnereinstellungen (für die einzelnen Ordner im Projekt) definiert ist, für alle Mappings im Projekt Code zu generieren, verwenden Sie:

```
MapForce.exe <dateiname>.mfp /GENERATE /LOG <logdateiname>
```

7) Um für alle Mappings in der Projektdatei Java-Code zu generieren, verwenden Sie:

```
MapForce.exe <dateiname>.mfp /JAVA /LOG <logdateiname>
```

Beachten Sie, dass die in den Ordnereinstellungen definierte Codegenerierungssprache ignoriert wird und für alle Mappings Java verwendet wird.

8) Um in der Befehlszeile für ein zuvor kompiliertes Java-Mapping Input- und Output-Dateien anzugeben, verwenden Sie:

```
java -jar <mappingdatei>.jar /InputFileName <inputdateiname> /OutputFileName
<outputdateiname>
```

Die `/InputFileName` und `/OutputFileName` Parameter sind die Namen spezieller Input-Komponenten im MapForce Mapping, die Ihnen gestatten, Parameter in der Befehlszeilenausführung zu verwenden (siehe Bereitstellen von Parametern für das Mapping 164).

9) Um ein Mapping zu einer MapForce Server-Ausführungsdatei für die MapForce Server Version 2025 zu kompilieren und XML-Signaturen zu unterdrücken, verwenden Sie:

```
MapForce.exe <dateiname>.mfd /COMPILE:NOXMLSIGNATURES
<ausgabeverz> /MFXVERSION:2025 /LOG <logdateiname>
```

8 Globale Altova-Ressourcen

Globale Altova-Ressourcen sind Aliasse für Datei-, Ordner und Datenbankressourcen. Jeder Alias kann mehrere Konfigurationen haben, wobei jede Konfiguration genau einer Ressource zugeordnet wird. Wenn Sie daher eine globale Ressource verwenden, können Sie zwischen ihren Konfigurationen wechseln. So könnten Sie etwa eine Datenbank-Ressource mit zwei Konfigurationen erstellen: `Entwicklung` und `Produktion`. Je nachdem, was Sie bezwecken möchten, können Sie zwischen diesen Konfiguration wechseln.

Globale Ressourcen können applikationsübergreifend in verschiedenen Altova Applikationen verwendet werden (*siehe Unterabschnitt weiter unten*).

Globale Ressourcen in anderen Altova-Produkten

Wenn Datei-, Ordner- und Datenbankverbindungsinformationen als globale Ressourcen gespeichert werden, lassen sich diese in mehreren Altova-Applikationen wiederverwenden. Wenn Sie ein und dieselbe Datei z.B. häufig in verschiedenen Altova Desktop-Applikationen öffnen müssen, können Sie diese als globale Ressource definieren. Wenn Sie den Dateipfad ändern müssen, muss er nur an einer einzigen Stelle geändert werden. Derzeit können globale Ressourcen in den folgenden Altova-Produkten definiert und verwendet werden:

- Altova Authentic
- DatabaseSpy
- MobileTogether Designer
- MapForce
- StyleVision
- XMLSpy
- FlowForce Server
- MapForce Server
- RaptorXML Server/RaptorXML+XBRL Server

In diesem Abschnitt

In diesem Abschnitt wird erläutert, wie Sie verschiedene Arten von globalen Ressourcen erstellen und konfigurieren. Dieser Abschnitt ist in die folgenden Kapitel gegliedert:

- Einrichten globaler Ressourcen Teil 1 433
- Einrichten globaler Ressourcen Teil 2 435
- XML-Dateien als globale Ressourcen 438
- Ordner als globale Ressourcen 440

 © 2019-2025 Altova GmbH

8.1 Einrichten globaler Ressourcen Teil 1

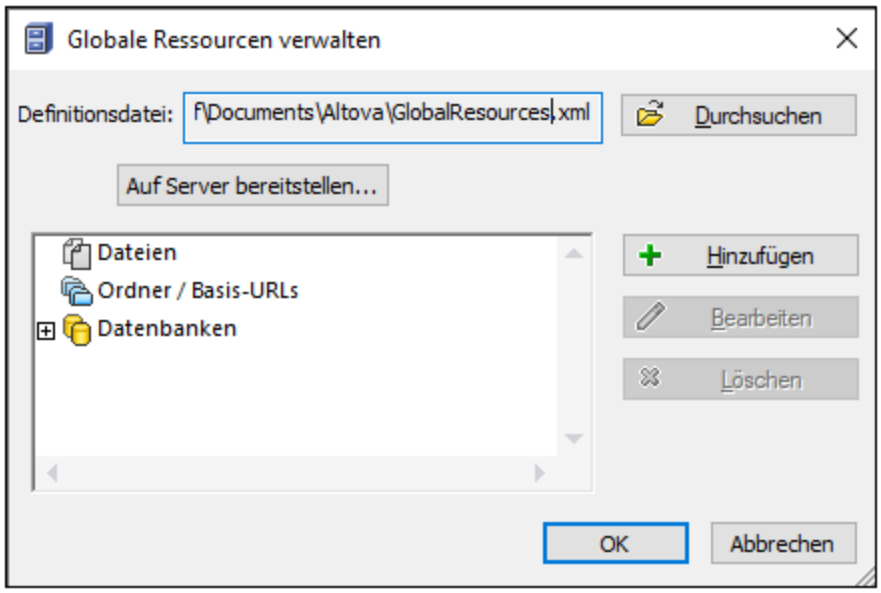
Die Konfiguration globaler Ressourcen besteht aus zwei Teilen: (i) dem Erstellen einer globalen Ressource im Dialogfeld **Globale Ressourcen verwalten** (*siehe unten*) und (ii) dem Definieren der Eigenschaften dieser globalen Ressource im Dialogfeld **Globale Ressource**. Der zweite Teil wird im nächsten Kapitel 435 behandelt.

Globale Altova-Ressourcen werden im Dialogfeld **Globale Ressourcen verwalten** definiert. Dieses Dialogfeld kann auf zwei Arten aufgerufen werden:

- Wählen Sie den Menübefehl **Extras | Globale Ressourcen**.
- Klicken Sie in der Symbolleiste "Globale Ressourcen" auf das Symbol **Globale Ressourcen verwalten** (*Abbildung unten*).

Die Definitionsdatei für globale Ressourcen

Die Informationen über globale Ressourcen, werden in einer XML-Datei, der Definitionsdatei für globale Ressourcen, gespeichert. Diese Datei wird erstellt, sobald die erste globale Ressource im Dialogfeld **Globale Ressourcen verwalten** (*Abbildung unten*) definiert und gespeichert wird.



Wenn Sie das Dialogfeld **Globale Ressourcen verwalten** zum ersten Mal öffnen, wird der Standardpfad und -name der Definitionsdatei für globale Ressourcen im Textfeld *Definitionsdatei* (*siehe Abbildung oben*) definiert:

```
C:\Benutzer\<Benutzername>\Dokumente\Altova\GlobalResources.xml.
```

Diese Datei ist bei allen Altova-Applikationen als Standard-Definitionsdatei für globale Ressourcen definiert. Eine globale Ressource kann von einer beliebigen Altova-Applikation aus in dieser Datei gespeichert werden und steht dann allen anderen Altova-Applikationen sofort als globale Ressource zur Verfügung. Um eine globale Ressource zu definieren und in der Definitionsdatei für globale Ressourcen zu speichern, fügen Sie die globale

Ressource im Dialogfeld **Globale Ressourcen verwalten** hinzu und klicken Sie auf **OK**.

Um eine bereits vorhandene Definitionsdatei für globale Ressourcen als aktive Definitionsdatei einer bestimmten Altova-Applikation auszuwählen, navigieren Sie über die Schaltfläche **Durchsuchen** des Textfelds *Definitionen* zu dieser Datei (*siehe Abbildung oben*).

Über das Dialogfeld **Globale Ressourcen verwalten** können Sie vorhandene globale Ressourcen auch bearbeiten und löschen.

Anmerkungen:

- Sie können der Definitionsdatei für globale Ressourcen jeden beliebigen Namen geben und ihn in einem beliebigen Ordner, auf den Ihre Altova-Applikationen Zugriff haben, speichern. Sie müssen diese Datei in Ihrer Applikation nur (im Textfeld *Definitionen*) als die Definitionsdatei für globale Ressourcen für die jeweilige Applikation definieren. Die Ressourcen lassen sich in allen Altova-Produkten als globale Ressourcen verwenden, wenn Sie in allen Altova-Produkten eine einzige Definitionsdatei verwenden.

- Sie können auch mehrere Definitionsdateien für globale Ressourcen erstellen. Es kann aber immer nur eine davon in einer Altova-Applikation aktiv sein und nur die in dieser Datei enthaltenen Definitionen stehen in der Applikation zur Verfügung. Sie können dadurch je nach Bedarf festlegen, welche Ressourcen nur eingeschränkt und welche in mehreren Produkten zur Verfügung stehen sollen.

© 2019-2025 Altova GmbH

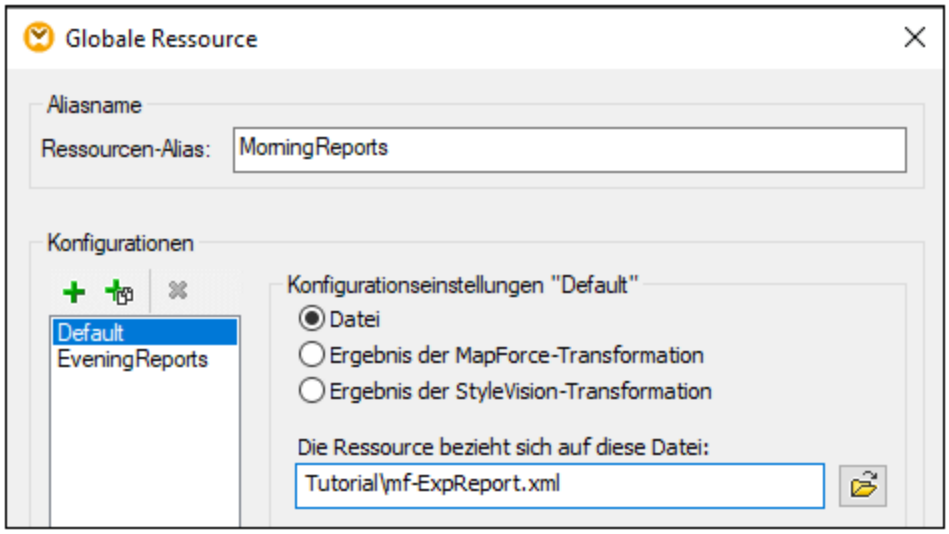
8.2 Einrichten globaler Ressourcen Teil 2

Im zweiten Teil der Konfiguration der globalen Ressourcen werden die Eigenschaften einer globalen Ressource im Dialogfeld **Globale Ressource** definiert. Die Eigenschaften sind von der Art der globalen Ressource abhängig (*siehe Unterabschnitte weiter unten*). Sie können das Dialogfeld **Globale Ressource** durch Klick auf die Schaltfläche **Hinzufügen** im Dialogfeld "Globale Ressourcen verwalten" 433 aufrufen.

Nähere Informationen über das Einrichten verschiedener Arten von globalen Ressourcen finden Sie in den folgenden Beispielen: XML-Dateien als globale Ressourcen 438, Ordner als globale Ressourcen 440.

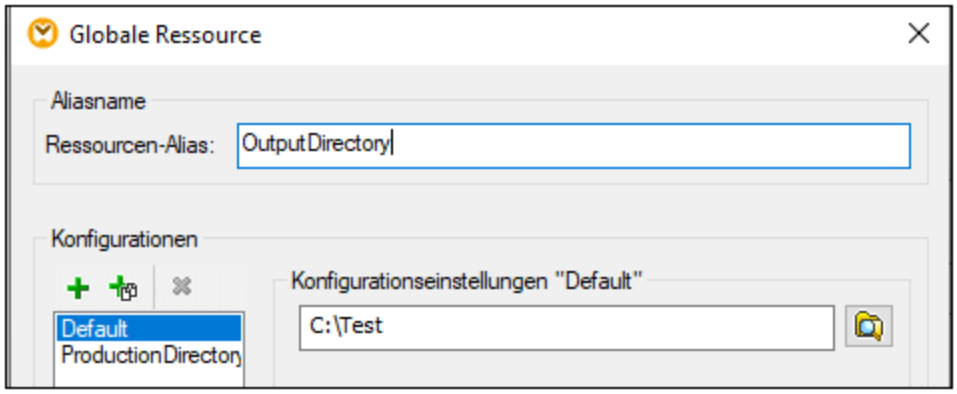
Dateien

Die dateispezifischen Eigenschaften werden im Dialogfeld **Globale Ressource** unten angezeigt. Die Konfiguration besteht aus den folgenden drei Hauptteilen: (i) dem Namen der Datei, (ii) dem Pfad dieser Datei und (iii) der Liste der für diesen Dateialias definierten Konfigurationen.



Ordner

Die ordnerspezifischen Eigenschaften werden im Dialogfeld **Globale Ressource** unten angezeigt. Die Konfiguration besteht aus den folgenden drei Hauptbereichen: (i) dem Namen des Ordners, (ii) dem Pfad dieses Ordners und (iii) der Liste der für diesen Ordneralias definierten Konfigurationen.



Schaltflächen im Dialogfeld "Globale Ressource"

	Konfiguration hinzufügen: Ruft das Dialogfeld "Konfiguration hinzufügen" auf, in das Sie den Namen der hinzuzufügenden Konfiguration eingeben.
	Konfiguration als Kopie hinzufügen: Ruft das Dialogfeld "Konfiguration hinzufügen" auf, in das Sie den Namen der Konfiguration, die als Kopie der ausgewählten Konfiguration erstellt werden soll, eingeben können.
	Löschen: Löscht die ausgewählte Konfiguration.
	Öffnen: Damit können Sie zur Datei navigieren, die als globale Ressource erstellt werden soll.
	Öffnen: Damit können Sie zum Ordner navigieren, der als globale Ressource erstellt werden soll.

Globale Ressourcenkonfiguration: Allgemeine Verfahren

Im Folgenden wird das Erstellen und Konfigurieren von globalen Ressourcen in groben Zügen beschrieben.

1. Klicken Sie auf die Symbolleisten-Schaltfläche (**Globale Ressourcen verwalten**). Gehen Sie alternativ dazu zum Menü **Extras** und klicken Sie auf **Globale Ressourcen**.
2. Klicken Sie auf **Hinzufügen** und wählen Sie den gewünschten Ressourcentyp aus (Datei, Ordner, Datenbank). Daraufhin wird das Dialogfeld **Globale Ressource** angezeigt.
3. Geben Sie in das Textfeld **Ressourcen-Alias** einen beschreibenden Namen ein (z.B. `Input-Datei`).
4. Die Konfiguration der Standardkonfiguration ist von der Art der globalen Ressource abhängig: (i) Navigieren Sie im Fall einer Datei oder eines Ordners zur Datei bzw. zum Ordner, auf die die Ressource standardmäßig verweisen soll; (ii) klicken Sie im Fall einer Datenbankverbindung auf **Datenbank auswählen** und befolgen Sie die Anweisungen des Datenbankverbindungsassistenten,

 © 2019-2025 Altova GmbH

um eine Verbindung zur Datenbank herzustellen . Diese Datenbankverbindung wird standardmäßig verwendet, wenn Sie das Mapping ausführen.

5. Wenn Sie eine weitere Konfiguration benötigen (z.B. einen zusätzlichen Ausgabeordner), klicken Sie im Dialogfeld **Globale Ressource** auf die Schaltfläche , geben Sie den Namen dieser Konfiguration ein und definieren Sie den Pfad zu dieser Konfiguration.
6. Wiederholen Sie den vorherigen Schritt für jede weitere benötigte Konfiguration.

Anmerkung: Datenbankverbindungen als globale Ressourcen werden nur in der MapForce Professional und Enterprise Edition unterstützt.

8.3 XML-Dateien als globale Ressourcen

In diesem Kapitel wird beschrieben, wie Sie XML-Dateien als globale Ressourcen verwenden. In manchen Fällen muss eine XML-Input-Datei mehrmals pro Tag geändert werden. Zum Beispiel, wenn Sie jeden Morgen ein bestimmtes Mapping ausführen müssen, um anhand einer bestimmten XML-Datei als Mapping Input einen Bericht zu generieren, derselbe Bericht aber jeden Abend anhand einer anderen XML-Datei generiert werden muss. Anstatt das Mapping mehrmals pro Tag zu bearbeiten (oder mehrere Kopien des Mappings zu verwenden), könnten Sie das Mapping so konfigurieren, dass eine als globale Ressource definierte Datei (ein so genannter *Datei-Alias*) ausgelesen wird. Unser Datei-Alias hat in diesem Beispiel zwei Konfigurationen:

1. mit der `Default`-Konfiguration wird eine XML-Datei für die Morgenausführung als Mapping Input bereitgestellt.
2. mit der Konfiguration `EveningReports` wird eine XML-Datei für die Abendausführung als Mapping Input bereitgestellt.

Um den Datei-Alias zu erstellen und zu konfigurieren, gehen Sie vor, wie unten beschrieben.

Schritt 1: Erstellen einer globalen Ressource

Zuerst muss ein Datei-Alias erstellt werden. Gehen Sie folgendermaßen vor:

1. Klicken Sie auf die Symbolleisten-Schaltfläche (**Globale Ressourcen verwalten**). Gehen Sie alternativ dazu zum Menü **Extras** und klicken Sie auf **Globale Ressourcen**.
2. Klicken Sie auf **Hinzufügen | Datei** und geben Sie in das Textfeld **Ressourcen-Alias** einen Namen ein. In diesem Beispiel nennen wir unsere Standardkonfiguration `MorningReports`.
3. Klicken Sie neben dem Textfeld **Die Ressource bezieht sich auf diese Datei** auf die **Durchsuchen**-Schaltfläche und wählen Sie die Datei **`Tutorial\mf-ExpReport.xml`** aus.
4. Klicken Sie im Abschnitt **Konfigurationen** auf und geben Sie der zweiten Konfiguration den Namen `EveningReports`.
5. Klicken Sie auf **Durchsuchen** und wählen Sie die Datei **`Tutorial\mf-ExpReport2.xml`**.

Schritt 2: Verwenden der globalen Ressource im Mapping

Wir können die neu erstellte globale Ressource nun in unserem Mapping verwenden. Damit das Mapping die Daten aus der globalen Ressource liest, gehen Sie folgendermaßen vor:

1. Öffnen Sie das Mapping **`Tutorial\Tut-ExpReport.mfd`**.
2. Doppelklicken Sie auf die Titelleiste der Quellkomponente, um das Dialogfeld **Komponenteneinstellungen** zu öffnen.
3. Klicken Sie neben **XML-Input-Datei** auf **Durchsuchen**, anschließend auf **Globale Ressourcen** und wählen Sie den Datei-Alias `MorningReports` aus. Klicken Sie auf **Öffnen**.
4. Öffnen Sie das Dialogfeld **Komponenteneinsetllungen** erneut: Als XML-Input-Datei wird nun **`altova://file_resource/MorningReports`** verwendet, d.h. für den Pfad wird eine globale Ressource verwendet.

Schritt 3: Ausführen des Mappings mit der gewünschten Konfiguration

Sie können nun die XML-Input-Datei vor Ausführung des Mappings ganz einfach wechseln:

- Um **`mf-ExpReport.xml`** als Input zu verwenden, wählen Sie den Menübefehl **Extras | Aktive Konfiguration | Default**.

 © 2019-2025 Altova GmbH

- Um **`mf-ExpReport2.xml`** als Input zu verwenden, wählen Sie den Menübefehl **Extras | Aktive Konfiguration | EveningReports**.

Alternativ dazu können Sie die gewünschte Konfiguration auch aus der Dropdown-Liste der **globalen Ressourcen** (*siehe Abbildung unten*) auswählen.

Um eine Vorschau auf das Mapping-Ergebnis mit einer der beiden Konfigurationen anzuzeigen, klicken Sie auf das Fenster **Ausgabe**.

8.4 Ordner als globale Ressourcen

In diesem Kapitel wird beschrieben, wie Sie Ordner als globale Ressourcen verwenden. Es kann vorkommen, dass dieselbe Ausgabe in verschiedenen Verzeichnissen generiert werden muss. Zu diesem Zweck erstellen wir einen Ordner-Alias mit zwei Konfigurationen:

1. Mit der Konfiguration `Default` wird die Ausgabe in **`C:\Test`** generiert.
2. Mit der Konfiguration `Production` wird die Ausgabe in **`C:\Production`** generiert.

Um den Ordner-Alias zu erstellen und zu konfigurieren, gehen Sie vor, wie unten beschrieben.

Schritt 1: Erstellen einer globalen Ressource

Zuerst muss ein Ordner-Alias erstellt werden. Gehen Sie folgendermaßen vor:

1. Klicken Sie auf die Symbolleisten-Schaltfläche (**Globale Ressourcen verwalten**). Gehen Sie alternativ dazu zum Menü **Extras** und klicken Sie auf **Globale Ressourcen**.
2. Klicken Sie auf **Hinzufügen | Ordner** und geben Sie in das Textfeld **Ressourcen-Alias** einen Namen ein. In diesem Beispiel nennen wir unsere Standardkonfiguration `OutputDirectory`.
3. Klicken Sie neben dem Textfeld **Konfigurationseinstellungen "Default"** auf die Schaltfläche **Durchsuchen** und wählen Sie **`C:\Test`** aus. Stellen Sie sicher, dass dieser Ordner auf Ihrem Rechner vorhanden ist.
4. Klicken Sie auf und geben Sie den Namen der zweiten Konfiguration ein. In diesem Beispiel nennen wir unsere zweite Konfiguration `ProductionDirectory`.
5. Klicken Sie auf **Durchsuchen** und wählen Sie den Ordner **`C:\Production`** aus. Stellen Sie sicher, dass dieser Ordner auf Ihrem Rechner vorhanden ist.

Schritt 2: Verwenden der globalen Ressource im Mapping

Im nächsten Schritt sorgen wir dafür, dass der soeben erstellte Ordner-Alias im Mapping verwendet wird. Gehen Sie folgendermaßen vor:

1. Öffnen Sie das Mapping **`Tutorial\Tut-ExpReport.mfd`**.
2. Doppelklicken Sie auf die Titelleiste der Zielkomponente, um das Dialogfeld **Komponenteneinstellungen** zu öffnen.
3. Klicken Sie auf **Globale Ressourcen** und anschließend auf **Speichern**.
4. Speichern Sie die XML-Ausgabedatei als **`Output.xml`**. Der Pfad der XML-Output-Datei lautet nun **`altova://folder_resource/OutputDirectory/Output.xml`**, was darauf hinweist, dass der Pfad als globale Ressource definiert wurde.

Schritt 3: Ausführen des Mappings mit der gewünschten Konfiguration

Sie können nun die Ausgabeordner vor Ausführung des Mappings ganz einfach wechseln:

- Um **`C:\Test`** als Ausgabeverzeichnis zu verwenden, wählen Sie den Menübefehl **Extras | Aktive Konfiguration | Default**.
- Um **`C:\Production`** als Ausgabeverzeichnis zu verwenden, wählen Sie den Menübefehl **Extras | Aktive Konfiguration | ProductionDirectory**.

© 2019-2025 Altova GmbH

Die Mapping-Ausgabe wird standardmäßig als temporäre Datei geschrieben, außer Sie haben in MapForce explizit konfiguriert, dass die Ausgabe in permanente Dateien geschrieben wird. Um MapForce so zu konfigurieren, dass permanente Dateien generiert werden, gehen Sie folgendermaßen vor:

1. Klicken Sie im Menü **Extras** auf **Optionen.**
2. Wählen Sie im Abschnitt **Allgemein** die Option **Direkt in endgültige Output-Dateien schreiben.**

9 Kataloge in MapForce

MapForce unterstützt einen Teil der OASIS XML-Kataloge. Mit Hilfe dieser Methode kann MapForce häufig verwendete Schemas (und andere Dateien) aus lokalen Benutzerordnern abrufen. Dies erhöht die Verarbeitungsgeschwindigkeit, Benutzer können dadurch auch offline arbeiten (d.h. ohne mit einem Netzwerk verbunden zu sein) und Dokumente werden leichter übertragbar (da die URIs dann nur in den Katalogdateien geändert werden müssten).

Der Katalogmechanismus in MapForce funktioniert, wie in diesem Abschnitt beschrieben:

- Funktionsweise von Katalogen 443
- Katalogstruktur in MapForce 445
- Anpassen von Katalogen 447
- Umgebungsvariablen 449

Nähere Informationen zu Katalogen finden Sie in der XML-Katalogspezifikation.

© 2019-2025 Altova GmbH

9.1 Funktionsweise von Katalogen

Mit Hilfe von Katalogen können Umleitungen sowohl zu DTDs als auch XML-Schemas definiert werden. Das Prinzip des Mechanismus ist in beiden Fällen dasselbe, doch unterscheidet er sich in einigen weiter unten beschriebenen Details.

DTDs

Kataloge dienen normalerweise dazu, einen Aufruf von einer DTD an eine lokale URI umzuleiten. Dies geschieht in der Katalogdatei durch Mappen von Public und System Identifiern auf die gewünschte lokale URI. Wenn also die **DOCTYPE**-Deklaration in einer XML-Datei gelesen wird, findet ihr Public oder System Identifier über das Katalogdatei-Mapping die gewünschte lokale Ressource.

Für gebräuchliche Schemas ist der **PUBLIC** Identifier normalerweise vordefiniert, sodass für die URI in der Katalogdatei nur der **PUBLIC** Identifier auf den korrekten lokalen Identifier gemappt werden muss. Wenn das XML-Dokument geparst wird, wird der **PUBLIC** Identifier darin gelesen. Wenn dieser Identifier in einer Katalogdatei gefunden wird, wird die entsprechende URL in der Katalogdatei nachgeschlagen und das Schema wird von dort aus gelesen. Wenn also die folgende SVG-Datei in MapForce geöffnet wird:

```
<?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>
```

Der Katalog wird nach dem **PUBLIC** Identifier dieser SVG-Datei durchsucht. Angenommen, die Katalogdatei enthält den folgenden Eintrag:

```
<catalog>
  ...
    <public publicId="-//W3C//DTD SVG 1.1//EN" uri="schemas/svg/svg11.dtd"/>
  ...
</catalog>
```

In diesem Fall gibt es einen Treffer für den **PUBLIC** Identifier, sodass der Lookup-Mechanismus für die SVG DTD an die URL **schemas/svg/svg11.dtd** umgeleitet wird; (dieser Pfad ist relativ zur Katalogdatei). Diese lokale Datei wird dann als DTD für die SVG-Datei verwendet. Wenn im Katalog kein passender Treffer für den **Public** Identifier gefunden wird, wird die URL im XML-Dokument verwendet (im SVG Beispiel oben ist dies die Internet URL **http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd**).

XML-Schemas

In MapForce können Sie eine **Umleitung zu einem XML-Schema** auch mit Hilfe von Katalogen durchführen. In der XML-Instanzdatei erfolgt die Referenz zum Schema im **xsi:schemaLocation** Attribut des Elements der obersten Ebene des XML-Dokuments. Beispiel:

```
xsi:schemaLocation="http://www.xmlspy.com/schemas/orgchart OrgChart.xsd"
```

Der Wert des **xsi:schemaLocation**-Attributs besteht aus zwei Teilen: einem Namespace-Teil (oben grün) und einem URI-Teil (markiert). Anhand des Namespace-Teils erfolgt im Katalog das Mapping auf die alternative Ressource. So erfolgt etwa im folgenden Katalogeintrag eine Umleitung der Schemareferenz oben auf ein Schema unter einem anderen Pfad.

```
<uri name="http://www.xmlspy.com/schemas/orgchart" uri="C:\MySchemas\OrgChart.xsd"/>
```

Normalerweise ist der URI-Teil des Attributwerts von **xsi:schemaLocation** ein Pfad zum aktuellen Schema. Wenn das Schema jedoch über einen Katalog referenziert wird, muss der URI-Teil nicht auf ein tatsächliches XML-Schema verweisen, muss aber vorhanden sein, damit das Attribut **xsi:schemaLocation** lexikalisch gültig ist. So würde z.B. der Wert `foo` als URI-Teil des Attributwerts genügen, um gültig zu sein.

© 2019-2025 Altova GmbH

9.2 Katalogstruktur in MapForce

MapForce lädt beim Start eine Datei namens **RootCatalog.xml**(*Struktur siehe unten*), die eine Liste von Katalogdateien enthält, die durchsucht werden. Sie können diese Datei bearbeiten und beliebig viele Katalogdateien definieren, die durchsucht werden sollen. Jede davon wird in einem **nextCatalog** referenziert. Diese Katalogdateien werden durchsucht und die URIs darin werden entsprechend ihren Mappings aufgelöst.

Codefragment von RootCatalog.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<catalog xmlns="urn:oasis:names:tc:entity:xmlns:xml:catalog"
   xmlns:spy="http://www.altova.com/catalog_ext"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="urn:oasis:names:tc:entity:xmlns:xml:catalog Catalog.xsd">
  <nextCatalog catalog="%PersonalFolder%/Altova/%AppAndVersionName%/CustomCatalog.xml"/>
  <!-- Include all catalogs under common schemas folder on the first directory level -->
  <nextCatalog spy:recurseFrom="%CommonSchemasFolder%" catalog="catalog.xml"
spy:depth="1"/>
  <nextCatalog spy:recurseFrom="%ApplicationWritableDataFolder%/pkgs/.cache"
catalog="remapping.xml" spy:depth="0"/>
  <nextCatalog catalog="CoreCatalog.xml"/>
</catalog>
```

Im obigen Codefragment wird ein benutzerdefinierter Katalog namens **CustomCatalog.xml** und eine Gruppe von Katalogen, die gebräuchliche Schemas (wie z.B. W3C XML-Schemas udn das SVG-Schema) referenzieren, referenziert.

- **CustomCatalog.xml** befindet sich in Ihrem persönlichen Ordner (der über die Variable **%PersonalFolder%**) gefunden wird. erstellen. Diese Datei ist eine Skelettdatei, in der Sie Ihre eigenen Mappings erstellen können. Sie können für jedes gewünschte Schema, das nicht in den Katalogdateien im Altova Ordner "Common Schemas" enthalten ist, Mappings zu **CustomCatalog.xml** hinzufügen. Verwenden Sie dazu die unterstützten Elemente des OASIS-Katalogmechanismus (*siehe nächster Abschnitt*).
- Der durch die Variable **%CommonSchemasFolder%** definierte Ordner "Common Schemas" enthält eine Reihe gebräuchlicher Schemas. Innerhalb dieser einzelnen Schema-Ordner befindet sich eine **catalog.xml**-Datei, die Public und/oder System Identifier auf URIs mappt, die auf lokal gespeicherte Kopien des jeweiligen Schemas verweisen.
- **CoreCatalog.xml** befindet sich im MapForce-Applikationsordner und dient zum Auffinden von Schemas und Stylesheets, die von MapForce-spezifischen Prozessen wie z.B. StyleVision Power Stylesheets, anhand derer die Altova-Authentic-Ansicht von XML-Dokumenten generiert wird.

Pfadvariablen

Die in **RootCatalog.xml** verwendeten Variablen (*Codefragment oben*) haben die folgenden Werte:

%PersonalFolder%	Der persönliche Ordner des aktuellen Benutzers, z.B. C:\Benutzer\<Name>\Dokumente
%CommonSchemasFolder%	C:\ProgramData\Altova\Common2025\Schemas
%ApplicationWritableDataFolder%	C:\ProgramData\Altova

Speicherpfad von Katalogdateien und Schemas
Beachten Sie die Pfade der verschiedenen Katalogdateien.

- Die Dateien **`RootCatalog.xml`** und **`CoreCatalog.xml`** befinden sich im MapForce Applikationsordner.
- **`CustomCatalog.xml`** befindet sich in Ihrem Ordner **`EigeneDokumente\Altova\MapForce`**.
- Die **`catalog.xml`**-Dateien befinden sich jeweils in einem eigenen Schemaordner, wobei sich diese Schemaordner innerhalb des Ordners "Common Schemas" befinden:

© 2019-2025 Altova GmbH

9.3 Anpassen Ihrer Kataloge

Wenn Sie Einträge für **CustomCatalog.xml** (oder jede andere Katalogdatei, die von MapForce gelesen werden soll) erstellen, verwenden Sie nur die folgenden Elemente der OASIS-Katalogspezifikation. Jedes der unten angeführten Elemente wird mit einer Erläuterung der Attributwerte aufgelistet. Eine ausführlichere Beschreibung finden Sie in der XML Catalogs Specification. Beachten Sie, dass jedes Element das Attribut **xml:base**, mit dem die Basis-URI dieses Elements definiert wird, erhalten kann.

- <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"/>`

Beachten Sie die folgenden Punkte:

- In Fällen, in denen es keinen Public Identifier gibt, kann der System Identifier einer URL wie z.B. bei den meisten Stylesheets über das **system** Element direkt auf eine URL gemappt werden.
- Eine URI kann über das `uri` Element auf eine andere URI gemappt werden.
- Mit Hilfe der Elemente `rewriteURI` und `rewriteSystem` kann der Anfangsteil einer URI bzw. eines System Identifiers neu geschrieben werden. Dadurch kann der Anfang eines Dateipfads ersetzt werden, sodass ein anderes Verzeichnis als Ziel gewählt werden kann. Nähere Informationen zu diesen Elementen finden Sie in der XML Catalogs Specification.

Ab Release 2014 entspricht MapForce weitgehend der XML Catalogs Specification (OASIS Standard V1.1, 7 Oktober 2005). In dieser Spezifikation wird streng zwischen externen Identifier Look-ups (jenen mit einer öffentlichen ID oder einer System-ID) und URI Look-ups (URIs, die keine öffentlichen IDS oder System-IDs sind) getrennt. Namespace URIs müssen daher einfach als URIs - und nicht Public IDs oder System-IDs - behandelt werden und folglich als URI Look-ups anstelle von externen Identifier Look-ups verwendet werden. In MapForce Versionen vor Version 2014 wurden Schema Namespace URIs über **<public>** Mappings übersetzt. Ab Version 2014 müssen **<uri>** Mappings verwendet werden.

Vor v2014: `<public publicID="http://www.MyMapping.com/ref" uri="file:///C:/MyDocs/Catalog/test.xsd"/>`
Ab V-2014: `<uri name="http://www.MyMapping.com/ref" uri="file:///C:/MyDocs/Catalog/test.xsd"/>`

Wie findet MapForce ein referenziertes Schema

Ein Schema wird in einem XML-Dokument über das Attribut **xsi:scemaLocation** (*siehe unten*) referenziert. Der Wert des **xsi:schemaLocation**-Attributs besteht aus zwei Teilen: einem Namespace-Teil (grün) und einem URI-Teil (markiert).

```
xsi:schemaLocation="http://www.xmlspy.com/schemas/orgchart OrgChart.xsd"
```

Im Folgenden wird Schritt für Schritt beschrieben, wie MapForce vorgeht, um ein referenziertes Schema zu finden: Das Schema wird beim ersten erfolgreichen Schritt geladen.

1. Durchsuchen des Katalogs nach dem URI-Teil des **xsi:schemaLocation**-Werts. Wenn ein Mapping

gefunden wird - darunter auch in **rewriteURI** Mappings wird die erzeugte URI verwendet, um das Schema zu laden.

2. Durchsuchen des Katalogs nach dem Namespace-Teil des **xsi:schemaLocation**-Werts. Wenn ein Mapping gefunden wird - darunter auch in **rewriteURI** Mappings wird die erzeugte URI verwendet, um das Schema zu laden.
3. Der URI-Teil des **xsi:schemaLocation**-Werts wird zum Laden des Schemas verwendet.

XML-Schema-Spezifikationen

Die XML-Schemaspezifikationsinformationen sind in MapForce integriert und die Gültigkeit von XML-Schema-(`.xsd`)-Dokumenten wird anhand dieser internen Informationen überprüft. Daher sollte in einem XML-Schema-Dokument kein Verweis auf ein Schema, das die XML-Schema-Spezifikation definiert, vorgenommen werden.

Die Datei **catalog.xml** im Ordner `%AltovaCommonSchemasFolder%\Schemas\schema` enthält Referenzen auf DTDs, die ältere XML-Schema-Spezifikationen implementieren. Sie sollten Ihre XML-Schema-Dokumente nicht anhand dieser Schemas validieren. Zweck dieser beiden DTDs ist es einzig und allein, für die Eingabehilfen von MapForce zu Bearbeitungszwecken Informationen bereitzustellen, falls Sie Dateien gemäß diesen älteren Empfehlungen erstellen wollen.

 © 2019-2025 Altova GmbH

9.4 Umgebungsvariablen

Shell-Umgebungsvariablen können im `nextCatalog` Element verwendet werden, um den Pfad zu Systemordnern zu definieren (*siehe RootCatalog.xml-Liste oben*). Es werden die folgenden Shell-Umgebungsvariablen unterstützt:

`%PersonalFolder%`	Vollständiger Pfad zum persönlichen Ordner des aktuellen Benutzers, z.B. `C:\Benutzer\<Name>\Dokumente`
`%CommonSchemasFolder%`	`C:\ProgramData\Altova\Common2025\Schemas`
`%ApplicationWritableDataFolder%`	`C:\ProgramData\Altova`
`%AltovaCommonFolder%`	C:\Programme\Altova\Common2025
`%DesktopFolder%`	Vollständiger Pfad des Desktop-Ordners des aktuellen Benutzers.
`%ProgramMenuFolder%`	Vollständiger Pfad zum Ordner 'Programme" des aktuellen Benutzers.
`%StartMenuFolder%`	Vollständiger Pfad zum Startmenüordner des aktuellen Benutzers.
`%StartUpFolder%`	Vollständiger Pfad zum Startordner des aktuellen Benutzers.
`%TemplateFolder%`	Vollständiger Pfad des Template-Ordners des aktuellen Benutzers.
`%AdminToolsFolder%`	Vollständiger Pfad zum Dateisystemverzeichnis, in dem Verwaltungstools des aktuellen Benutzers gespeichert sind.
`%AppDataFolder%`	Vollständiger Pfad zum Ordner "Anwendungsdaten" des aktuellen Benutzers.
`%CommonAppDataFolder%`	Vollständiger Pfad zum Dateisystem, das die Applikationsdaten aller Benutzer enthält.
`%FavoritesFolder%`	Vollständiger Pfad zum Ordner 'Favoriten' des aktuellen Benutzers.
`%PersonalFolder%`	Vollständiger Pfad zum Ordner "Personal" des aktuellen Benutzers.
`%SendToFolder%`	Vollständiger Pfad zum SendTo-Ordner des aktuellen Benutzers.
`%FontsFolder%`	Vollständiger Pfad zum Ordner "Systemschriftarten".
`%ProgramFilesFolder%`	Vollständiger Pfad zum Ordner 'Programme" des aktuellen Benutzers.
`%CommonFilesFolde`	Vollständiger Pfad zum Ordner "Gemeinsame Dateien" des aktuellen Benutzers.

r%	
%WindowsFolder%	Vollständiger Pfad zum Ordner "Windows" des aktuellen Benutzers.
%SystemFolder%	Vollständiger Pfad zum Ordner "System" des aktuellen Benutzers.
%LocalAppDataFolder%	Vollständiger Pfad zum Dateisystemverzeichnis, das als Datenspeicher für lokale (nicht-Roaming) Applikationen dient.
%MyPicturesFolder%	Vollständiger Pfad zum Ordner "Meine Bilder".

© 2019-2025 Altova GmbH

10 ActiveX Integration

Die in diesem Abschnitt beschriebene MapForce-Benutzeroberfläche und deren Funktionalitäten können in benutzerdefinierte Applikationen integriert werden, die ActiveX Controls verwenden können. Mit Hilfe der ActiveX-Technologie können die verschiedensten Programmiersprachen wie z.B. C++, C# und VB.NET für die Integration verwendet werden. Alle Komponenten sind vollständige OLE Controls, Die Integration in Java wird durch Wrapper-Klassen möglich gemacht.

Um ActiveX Controls in Ihren benutzerdefinierten Code zu integrieren, müssen Sie das MapForce-Integrationspaket installieren (siehe https://www.altova.com/de/components/download). Stellen Sie sicher, dass Sie zuerst MapForce installieren und dann erst das MapForce-Integrationspaket. Je nach Sprache und Plattform gelten andere Voraussetzungen (siehe Voraussetzungen 452).

Sie haben die Wahl zwischen zwei verschiedenen Ebenen der Integration: auf Applikations- und auf Dokumentebene.

Bei einer Integration auf Applikationsebene wird die komplette Benutzeroberfläche von MapForce (einschließlich aller Menüs, Symbolleisten, Fenster usw.) als ActiveX Control in Ihre benutzerdefinierte Applikation eingebettet. So könnte Ihre benutzerdefinierte Applikation im einfachsten Szenario z.B. aus nur einem Formular bestehen, in das die grafische Benutzeroberfläche von MapForce eingebettet ist. Diese Methode ist einfacher zu implementieren als die Integration auf Dokumentebene, ist aber möglicherweise nicht geeignet, wenn Sie die grafische Benutzeroberfläche von MapForce Ihren Anforderungen gemäß flexibel konfigurieren möchten.

Bei der Integration auf Dokumentebene wird MapForce Stück für Stück in Ihre eigene Applikation eingebettet. Dabei werden nicht nur das MapForce Haupt-Control, sondern auch das Dokument-Editor-Hauptfenster und optional zusätzliche Fenster implementiert. Bei dieser Methode haben Sie größere Flexibilität beim Konfigurieren der grafischen Benutzeroberfläche, es ist aber mehr Interaktion mit den ActiveX Controls der Sprache Ihrer Wahl erforderlich.

In den Abschnitten Integration auf Applikationsebene 455 und Integration auf Dokumentebene 457 werden die grundlegenden Schritte auf diesen Ebenen beschrieben. Im Abschnitt Beispiele zur ActiveX-Integration 461 finden Sie Beispiele in C# und Java. Diese sollen Ihnen dabei helfen, rasch die richtige Entscheidung zu treffen. Der Abschnitt Objektreferenz 468 enthält eine Beschreibung aller für die Integration verwendbaren COM-Objekte mit ihren Eigenschaften und Methoden.

Informationen zur Verwendung von MapForce als Visual Studio Plug-in finden Sie unter

10.1 Voraussetzungen

Um das MapForce ActiveX Control in eine benutzerdefinierte Applikation zu integrieren, müssen die folgenden Programme auf Ihre Computer installiert sein:

- MapForce
- Das MapForce Integrationspaket, das Sie von https://www.altova.com/de/components/download herunterladen können.

Um das 64-Bit ActiveX Control zu integrieren, installieren Sie die 64-Bit-Version von MapForce und dem MapForce Integrationspaket. Für Applikationen, die mit Hilfe von Visual Studio unter der Microsoft .NET-Plattform entwickelt wurden, müssen, wie unten erläutert, die 32- und die 64-Bit-Version von MapForce und dem MapForce Integrationspaket installiert sein.

Microsoft .NET (C#, VB.NET) mit Visual Studio

Um das MapForce ActiveX Control in eine unter Microsoft .NET entwickelte 32-Bit-Applikation zu integrieren, müssen die folgenden Programme auf Ihrem Computer installiert sein:

- Microsoft .NET Framework 4.0 oder höher
- Visual Studio 2012/2013/2015/2017/2019/2022
- MapForce 32-Bit und MapForce Integrationspaket 32-Bit
- Die ActiveX Controls müssen zur Visual Studio Toolbox hinzugefügt werden (siehe Hinzufügen der ActiveX Controls zur Toolbox 454).

Falls Sie das 64-Bit ActiveX Control integrieren möchten, sind zusätzlich zu den oben erwähnten Voraussetzungen die folgenden erforderlich:

- MapForce 32-Bit und das MapForce Integrationspaket 32-Bit müssen weiterhin installiert sein (dies ist erforderlich, damit das 32-Bit ActiveX Control dem Visual Studio Designer zur Verfügung steht, da Visual Studio unter 32-Bit läuft)
- MapForce 64-Bit und das MapForce Integrationspaket 64-Bit müssen installiert sein (stellt Ihrer benutzerdefinierten Applikation zur Laufzeit das eigentliche 64-Bit ActiveX Control zur Verfügung)
- Erstellen Sie in Visual Studio eine 64-Bit Build-Konfiguration und bauen Sie Ihre Applikation mit Hilfe dieser Konfiguration. Ein Beispiel dazu finden Sie unter Ausführen der C#-Beispiellösung 461.

Java

Um das MapForce ActiveX Control über die Eclipse-Entwicklungsumgebung in die Java-Applikation zu integrieren, müssen die folgenden Programme auf Ihrem Computer installiert sein:

- Java Runtime Environment (JRE) oder Java Development Kit (JDK) 7 oder höher
- Eclipse
- MapForce und das MapForce Integrationspaket

Anmerkung: Verwenden Sie zur Ausführung der 64-Bit-Version des MapForce ActiveX Control eine 64-Bit-Version von Eclipse sowie eine 64-Bit-Version von MapForce und dem MapForce Integrationspaket.

 © 2019-2025 Altova GmbH

MapForce Integration und Bereitstellung auf Client-Rechnern

Wenn Sie eine .NET-Applikation erstellen und beabsichtigen, diese auf anderen Client-Rechnern zur Verfügung zu stellen, müssen auf dem/den Client-Rechner(n) die folgenden Programme installiert sein:

- MapForce
- Das MapForce Integrationspaket
- Der benutzerdefinierte Integrationscode oder die benutzerdefinierte Applikation.

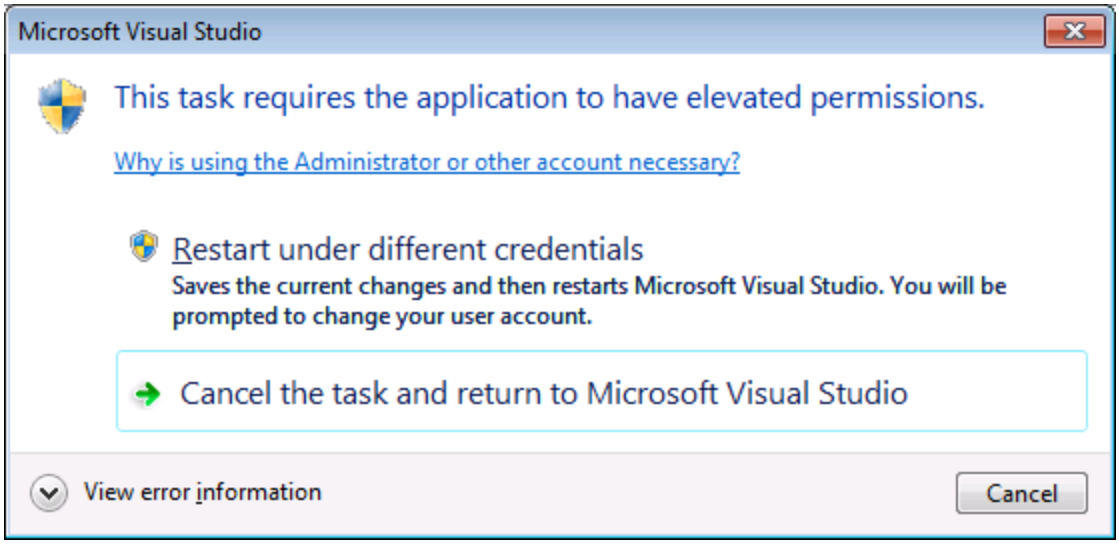
10.2 Hinzufügen der ActiveX Controls zur Toolbox

Um die MapForce ActiveX Controls in einer mit Visual Studio entwickelten Applikation verwenden zu können, fügen Sie diese folgendermaßen zu Ihrer Visual Studio Toolbox hinzu:

1. Klicken Sie im Menü **Tools** von Visual Studio auf **Choose Toolbox Items**.
2. Aktivieren Sie auf dem Register **COM Components** die Kontrollkästchen neben dem MapForceControl, dem MapForceControl-Dokument und dem MapForceControl-Platzhalter.

Falls die oben erwähnten Controls nicht zur Verfügung stehen, gehen Sie folgendermaßen vor:

1. Klicken Sie auf dem Register **COM Components** auf **Browse** und wählen Sie die Datei aus dem MapForce-Installationsordner aus. Beachten Sie, dass das MapForce-Integrationspaket installiert sein muss, da die Datei sonst nicht zur Verfügung steht (siehe Voraussetzungen 452).
3. Sobald Sie aufgefordert werden, Visual Studio mit erweiterten Berechtigungen zu starten, klicken Sie auf **Restart under different credentials**.



Wenn die obigen Schritte erfolgreich ausgeführt wurden, stehen die MapForce ActiveX Controls in der Visual Studio Toolbox zur Verfügung.

Anmerkung: Bei einer Integration auf Applikationsebene wird nur das **MapForceControl** ActiveX Control verwendet (siehe Integration auf Applikationsebene 455). Die Controls **MapForceControl Document** und **MapForceControl Placeholder** werden für die Integration auf Dokumentebene verwendet (siehe Integration auf Dokumentebene 457).

© 2019-2025 Altova GmbH

10.3 Integration auf Applikationsebene

Bei der Integration auf Applikationsebene können Sie die gesamte Benutzeroberfläche von MapForce in ein Fenster Ihrer Applikation einbetten. Bei dieser Art von Integration steht Ihnen die gesamte Benutzeroberfläche von MapForce einschließlich aller Menüs, Symbolleisten, der Statusleiste, der Dokumentfenster und Eingabehilfen zur Verfügung. Die Anpassung der Benutzeroberfläche der Applikation ist auf die Optionen eingeschränkt, die MapForce bietet. Dazu gehören die Neuanordnung und Anpassung der Größe der Eingabehilfen und die Anpassung von Menüs und Symbolleisten.

Das einzige ActiveX Control, das Sie integrieren müssen, ist `MapForceControl` [471]. Bei Integration auf Applikationsebene dürfen `MapForceControlDocument` [478] oder `MapForceControlPlaceHolder` [484] ActiveX Controls nicht instantiiert oder aufgerufen werden.

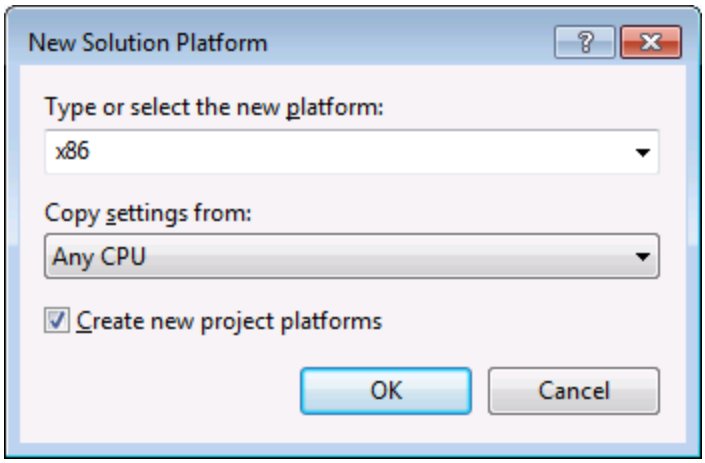
Wenn Sie Initialisierungen vornehmen müssen oder ein bestimmtes Verhalten von MapForce automatisieren wollen, verwenden Sie die für `MapForceControl` [471] beschriebenen Eigenschaften, Methoden und Events. Um komplexere MapForce Funktionsaufrufe auszuführen, sollten Sie eventuell `MapForceControl.Application` [472] verwenden.

Gehen Sie in C# oder VB.NET bei Verwendung von Visual Studio folgendermaßen vor, um eine aus einem Formular bestehende Applikation zu erstellen, in der die MapForce ActiveX Controls auf Applikationsebene integriert sind:

1. Überprüfen Sie, ob alle Voraussetzungen erfüllt werden (siehe Voraussetzungen [452]).
2. Erstellen Sie ein Visual Studio Windows Forms-Projekt mit einem neuen leeren Formular.
3. Fügen Sie die ActiveX Controls zur Toolbox hinzu, falls Sie das noch nicht getan haben (siehe Hinzufügen der ActiveX Controls zur Toolbox [454]).
4. Ziehen Sie das **MapForceControl** aus der Toolbox in Ihr neues Formular.
5. Wählen Sie das **MapForceControl** im Formular aus und definieren Sie im Fenster "Properties" für die Eigenschaft **IntegrationLevel** den Wert **ICActiveXIntegrationOnApplicationLevel**.

6. Erstellen Sie eine Build-Plattform-Konfiguration für die Plattform, unter der Sie die Projektmappe erstellen möchten (x86, x64). So erstellen Sie die Build-Konfiguration:

   a. Klicken Sie mit der rechten Maustaste in Visual Studio auf die Projektmappen und wählen Sie **Configuration Manager**.
   b. Wählen Sie unter **Active solution platform** den Befehl **New...** und wählen Sie anschließend die x86- oder x64-Konfiguration aus (in diesem Beispiel **x86**).



Sie sind nun fertig und können die Projektmappe in Visual Studio erstellen und ausführen. Denken Sie daran, die Projektmappe mit der richtigen Konfiguration für Ihre Zielplattform (x86, x64) zu erstellen.

© 2019-2025 Altova GmbH

10.4 Integration auf Dokumentebene

Im Vergleich zur Integration auf Applikationsebene ist die Integration auf Dokumentebene komplexer, bietet aber dafür mehr Flexibilität beim Einbetten von MapForce-Funktionalitäten in Ihre Applikation mit Hilfe von ActiveX Controls. Auf diese Art haben Sie selektiven Zugriff auf die folgenden Teile der MapForce-Benutzeroberfläche:

- Dokument-Bearbeitungsfenster
- Projektfenster

Wie bereits im Abschnitt Integration auf Applikationsebene 455 erwähnt, ist für eine ActiveX-Integration auf Applikationsebene nur ein Control, nämlich das **MapForceControl**, erforderlich. Für eine ActiveX-Integration auf Dokumentebene werden die MapForce-Funktionalitäten jedoch durch die folgenden ActiveX Controls zur Verfügung gestellt:

- MapForceControl 471
- MapForceControl Document 478
- MapForceControl Placeholder 484

Diese Controls werden durch die Datei aus dem Applikationsinstallationsordner von MapForce zur Verfügung gestellt. Wenn Sie die ActiveX-Integration mit Hilfe von Visual Studio erstellen, benötigen Sie über die Visual Studio Toolbox Zugriff auf diese Controls (siehe Hinzufügen der ActiveX Controls zur Toolbox 454).

Die grundlegenden Schritte, um die ActiveX Controls auf Dokumentebene in Ihre Applikation zu integrieren sind die folgenden:

1. Instantiieren Sie in Ihrer Applikation zuerst **MapForceControl**. Die Instantiierung dieses Control ist obligatorisch; Es ermöglicht die Unterstützung für die oben erwähnten Controls **MapForceControl Document** und **MapForceControl Placeholder**. Die Eigenschaft `IntegrationLevel` 473 muss auf **ICActiveXIntegrationOnDocumentLevel** (oder "1") gesetzt werden. Um das Control für den Benutzer auszublenden, setzen Sie seine Eigenschaft **Visible** auf **False**. Anmerkung: Verwenden Sie für die Integration auf Dokumentebene nicht die **Open**-Methode des **MapForceControl**, da dies zu unerwünschten Ergebnissen führen könnte. Verwenden Sie stattdessen die entsprechenden open-Methoden von **MapForceControl Document** und **MapForceControl PlaceHolder**.

2. Erstellen Sie mindestens eine Instanz von **MapForceControl Document** in Ihrer Applikation. Dieses Control stellt Ihrer Applikation das Dokumentbearbeitungsfenster von MapForce zur Verfügung und kann bei Bedarf mehrmals instantiiert werden.erwenden Sie die Methode **Open,** um eine vorhandene Datei zu laden. Um Funktionen im Zusammenhang mit einem Dokument aufzurufen, verwenden Sie die Methoden **Path** und **Save** oder Methoden und Eigenschaften, die über die Eigenschaft **Document** zur Verfügung stehen. Anmerkung: Das Control unterstützt keinen schreibgeschützten Modus. Der Wert der Eigenschaft **ReadOnly** wird ignoriert.

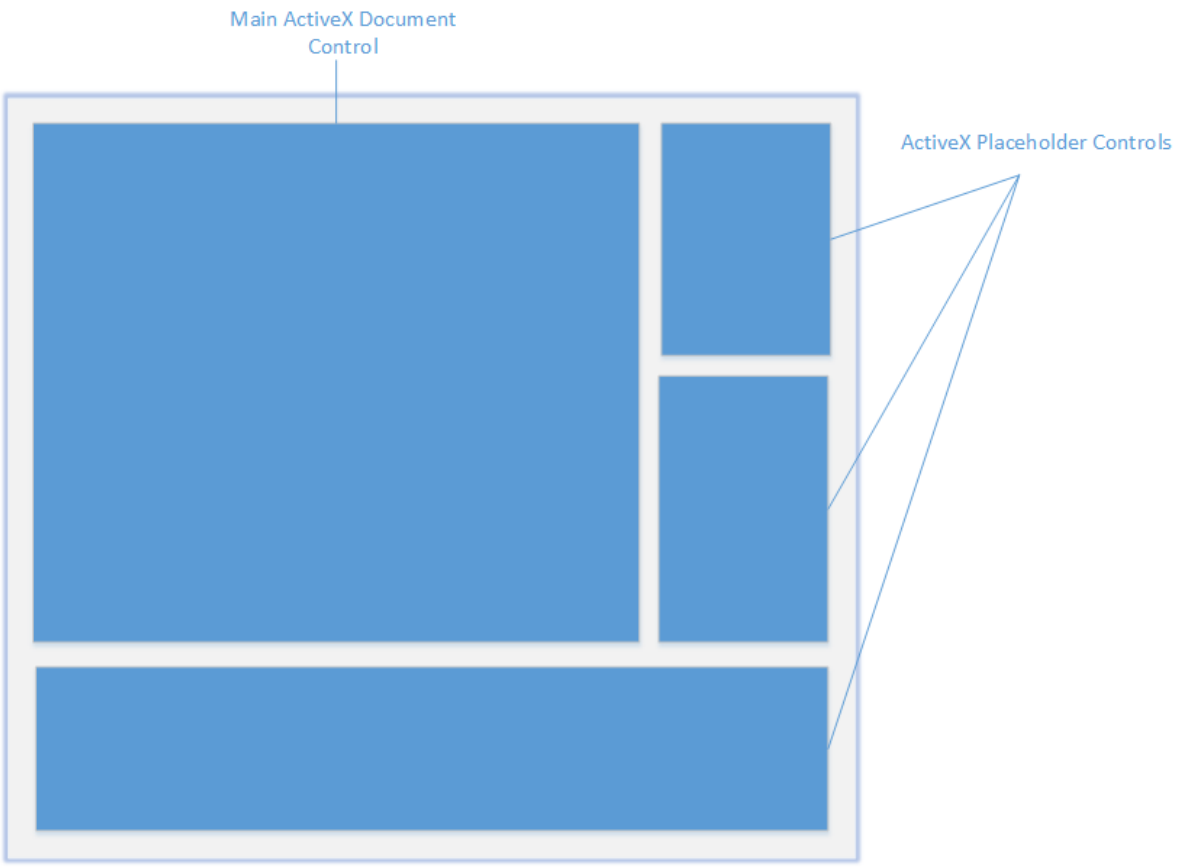
3. Fügen Sie optional für jedes weitere Fenster (bei dem es sich nicht um das Dokumentfenster handelt) und das Ihrer Applikation zur Verfügung stehen soll, den **MapForceControl Placeholder** zur Ihrer Applikation hinzu. Mit Hilfe von Instanzen von **MapForceControl PlaceHolder** können Sie selektiv zusätzliche Fenster von MapForce in Ihre Applikation einbetten. Die Art des Fensters (z.B. Projektfenster) wird durch die Eigenschaft **PlaceholderWindowID** definiert. Um daher die Art des Fensters zu definieren, definieren Sie die Eigenschaft **PlaceholderWindowID**. Gültige Fenster-IDs finden Sie unter . Verwenden Sie für jeden Windows Identifier nur einen **MapForceControl PlaceHolder**.

Für Placeholder Controls, die das MapForce-Projektfenster auswählen, stehen zusätzliche Methoden zur Verfügung. Mit Hilfe von **OpenProject** können Sie ein MapForce-Projekt laden. Mit Hilfe der Eigenschaft "Project" und der Methoden und Eigenschaften aus der MapForce Automation-Schnittstelle können Sie andere Operationen im Zusammenhang mit einem Projekt durchführen.

Ein Beispiel, wie Sie in C# oder VB.NET mit Visual Studio eine einfache, aus einem Formular bestehende Applikation, in der die MapForce ActiveX Controls auf Dokumentebene integriert sind, erstellen, sehen Sie in der Beschreibung unten. Beachten Sie, dass Ihre Applikation bei Bedarf auch komplexer sein kann. In der Anleitung unten sehen Sie jedoch, welche Voraussetzungen mindestens gegeben sein müssen, damit eine ActiveX-Integration auf Dokumentebene erfolgen kann.

1. Erstellen Sie ein neues Visual Studio Windows Forms-Projekt mit einem neuen leeren Formular.
2. Fügen Sie die ActiveX Controls zur Toolbox hinzu, falls dies noch nicht geschehen ist (siehe Hinzufügen der ActiveX Controls zur Toolbox 454).
3. Ziehen Sie das **MapForceControl** 471 aus der Toolbox in Ihr neues Formular.
4. Setzen Sie die Eigenschaft **IntegrationLevel** des **MapForceControl** auf **ICActiveXIntegrationOnDocumentLevel** und die Eigenschaft **Visible** auf **False**. Sie können dies entweder über den Code oder über das Fenster **Properties** tun.
5. Ziehen Sie das **MapForceControl Document** 478 aus der Toolbox in das Formular. Dieses Control stellt das Dokument-Hauptfenster von MapForce für Ihre Applikation zur Verfügung. Eventuell müssen Sie die Größe des Fensters an die Ihres Dokuments anpassen.
6. Fügen Sie optional ein oder mehrere **MapForceControl Placeholder** 484 Controls zum Formular hinzu (eines für jeden zusätzlichen Fenstertyp, der für Ihre Applikation benötigt wird, z.B. das Projektfenster). Solche zusätzlichen Placeholder Controls werden normalerweise entweder unterhalb oder rechts oder links vom Dokument-Haupt-Control platziert z.B.:

 © 2019-2025 Altova GmbH



7. Setzen Sie die Eigenschaft **PlaceholderWindowID** jedes **MapForceControl Placeholder** auf einen gültigen Fenster-Identifier. Eine Liste der gültigen Werte finden Sie unter .
8. Fügen Sie, wie unten gezeigt, Befehle zu Ihrer Applikation hinzu (Sie benötigen zumindest Befehle zum Öffnen, Speichern und Schließen von Dokumenten).

Abfragen von MapForce-Befehlen

Bei einer Integration auf Dokumentebene stehen in Ihrer Applikation kein MapForce-Menü- und keine MapForce-Symbolleiste zur Verfügung. Sie können statt dessen die benötigten Befehle aufrufen, deren Status anzeigen und diese programmatisch ausführen, wie folgt:

- Um alle verfügbaren Befehle abzurufen, verwenden Sie die Eigenschaft `CommandsList` 473 des **MapForceControl**.
- Um die Befehle, geordnet nach ihrer Menüstruktur abzurufen, verwenden Sie die Eigenschaft `MainMenu` 473.
- Um die Befehle, geordnet nach der Symbolleiste, in der sie vorkommen, abzurufen, verwenden Sie die Eigenschaft `Toolbars` 474.
- Um Befehle an MapForce zu senden, verwenden Sie die Methode `Exec` 474.
- Um abzufragen, ob ein Befehl gerade aktiviert oder deaktiviert ist, verwenden Sie die Methode `QueryStatus` 475.

Auf diese Art können Sie MapForce-Befehle flexibel in die Menüs und Symbolleisten Ihrer Applikation integrieren.

Über Ihre Installation von MapForce stehen Ihnen auch Symbole für in MapForce verwendete Befehle zur Verfügung. Im Ordner **<Applicationsordner>\Examples\ActiveX\Images** Ihrer MapForce-Installation finden Sie Symbole im GIF-Format. Die Dateinamen entsprechen den im Abschnitt Befehlsreferenz 467 aufgelisteten Befehlsnamen.

Allgemeines

Um das Verhalten von MapForce zu automatisieren, verwenden Sie die für **MapForceControl** 471, **MapForceControl Document** 478 und **MapForceControl Placeholder** 484 beschriebenen Eigenschaften, Methoden und Ereignisse.

Für einen komplexeren Zugriff auf MapForce-Funktionalitäten sollten Sie die folgenden Eigenschaften verwenden:

- `MapForceControl.Application` 472
- `MapForceControlDocument.Document` 479
- `MapForceControlPlaceHolder.Project` 485

Über diese Eigenschaften erhalten Sie Zugriff auf die MapForce Automation-Schnittstelle (<%APPAPI%)

Anmerkung: Verwenden zum Öffnen eines Dokuments immer `MapForceControlDocument.Open` 481 oder `MapForceControlDocument.New` 481 für das entsprechenden Dokument-Control. Verwenden Sie zum Öffnen eines Projekts immer `MapForceControlPlaceHolder.OpenProject` 486 in einem Placeholder Control, das ein MapForce-Projektfenster einbettet.

Beispiele, wie Sie die erforderlichen Controls in unterschiedlichen Programmierumgebungen instantiieren und aufrufen finden Sie unter Beispiele zur ActiveX-Integration 461.

© 2019-2025 Altova GmbH

10.5 Beispiele zur ActiveX-Integration

Dieser Abschnitt enthält Beispiele für die Integration von MapForce auf Dokumentebene über unterschiedliche Container-Umgebungen und Programmiersprachen. Der Quellcode für alle Beispiele steht im Ordner `<ApplicationFolder>\Examples\ActiveX` Ihrer MapForce Installation zur Verfügung.

10.5.1 C#

Im Ordner `<ApplicationFolder>\Examples\ActiveX\C#` finden Sie ein einfaches Beispiel für eine ActiveX-Integration mittels C# und Visual Studio. Bevor Sie den Quellcode kompilieren und das Beispiel ausführen, stellen Sie sicher, dass alle Voraussetzungen erfüllt werden (siehe Ausführen der C#-Beispiellösung 461).

10.5.1.1 Ausführen der C#-Beispiellösung

In der Visual Studio-Beispielprojektmappe im Ordner **<ApplicationFolder>\Examples\ActiveX\C#** wird gezeigt, wie die MapForce ActiveX Controls verwendet werden. Beachten Sie die folgenden Schritte, bevor Sie versuchen, diese Projektmappen zu erstellen und auszuführen:

Schritt 1: Überprüfen Sie die Voraussetzungen

Um die Beispielprojektmappen öffnen zu können, benötigen Sie Visual Studio 2010 oder höher. Eine Liste der Voraussetzungen finden Sie unter Voraussetzungen 452.

Schritt 2: Kopieren Sie das Beispiel in ein Verzeichnis, auf das Sie Schreibzugriff haben

Um Visual Studio nicht als Administrator ausführen zu müssen, kopieren Sie den Quellcode in ein Verzeichnis, auf das Sie Schreibzugriff haben, anstatt ihn vom Standardverzeichnis aus auszuführen.

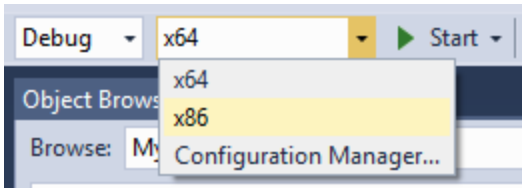
Schritt 3: Überprüfen und definieren Sie alle erforderlichen Control-Eigenschaften

Die Beispielapplikation enthält eine Instanz von MapForceControlDocument 478 und mehrere Instanzen von MapForceControlPlaceHolder 484 Controls. Überprüfen Sie noch einmal, ob die folgenden Eigenschaften dieser Controls, wie in der Tabelle unten angegeben, konfiguriert sind:

So können Sie die Eigenschaften eines ActiveX Control anzeigen oder definieren:

1. Öffnen Sie das Formular **MDIMain.cs** im Designer-Fenster.

Anmerkung: Unter 64-Bit Windows müssen Sie die Build-Konfiguration der Visual Studio-Projektmappen eventuell in "x86" ändern, **bevor** Sie das Designer-Fenster öffnen. Wenn Sie das Beispiel als 64-Bit-Applikation erstellen müssen, schlagen Sie nach unter Voraussetzungen 452.



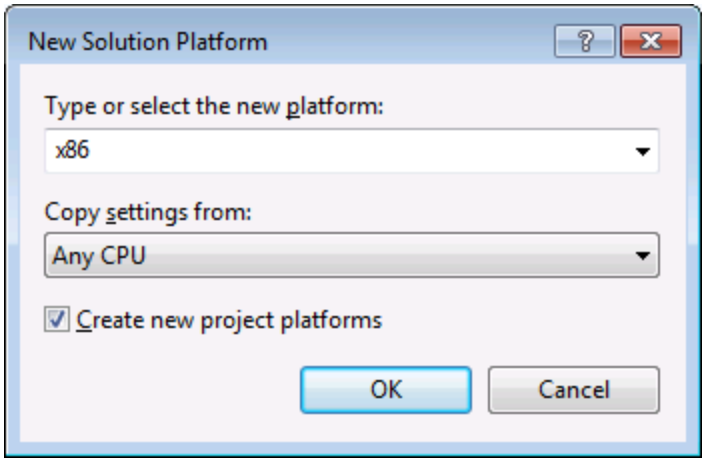
2. Öffnen Sie das **Document Outline**-Fenster von Visual Studio (Klicken Sie im Menü **View** auf **Other Windows | Document Outline**).

3. Klicken Sie im Fenster **Document Outline** auf ein ActiveX Control und bearbeiten Sie die benötigte Eigenschaft im Fenster **Properties,** z.B.:

Schritt 4: Definieren Sie die Build-Plattform

- Erstellen Sie eine Build-Plattform-Konfiguration für die Plattform, unter der Sie die Projektmappe erstellen möchten (x86, x64). So erstellen Sie die Build-Konfiguration:

  a. Klicken Sie mit der rechten Maustaste auf die Projektmappe in Visual Studio und wählen Sie **Configuration Manager**.
  b. Wählen Sie unter **Active solution platform** den Befehl **New...** und wählen Sie anschließend die x86- oder x64-Konfiguration (in diesem Beispiel **x86**).



Sie sind nun fertig und können die Projektmappe in Visual Studio erstellen und ausführen. Denken Sie daran, die Projektmappe mit der richtigen Konfiguration für Ihre Zielplattform (x86, x64) zu erstellen, da es sonst zu Laufzeitfehlern kommen kann.

10.5.2 Java

MapForce ActiveX-Komponenten können von Java-Code aus aufgerufen werden. Die Java-Integration wird von den unten aufgelisteten Bibliotheken zur Verfügung gestellt. Diese Bibliotheken stehen im Ordner `<ApplicationFolder>\Examples\JavaAPI` Ihrer MapForce Installation zur Verfügung, nachdem Sie MapForce und das MapForce Integrationspaket installiert haben (siehe auch Voraussetzungen 452).

- `AltovaAutomation.dll`: ein JNI Wrapper für Altova Automation Server (bei 32-Bit-Installationen von MapForce)

 © 2019-2025 Altova GmbH

- `AltovaAutomation_x64.dll`: ein JNI-Wrapper für Altova Automation Server (bei 64-Bit-Installationen von MapForce)
- `AltovaAutomation.jar`: Java-Klassen zum Aufrufen von Altova Automation Servern

Anmerkung: Um die Java ActiveX Integration verwenden zu können, müssen sich die .dll- und .jar-Dateien im Java Class-Suchpfad befinden.

Java-Beispielprojekt

Im Lieferumfang Ihres Produkts ist ein Java-Beispielprojekt enthalten. Sie können das Java-Projekt nach Belieben testen und verwenden. Nähere Informationen finden Sie im Abschnitt Java-Beispielprojekt 464.

Regeln für das Mappen der ActiveX Control-Namen auf Java

Eine Dokumentation zu den ActiveX Controls finden Sie in der Objektreferenz 468. Beachten Sie, dass sich die Objektbenennungskonventionen in Java etwas von anderen Sprachen unterscheiden. Für das Mapping zwischenden ActiveX Controls und dem Java Wrapper gelten die folgenden Regeln:

- ***Klassen und Klassennamen***
  Für jede Komponente des MapForce ActiveX Interface gibt es eine Java-Klasse mit dem Namen der Komponente.

- ***Methodennamen***
  Die Methodennamen im Java Interface sind dieselben wie die in den COM Interfaces, beginnen aber aufgrund der Java-Namenskonventionen mit einem Kleinbuchstaben. Zum Aufrufen von COM-Eigenschaften können dem Eigenschaftsnamen vorangestellte Java-Methoden mit `get` und `set` verwendet werden. Wenn eine Eigenschaft keinen Schreibzugriff ermöglicht, steht keine Setter-Methode zur Verfügung. Beispiel: Für die Eigenschaft `IntegrationLevel` des MapForceControl stehen die Java-Methoden `getIntegrationLevel` und `setIntegrationLevel` zur Verfügung.

- ***Enumerationen***
  Für jede im ActiveX Interface definierte Enumeration ist eine Java-Enumeration desselben Namens und mit denselben Werten definiert.

- ***Events und Event Handler***
  Für jedes Interface im Automation Interface, das Events unterstützt, steht ein Java-Interface desselben Namens plus '`Event`' zur Verfügung. Um das Überladen von Einzel-Events zu vereinfachen, gibt es eine Java-Klasse mit Standardimplementierungen für alle Events. Der Name dieser Java-Klasse ist der Name des Event Interface plus '`DefaultHandler`'. Beispiel:
  `MapForceControl`: Java-Klasse zum Aufrufen der Applikation
  `MapForceControlEvents`: Events Interface für das MapForceControl Control
  `MapForceControlEventsDefaultHandler`: Standard-Handler für `MapForceControlEvents`

Ausnahmen für Mapping-Regeln

Zu den oben aufgelisteten Regeln gibt es die folgenden Ausnahmen:

Dieser Abschnitt

In diesem Abschnitt wird erklärt, wie einige grundlegende MapForce ActiveX-Funktionen über Java-Code aufgerufen werden können. Der Abschnitt ist in die folgenden Unterabschnitte gegliedert:

- Java-Beispielprojekt 464
- Erstellen der ActiveX Controls 465
- Laden der Daten in die Controls 465
- Behandlung von Events 466
- Menüs 466
- Behandlung von UI Update Events 466

10.5.2.1 Java-Beispielprojekt

Das MapForce-Installationspaket enthält ein Java-Beispielprojekt, das Sie Ordner "ActiveX Examples" des Applikationsordners `<ApplicationFolder>\Examples\ActiveX\Java\` finden.

Im Java-Beispiel wird gezeigt, wie Sie das MapForceControl in eine mit Java erstellte Desktop Applikation integrieren können. Sie können das Beispielprojekt mit Hilfe der Batch-Datei `BuildAndRun.bat`, direkt über die Befehlszeile testen oder Sie können es in Eclipse kompilieren und ausführen. Anleitungen dafür finden Sie weiter unten.

Dateiliste

Der Ordner für die Java-Beispiele enthält alle zum Ausführen des Beispielprojekts erforderlichen Dateien. Diese Dateien sind unten aufgelistet:

Funktionen in diesem Beispiel

Sie können das Beispiel nach Ihren Wünschen modifizieren.

In den Codefragmenten werden die folgenden spezifischen Schritte beschrieben:

- Erstellen der ActiveX Controls 465: Startet MapForce, das als Automation Server registriert ist, bzw. aktiviert das Programm, wenn MapForce bereits ausgeführt wird.
- Laden der Daten in die Controls 465: Navigiert zu einem der mit MapForce installierten Beispieldokumente und öffnet es.
- Grundlegendes zur Event-Behandlung 466: Wechselt von der Ansicht aller offenen Dokumente in die Textansicht. Im Code wird auch gezeigt, wie man durch offene Dokumente iteriert.
- Menüs 466: Validiert das aktive Dokument und zeigt das Ergebnis in einem Meldungsfeld an. Im Code wird gezeigt, wie Ausgabeparameter verwendet werden.
- Behandlung von UI Update Events 466: Zeigt, wie MapForce Events behandelt werden.
-

Aktualisieren des Pfads zum Ordner "Examples"

Bevor Sie das zur Verfügung gestellte Beispiel ausführen, müssen Sie die **MapForceContainer.java**-Datei möglicherweise bearbeiten und überprüfen, ob der folgende Pfad auf den tatsächlichen Ordner verweist, in dem die MapForce Beispieldateien auf Ihrem Betriebssystem gespeichert sind.

```
// Locate samples installed with the product.
final String strExamplesFolder = System.getenv( "USERPROFILE" ) + "\\Documents\\Altova\
\MapForce2025\\MapForceExamples\\";
```

 © 2019-2025 Altova GmbH

Ausführen des Beispiels über die Befehlszeile

So führen Sie das Beispiel über die Befehlszeile aus:

1. Überprüfen Sie, ob alle Voraussetzungen erfüllt werden (siehe Voraussetzungen 452).
2. Öffnen Sie ein Eingabeaufforderungsfester, wechseln Sie in den Java-Beispielprojektordner und geben Sie folgende Zeile ein:

```
buildAndRun.bat "<Pfad-zum-Java-bin-Ordner>"
```

1. Drücken Sie die **Eingabetaste**.

Der Java-Quellcode in `MapForceContainer.java` wird kompiliert und anschließend ausgeführt.

Kompilieren und Ausführen des Beispiels in Eclipse

So importieren Sie das Java-Beispielprojekt in Eclipse:

3. Überprüfen Sie, ob alle Voraussetzungen erfüllt werden (siehe Voraussetzungen 452).
1. Klicken Sie im Menü **File** auf **Import**.
2. Wählen Sie **Existing Projects into Workspace** und navigieren Sie zur Eclipse-Projektdatei unter `<ApplicationFolder>\Examples\ActiveX\Java\`. Da Sie möglicherweise keinen Schreibzugriff auf diesen Ordner haben, wird empfohlen, im Importdialogfeld das Kontrollkästchen **Copy projects into workspace** zu aktivieren.

Um die Beispielapplikation zu starten, klicken Sie mit der rechten Maustaste im Package Explorer auf das Paket und wählen Sie den Befehl **Run as | Java Application**.

Hilfe zu Java API-Klassen steht in Form von Kommentaren im Code sowie als Javadoc-Ansicht von Eclipse zur Verfügung. Um die Javadoc-Ansicht in Eclipse zu aktivieren, wählen Sie den Menübefehl **Window | Show View | JavaDoc**.

10.5.2.2 Erstellen der ActiveX Controls

Im unten gezeigten Code sehen Sie wie ActiveX Controls erstellt werden. Die Konstruktoren erstellen die Java Wrapper-Objekte. Durch Hinzufügen dieser aus Canvas stammenden Objekte zu einem Bereich oder Rahmen wird die Erstellung des in Wrapper verpackten ActiveX-Objekts ausgelöst.
).

10.5.2.3 Laden der Daten in die Controls

Im unten gezeigten Code sehen Sie wie Daten in ActiveX Controls geladen werden können.

10.5.2.4 Grundlegendes zur Event-Behandlung

Im unten gezeigten Code sehen Sie wie grundlegende Events behandelt werden. Bei Aufruf der MapForceControl-Methode `open` oder beim Öffnen einer Datei über das Menü oder die Projektstruktur wird das `onOpenedOrFocused` Event an den dazugehörigen Event Handler gesendet. Im Prinzip wird dieses Event behandelt, indem die Datei durch Aufruf der `open`-Methode des MapForceDocumentControl geöffnet wird.

10.5.2.5 Menüs

Im unten gezeigten Code sehen Sie, wie Menüeinträge erstellt werden können. Jedes `MapForceCommand` Objekt holt ein entsprechendes `MenuItem` Objekt, wobei der `ActionCommand` auf die ID des Befehls gesetzt wird. Die von allen Menüeinträgen generierten Aktionen werden von derselben Funktion gehandelt. Diese kann bestimmte Handlings (wie z.B Neuinterpretieren der Schließfunktion) durchführen oder die Ausführung durch Aufruf seiner `exec`-Methode an das `MapForceControl` Objekt delegieren. Das `menuMap` Objekt, das bei der Menüerstellung befüllt wird, wird später verwendet (siehe Abschnitt Behandlung von UI Update Events 466).

10.5.2.6 Behandlung von UI Update Events

Im unten gezeigten Code, sehen Sie wie ein UI-Update Event Handler erstellt werden kann.

 © 2019-2025 Altova GmbH

10.6 Befehlsreferenz

In diesem Abschnitt werden die Namen und Identifier aller Menübefehle aufgelistet, die in MapForce zur Verfügung stehen. In den einzelnen Unterabschnitten werden jeweils die Befehle aus dem entsprechenden Menü von MapForce aufgelistet. Die Befehlstabellen sind folgendermaßen gegliedert:

- In der Spalte "Menübefehl" sehen Sie den Menütext des Befehls, wie er in MapForce, angezeigt wird, damit Sie die dem Befehl zugrunde liegende Funktionalität leichter erkennen.
- In der Spalte "Befehlsname" ist der String angegeben, anhand dessen ein Symbol desselben Namens aus dem Ordner **ActiveX\Images** des MapForce Installationsverzeichnis aufgerufen werden kann.
- In den "ID"-Spalten sehen Sie die numerischen Identifier der Spalte, die als Argument an die Methoden geliefert wird, die diesen Befehl ausführen oder abfragen.

Um einen Befehl auszuführen, verwenden Sie die Methoden `MapForceControl.Exec` 474 oder `MapForceControlDocument.Exec` 480. Um den Status eines Befehls abzufragen, verwenden Sie die Methoden `MapForceControl.QueryStatus` 475 oder `MapForceControlDocument.QueryStatus` 481.

Einige dieser Befehle werden je nach installierter MapForce Edition eventuell nicht unterstützt.

10.7 Objektreferenz

Objekte:
MapForceCommand 468
MapForceCommands 470
MapForceControl 471
MapForceControlDocument 478
MapForceControlPlaceHolder 484

Um Zugriff auf die MapForce Standardfunktionalitäten zu erhalten, können auch Objekte des **MapForce Automation Interface** aufgerufen werden. Nähere Informationen dazu finden Sie unter MapForceControl.Application 472, MapForceControlDocument.Document 479 und MapForceControlPlaceHolder.Project 485.

10.7.1 MapForceCommand

Eigenschaften:
`ID` 469
`Label` 469
`Name` 470
`IsSeparator` 469
`ToolTip` 470
`StatusText` 470
`Accelerator` 469
`SubCommands` 470

Beschreibung:
Jedes Command Objekt kann einer von drei möglichen Typen sein: ein ausführbarer Befehl, ein Befehlscontainer (z.B. ein Menü, ein Untermenü oder eine Symbolleiste) oder ein Menütrennzeichen. Um herauszufinden, welche Art von Informationen im aktuellen `Command`-Objekt gespeichert sind, fragen Sie seine Eigenschaften `ID`, `IsSeparator` und `SubCommands` folgendermaßen ab.

Das Befehlsobjekt ist...	wenn...
ein ausführbarer Befehl	• `ID` größer als Null ist • `IsSeparator` "false" ist • `SubCommands` leer ist
ein Befehlscontainer	• `ID` Null ist • `IsSeparator` "false" ist • `SubCommands` eine Sammlung von `Command` Objekten enthält
ein Trennzeichen	• `ID` Null ist • `IsSeparator` "true" ist

© 2019-2025 Altova GmbH

10.7.1.1 Accelerator

Eigenschaft: `Accelerator` als string

Beschreibung:
Gibt die für den Befehl definierte Zugriffstaste zurück. Wenn dem Befehl keine Zugriffstaste zugewiesen wurde, gibt diese Eigenschaft den leeren String zurück. Die String-Darstellung der Zugriffstaste hat das folgenden Format:

```
[ALT+][CTRL+][SHIFT+]key
```

`key` wird mittels der Windows Plattform SDK-Funktion `GetKeyNameText` konvertiert.

10.7.1.2 ID

Eigenschaft: `ID` als long

Beschreibung:
Mit dieser Eigenschaft wird der eindeutige Identifier des Befehls abgerufen. Die ID eines Befehls wird benötigt, um den Befehl (mittels `Exec` 474) auszführen oder seinen Status (mittels `QueryStatus` 475) abzurufen. Wenn der Befehl ein Container für andere Befehle (z.B. für ein Menü der obersten Ebene) oder ein Trennzeichen ist, ist die ID 0.

10.7.1.3 IsSeparator

Eigenschaft: `IsSeparator` als boolean

Beschreibung:
Die Eigenschaft gibt `true` zurück, wenn das Befehlsobjekt ein Menütrennzeichen ist; andernfalls wird `false` zurückgegebven. Siehe auch Command 468.

10.7.1.4 Label

Eigenschaft: `Label` als string

Beschreibung:
Diese Eigenschaft ruft den Text des Befehls, wie er auf der grafischen Benutzeroberfläche von MapForce angezeigt wird, ab. Wenn der Befehl ein Trennzeichen ist, ist "Label" ein leerer String. Diese Eigenschaft kann für einige Symbolleistenbefehle, zu denen es keinen GUI-Text gibt, auch einen leeren String zurückgeben.

10.7.1.5 Name

Eigenschaft: `Name` als string

Beschreibung:
Diese Eigenschaft ruft den eindeutigen Namen des Befehls ab. Anhand dieses Werts kann die Symboldatei das Befehls, falls verfügbar, abgerufen werden. Die verfügbaren Symboldateien befinden sich im Ordner **<ApplicationFolder>\Examples\ActiveX\Images** Ihrer MapForce-Installation.

10.7.1.6 StatusText

Eigenschaft: `Label` als string

Beschreibung:
Der Statustext ist der Text, der in der Statusleiste von MapForce angezeigt wird, wenn der Befehl ausgewählt wird. Dies gilt nur für Befehlsobjekte, die keine Trennzeichen oder Container von anderen Befehlen sind. Andernfalls ist die Eigenschaft ein leerer String.

10.7.1.7 SubCommands

Eigenschaft: `SubCommands` als `Commands` 470

Beschreibung:
Die Eigenschaft `SubCommands` ruft die Sammlung von `Command` 468-Objekten, die dem aktuellen Befehl untergeordnet sind, auf. Die Eigenschaft gilt nur für Befehle, die Container für andere Befehle sind (Menüs, Untermenüs oder Symbolleisten). Bei solchen Container-Befehlen ist die `ID` auf `0` gesetzt und die Eigenschaft `IsSeparator` auf `false`.

10.7.1.8 ToolTip

Eigenschaft: `ToolTip` als string

Beschreibung:
Diese Eigenschaft ruft den Text, der als Tooltipp zu den einzelnen Befehlen angezeigt wird, ab. Wenn der Befehl keinen Tooltipp-Text hat, so gibt die Eigenschaft einen leeren String zurück.

10.7.2 MapForceCommands

Eigenschaften:
`Count` 471
`Item` 471

Beschreibung:

© 2019-2025 Altova GmbH

Sammlung von Command[468]-Objekten für den Zugriff auf Befehlsbezeichnungen und IDs des MapForceControl. Diese Befehle können mit der Exec[474] Methode ausgeführt werden und ihr Status kann mittels QueryStatus[475] abgefragt werden.

10.7.2.1 Count

Eigenschaft: `Count` als long

Beschreibung:
Anzahl der Command[468] Objekte auf dieser Ebene der Collection

10.7.2.2 Item

Eigenschaft: `Item` (n als long) als Command[468]

Beschreibung:
Ruft den Befehl mit den Index n in dieser Sammlung auf. Der Index basiert auf 1.

10.7.3 MapForceControl

Eigenschaften:
IntegrationLevel[473]
Appearance[472]
Application[472]
BorderStyle[472]
CommandsList[473]
EnableUserPrompts[473]
MainMenu[473]
Toolbars[474]

Methoden:
Open[475]
Exec[474]
QueryStatus[475]

Events:
OnUpdateCmdUI[477]
OnOpenedOrFocused[477]
OnCloseEditingWindow[476]
OnFileChangedAlert[476]
OnDocumentOpened[476]
OnValidationWindowUpdated[478]

Dieses Objekt ist ein vollständiges ActiveX Control und sollte nur sichtbar sein, wenn die MapForce Bibliothek im Applikationsebenenmodus verwendet wird.

10.7.3.1 Eigenschaften

Es sind die folgenden Eigenschaften definiert:

IntegrationLevel 473
EnableUserPrompts 473
Appearance 472
BorderStyle 472

Befehlsbezogene Eigenschaften:
CommandsList 473
MainMenu 473
Toolbars 474

Zugriff auf die MapForceAPI:
Application 472

10.7.3.1.1 Appearance

Eigenschaft: `Appearance` als short

Dispatch Id: *-520*

Beschreibung:
Bei einem Wert, der nicht gleich `0` ist, wird ein Client-Rand rund um das Control angezeigt. Der Standardwert ist `0`.

10.7.3.1.2 Application

Eigenschaft: `Application` als Application

Dispatch Id: *1*

Beschreibung:
Mit der Eigenschaft `Application` erhalten Sie Zugriff auf das `Application` Objekt der vollständigen MapForce Automation Server API. Die Eigenschaft ist schreibgeschützt.

10.7.3.1.3 BorderStyle

Eigenschaft: `BorderStyle` als short

Dispatch Id: *-504*

Beschreibung:
Bei einem Wert `1` wird das Control mit einer dünnen Umrandung angezeigt. Der Standardwert ist `0`.

 © 2019-2025 Altova GmbH

10.7.3.1.4 CommandsList

Eigenschaft: `CommandList` als `Commands` 470 (schreibgeschützt)

Dispatch Id: *1004*

Beschreibung:
Diese Eigenschaft gibt eine flache Liste aller Befehle zurück, die mit MapForceControl verfügbar sind. Um Befehle, geordnet nach Menüstruktur, abzurufen, verwenden Sie `MainMenu` 473. Um Symbolleistenbefehle abzurufen, verwenden Sie `Toolbars` 474.

C#-Beispiel

10.7.3.1.5 EnableUserPrompts

Eigenschaft: `EnableUserPrompts` als boolean

Dispatch Id: *1006*

Beschreibung:
Wenn Sie diese Eigenschaft auf *false* setzen, wird die Eingabeaufforderung im Control deaktiviert. Der Standardwert ist *true*.

10.7.3.1.6 IntegrationLevel

Eigenschaft: `IntegrationLevel` als `ICActiveXIntegrationLevel` 487

Dispatch Id: *1000*

Beschreibung:
Die Eigenschaft `IntegrationLevel` bestimmt den Operationsmodus des Control. Nähere Informationen dazu siehe auch Integration auf Applikationsebene 455 und Integration auf Dokumentebene 457.

Anmerkung: Diese Eigenschaft muss unbedingt sofort nach Erstellung des `MapForceControl` Objekts definiert werden.

10.7.3.1.7 MainMenu

Eigenschaft: `MainMenu` als `Command` 468 (schreibgeschützt)

Dispatch Id: *1003*

Beschreibung:
Diese Eigenschaft enthält Informationen über die Struktur und die Befehle im MapForceControl-Hauptmenü als `Command`-Objekt. Das `Command`-Objekt enthält alle verfügbaren Untermenüs von MapForce (z.B. Datei, Bearbeiten, Ansicht, usw.). Verwenden Sie die Eigenschaft `SubCommands` der Eigenschaft `MainMenu`, um die Untermenüobjekte abzurufen. Jedes Untermenü ist ebenfalls ein `Command`-Objekt. Sie können bei jedem Untermenü weiter durch dessen `SubCommands`-Eigenschaft iterieren, um die jeweiligen Child-Befehle und Trennzeichen dieser Untermenüs abzurufen (Auf diese Art können Sie z.B. das Applikationsmenü programmatisch erstellen). Beachten Sie, dass einige Menübefehle als Container ("Parents") für andere Menübefehle dienen. In diesen Fällen haben diese ebenfalls eine Eigenschaft `SubCommands`. Um die Struktur aller Menübefehle programmatisch abzurufen, müssen Sie wahrscheinlich eine rekursive Funktion erstellen.

C# example

10.7.3.1.8 Toolbars

Eigenschaft: `Toolbars` als Commands [470] (schreibgeschützt)

Dispatch Id: *1005*

Beschreibung:
Diese Eigenschaft enthält Informationen über die Struktur von MapForceControl-Symbolleisten, als `Command`-Objekt. Das `Command`-Objekt enthält alle verfügbaren Symbolleisten von MapForce. Verwenden Sie die Eigenschaft `SubCommands` der Eigenschaft `Toolbars`, um die Symbolleisten abzurufen. Jede Symbolleiste ist ebenfalls ein `Command`-Objekt. Sie können bei jeder Symbolleiste weiter durch deren `SubCommands`-Eigenschaft iterieren, um deren Befehle abzurufen (Auf diese Art können Sie z.B. die Symbolleisten der Applikation programmatisch erstellen).

C# example

10.7.3.2 Methoden

Es sind die folgenden Methoden definiert:

`Open` [475]
`Exec` [474]
`QueryStatus` [475]

10.7.3.2.1 Exec

Methode: `Exec` (`nCmdID` als long) als boolean

Dispatch Id: *6*

Beschreibung:
`Exec` ruft den MapForce Befehl mit der ID `nCmdID` auf. Wenn der Befehl ausgeführt werden kann, gibt die Methode `true` zurück. Eine Liste aller verfügbaren Befehle finden Sie unter `CommandsList` [473]. Um den Status eines Befehls abzurufen, verwenden Sie `QueryStatus` [475].

 © 2019-2025 Altova GmbH

10.7.3.2.2 Open

Methode: `Open` (`strFilePath` als string) als boolean

Dispatch Id: *5*

Beschreibung:
Das Ergebnis der Methode ist von der Erweiterung abhängig, die im Argument `strFilePath` übergeben wird. Ist die Dateierweiterung `.sps`, wird ein neues Dokument geöffnet. Ist die Dateierweiterung `.svp`, wird das entsprechende Projekt geöffnet. Wird eine andere Dateierweiterung in die Methode übergeben, versucht das Control, die Datei als neue Komponente in das aktive Dokument zu laden.

Verwenden Sie diese Methode nicht, um Dokumente oder Projekte bei Verwendung des Control auf Dokumentebene zu laden. Verwenden Sie statt dessen `MapForceControlDocument.Open` 481 und `MapForceControlPlaceHolder.OpenProject` 486.

10.7.3.2.3 QueryStatus

Methode: `QueryStatus` (`nCmdID` als long) als long

Dispatch Id: *7*

Beschreibung:
`QueryStatus` gibt den Status "enabled/disabled" und "checked/unchecked" des von `nCmdID`. definierten Befehls zurück. Der Status wird als Bitmaske zurückgegeben.

Bit	Wert	Name	Bedeutung
0	1	Supported	Setzen, wenn der Befehl unterstützt wird.
1	2	Enabled	Setzen, wenn der Befehl aktiviert ist (ausgeführt werden kann).
2	4	Checked	Setzen, wenn der Befehl angehakt ist.

Das bedeutet, dass die Befehls-ID bei Rückgabe von 0 durch `QueryStatus` nicht als gültiger MapForce Befehl erkannt wird. Wenn `QueryStatus` einen Wert `1` oder `5` zurückgibt, wird der Befehl deaktiviert.

10.7.3.3 Events

Das MapForceControl ActiveX Control stellt die folgenden Verbindungspunkt-Events bereit:

`OnUpdateCmdUI` 477
`OnOpenedOrFocused` 477
`OnCloseEditingWindow` 476
`OnFileChangedAlert` 476
`OnDocumentOpened` 476
`OnValidationWindowUpdated` 478

10.7.3.3.1 OnCloseEditingWindow

Event: `OnCloseEditingWindow` (i_strFilePath als String) als boolean

Dispatch Id: *1002*

Beschreibung:
Dieses Event wird ausgelöst, wenn MapForce ein bereits geöffnetes Dokument schließen muss. Als Antwort auf dieses Event müssen Clients das mit *i_strFilePath* verknüpfte Bearbeitungsfenster schließen. Bei Rückgabe von *true* von diesem Event, wird angezeigt, dass der Client das Dokument geschlossen hat. Clients können *false* zurückgeben, wenn keine bestimmte Behandlung erforderlich ist und MapForceControl versuchen soll, das Bearbeitungsfenster zu schließen und das damit verknüpfte Dokument-Control zu zerstören.

10.7.3.3.2 OnDocumentOpened

Event: `OnDocumentOpened` (`objDocument` als Document)

Dispatch Id: *1*

Beschreibung:
Dieses Event wird immer, wenn ein Dokument geöffnet wird, ausgelöst. Das Argument `objDocument` ist ein `Document` Objekt aus dem MapForce Automation Interface und kann dazu verwendet werden, weitere Details zum Dokument abzurufen oder weitere Operationen durchzuführen. Bei Integration auf Dokumentebene ist es oft besser, stattdessen das Event `MapForceControlDocument.OnDocumentOpened` 483 zu verwenden.

10.7.3.3.3 OnFileChangedAlert

Event: `OnFileChangedAlert` (`i_strFilePath` als String) als bool

Dispatch Id: *1001*

Beschreibung:
Dieses Event wird ausgelöst, wenn eine mit MapForceControl geladene Datei auf der Festplatte von einer anderen Applikation geändert wurde. Clients sollten true zurückgeben, wenn Sie das Event behandelt haben oder false, wenn MapForce es auf die übliche Art behandeln soll, also fragen soll, ob das Dokument neu geladen werden soll.

10.7.3.3.4 OnLicenseProblem

Event: `OnLicenseProblem` (`i_strLicenseProblemText` als String)

Dispatch Id: *1005*

Beschreibung:
Dieses Event wird ausgelöst, wenn MapForceControl feststellt, dass für dieses Control keine gültige Lizenz vorhanden ist. Wenn die Lizenz auf eine bestimmte Anzahl an Benutzern beschränkt ist, kann dies auch einige Zeit nach Initialisierung des Control geschehen. Dieses Event sollte dazu verwendet werden, um den Zugriff auf

 © 2019-2025 Altova GmbH

die Funktionalität dieses Control zu deaktivieren. Nach diesem Event blockiert dieses Control den Zugriff auf seine Funktionalitäten (und zeigt z.B. leere Fenster in seinen Controls an und gibt bei Anfragen Fehlermeldungen zurück).

10.7.3.3.5 OnOpenedOrFocused

Event: `OnOpenedOrFocused` (`i_strFilePath` als String, `i_bOpenWithThisControl` als bool)

Dispatch Id: *1000*

Beschreibung:
Bei Integration auf Applikationsebene informiert dieses Event Clients, dass ein Dokument von MapForce geöffnet oder aktiv gemacht wurde.

Bei Integration auf Dokumentebene gibt dieses Event dem Client die Anweisung, die Datei `i_strFilePath` in einem Dokumentfenster zu öffnen. Wenn die Datei bereits offen ist, sollte das entsprechende Dokumentfenster zum aktiven Fenster gemacht werden.

Wenn `i_bOpenWithThisControl` `true` ist, muss das Dokument mit MapForceControl geöffnet werden, da ein interner Zugriff erforderlich ist. Andernfalls kann die Datei mit anderen Editoren geöffnet werden.

10.7.3.3.6 OnToolWindowUpdated

Event: `OnToolWindowUpdated(pToolWnd` als long `)`

Dispatch Id: *1006*

Beschreibung:
Dieses Event wird ausgelöst, wenn das Fenster "Extras" aktualisiert wird.

10.7.3.3.7 OnUpdateCmdUI

Event: `OnUpdateCmdUI ()`

Dispatch Id: *1003*

Beschreibung:
Wird häufig aufgerufen, um dem Programmierer die Möglichkeit zu geben, den Status von MapForce Befehls mittels `MapForceControl.QueryStatus` 475 zu überprüfen. Führen Sie an diesem Callback keine langen Operationen durch.

10.7.3.3.8 OnValidationWindowUpdated

Event: `OnValidationWindowUpdated` ()

Dispatch Id: *3*

Beschreibung:
Dieses Event wird ausgelöst, wenn das Fenster "Validierungsausgabe" mit neuen Informationen aktualisiert wird.

10.7.4 MapForceControlDocument

Eigenschaften:
Appearance 479
BorderStyle 479
Document 479
IsModified 479
Path 480
ReadOnly 480

Methoden:
Exec 480
New 481
Open 481
QueryStatus 481
Reload 482
Save 482
SaveAs 482

Events:
OnDocumentOpened 483
OnDocumentClosed 483
OnModifiedFlagChanged 484
OnFileChangedAlert 483
OnActivate 483

Wenn das MapForceControl im Modus auf Dokumentebene integriert ist, wird jedes Dokument in einem eigenen Objekt vom Typ `MapForceControlDocument` angezeigt. Das `MapForceControlDocument` enthält immer nur ein Dokument, kann aber wiederverwendet werden, um hintereinander mehrere Dateien anzuzeigen.

Dieses Objekt ist ein komplettes ActiveX Control.

10.7.4.1 Eigenschaften

Es sind die folgenden Eigenschaften definiert:

ReadOnly 480

IsModified 479
Path 480
Appearance 479
BorderStyle 479

Zugriff auf die MapForceAPI:
Document 479

10.7.4.1.1 Appearance

Eigenschaft: `Appearance` als short

Dispatch Id: *-520*

Beschreibung:
Bei einem Wert, der nicht gleich `0` ist, wird ein Client-Rand rund um das Control angezeigt. Der Standardwert ist `0`.

10.7.4.1.2 BorderStyle

Eigenschaft: `BorderStyle` als short

Dispatch Id: *-504*

Beschreibung:
Bei einem Wert `1` wird das Control mit einer dünnen Umrandung angezeigt. Der Standardwert ist `0`.

10.7.4.1.3 Document

Eigenschaft: `Document` als Document

Dispatch Id: *3*

Beschreibung:
Mit der Eigenschaft `Document` erhalten Sie Zugriff auf das `Document` Objekt der MapForce Automation Server API. Diese Schnittstelle bietet zusätzliche Funktionalitäten, die mit dem im Control geladenen Dokument verwendet werden können. Die Eigenschaft ist schreibgeschützt.

10.7.4.1.4 IsModified

Eigenschaft: `IsModified` als boolean (schreibgeschützt)

Dispatch Id: *1006*

Beschreibung:
`IsModified` ist *true,* wenn der Dokumentinhalt seit dem letzten Öffnen, Neuladen oder Speichern geändert wurde. Andernfalls ist es *false*.

10.7.4.1.5 Path

Eigenschaft: `Path` als String

Dispatch Id: *1005*

Beschreibung:
Definiert den vollständigen Pfadnamen des im Control geladenen Dokuments bzw. ruft diesen ab.

10.7.4.1.6 ReadOnly

Eigenschaft: `ReadOnly` als boolean

Dispatch Id: *1007*

Beschreibung:
Mit Hilfe dieser Eigenschaft können Sie den schreibgeschützten Status des Dokuments aktivieren und deaktivieren. Wenn `ReadOnly` `true` ist, können keine Änderungen vorgenommen werden.

10.7.4.2 Methoden

Es sind die folgenden Methoden definiert:

Behandlung von Dokumenten:
`New` 481
`Open` 481
`Reload` 482
`Save` 482
`SaveAs` 482

Behandlung von Befehlen:
`Exec` 480
`QueryStatus` 481

10.7.4.2.1 Exec

Methode: `Exec` (`nCmdID` as long) als boolean

Dispatch Id: *8*

 © 2019-2025 Altova GmbH

Beschreibung:
`Exec` ruft den MapForce Befehl mit der ID `nCmdID` auf. Wenn der Befehl ausgeführt werden kann, gibt die Methode `true` zurück. Diese Methode sollte nur dann aufgerufen werden, wenn in der Applikation gerade ein aktives Dokument verfügbar ist.

Um Befehle, geordnet nach Menüstruktur, abzurufen, verwenden Sie die Eigenschaft `MainMenu` 473 von MapForceControl. Um Symbolleistenbefehle abzurufen, verwenden Sie die Eigenschaft `Toolbars` 474 von MapForceControl.

10.7.4.2.2 New

Methode: `New` () als boolean

Dispatch Id: *1000*

Beschreibung:
Diese Methode initialisiert ein neues Mapping innerhalb des Control.

10.7.4.2.3 Open

Methode: `Open` (`strFilePath` als string) als boolean

Dispatch Id: *1001*

Beschreibung:
`Open` lädt die Datei `strFileName` als das neue Dokument in das Control.

10.7.4.2.4 QueryStatus

Methode: `QueryStatus` (`nCmdID` als long) als long

Dispatch Id: *9*

Beschreibung:
`QueryStatus` gibt den Status "enabled/disabled" und "checked/unchecked" des von `nCmdID`. definierten Befehls zurück. Der Status wird als Bitmaske zurückgegeben.

Bit	Wert	Name	Bedeutung
0	1	Supported	Setzen, wenn der Befehl unterstützt wird.
1	2	Enabled	Setzen, wenn der Befehl aktiviert ist (ausgeführt werden kann).
2	4	Checked	Setzen, wenn der Befehl angehakt ist.

Das bedeutet, dass die Befehls-ID bei Rückgabe von 0 durch `QueryStatus` nicht als gültiger MapForce Befehl erkannt wird. Wenn `QueryStatus` einen Wert `1` oder `5` zurückgibt, ist der Befehl deaktiviert. Der

Client sollte die `QueryStatus` Methode des Dokument-Control aufrufen, wenn in der Applikation gerade ein aktives Dokument vorhanden ist.

10.7.4.2.5 Reload

Methode: `Reload` () als boolean

Dispatch Id: *1002*

Beschreibung:
`Reload` aktualisiert den Dokumentinhalt aus dem Dateisystem.

10.7.4.2.6 Save

Methode: `Save` () als boolean

Dispatch Id: *1003*

Beschreibung:
`Save` speichert das aktuelle Dokument unter dem Pfad `Path` 480.

10.7.4.2.7 SaveAs

Methode: `SaveAs` (`strFileName` als string) als boolean

Dispatch Id: *1004*

Beschreibung:
`SaveAs` setzt `Path` 480 auf *strFileName* und speichert das Dokument anschließend unter diesem Pfad.

10.7.4.3 Events

Das MapForceControlDocument ActiveX Control stellt die folgenden Verbindungspunkt-Events zur Verfügung:

`OnDocumentOpened` 483
`OnDocumentClosed` 483
`OnModifiedFlagChanged` 484
`OnFileChangedAlert` 483
`OnActivate` 483
`OnSetEditorTitle` 484

 © 2019-2025 Altova GmbH

10.7.4.3.1 OnActivate

Event: `OnActivate` ()

Dispatch Id: *1005*

Beschreibung:
Dieses Event wird ausgelöst, wenn das Dokument-Control aktiviert ist, den Fokus hat und bereit für die Benutzereingabe ist.

10.7.4.3.2 OnDocumentClosed

Event: `OnClosed` ()

Dispatch Id: *1001*

Beschreibung:
Dieses Event wird immer dann ausgelöst, wenn das in dieses Control geladene Event geschlossen wird. Das Argument `objDocument` ist ein `Document` Objekt aus dem MapForce Automation Interface und sollte mit Vorsicht verwendet werden.

10.7.4.3.3 OnDocumentOpened

Event: `OnDocumentOpened` (`objDocument` als Document)

Dispatch Id: *1000*

Beschreibung:
Dieses Event wird immer dann ausgelöst, wenn ein Dokument in diesem Control geöffnet wird. Das Argument `objDocument` ist ein `Document` Objekt aus dem MapForce Automation Interface und dient dazu, nähere Informationen über das Dokument abzurufen oder weitere Operationen auszuführen.

10.7.4.3.4 OnDocumentSaveAs

Event: `OnContextDocumentSaveAs` (`i_strFileName` als String)

Dispatch Id: *1007*

Beschreibung:
Dieses Event wird ausgelöst, wenn dieses Dokument intern unter einem neuen Namen gespeichert wird.

10.7.4.3.5 OnFileChangedAlert

Event: `OnFileChangedAlert` `() als bool`

Dispatch Id: *1003*

Beschreibung:
Dieses Event wird ausgelöst, wenn eine in dieses Control geladene Datei auf der Festplatte von einer anderen Applikation geändert wurde. Clients sollten true zurückgeben, wenn Sie das Event behandelt haben oder false, wenn MapForce es auf die übliche Art behandeln soll, also fragen soll, ob das Dokument neu geladen werden soll.

10.7.4.3.6 OnModifiedFlagChanged

Event: `OnModifiedFlagChanged` (`i_bIsModified` als boolean)

Dispatch Id: *1002*

Beschreibung:
Dieses Event wird immer dann ausgelöst, wenn das Dokument zwischen dem Status "geändert" und "nicht geändert" wechselt. Der Parameter *i_bIsModifed* ist *true,* wenn sich der Inhalt des Dokuments vom ursprünglichen Inhalt unterscheidet und ist andernfalls *false*.

10.7.4.3.7 OnSetEditorTitle

Event: OnSetEditorTitle ()

Dispatch Id: *1006*

Beschreibung:
Dieses Event wird ausgelöst, wenn das enthaltene Dokument intern umbenannt wird.

10.7.5 MapForceControlPlaceHolder

Eigenschaften für alle Arten von Platzhalterfenstern:
`PlaceholderWindowID` 485

Eigenschaften für das Projekt-Platzhalterfenster:
`Project` 485

Methoden für das Projekt-Platzhalterfenster:
`OpenProject` 486
`CloseProject` 486

Das `MapForceControlPlaceHolder` Control dient dazu, die zusätzlichen MapForce Fenster, wie Übersicht, Bibliothek oder das Projektfenster anzuzeigen. Es wird wie jedes andere ActiveX Control verwendet und kann überall in die Client-Applikation platziert werden.

 © 2019-2025 Altova GmbH

10.7.5.1 Eigenschaften

Es sind die folgenden Eigenschaften definiert:

`PlaceholderWindowID` 485

Zugriff auf die MapForceAPI:
`Project` 485

10.7.5.1.1 Label

Eigenschaft: `Label` als String (schreibgeschützt)

Dispatch Id: *1001*

Beschreibung:
Mit dieser Eigenschaft erhalten Sie Zugriff auf den Titel des Platzhalters. Diese Eigenschaft ist schreibgeschützt.

10.7.5.1.2 PlaceholderWindowID

Eigenschaft: `PlaceholderWindowID als`

Dispatch Id: *1*

Beschreibung:
Mit Hilfe dieser Eigenschaft weiß das Objekt, welches MapForce Fenster im Client-Bereich des Control angezeigt werden soll. Die `PlaceholderWindowID` kann jederzeit auf jeden gültigen Wert der Enumeration gesetzt werden. Das Control ändert seinen Status sofort und zeigt das neue MapForce Fenster an.

10.7.5.1.3 Project

Eigenschaft: `Project` als Project (schreibgeschützt)

Dispatch Id: *2*

Beschreibung:
Mit der Eigenschaft `Project` erhalten Sie Zugriff auf das `Project` Objekt der MapForce Automation Server API. Diese Schnittstelle bietet zusätzliche Funktionalitäten, die mit dem in das Control geladenen Projekt verwendet werden können. Die Eigenschaft gibt nur dann eine gültige Projektschnittstelle zurück, wenn das Platzhalterfenster `PlaceholderWindowID` 485 mit einem Wert von `MapForceXProjectWindow (=3)` hat. Die Eigenschaft ist schreibgeschützt.

10.7.5.2 Methoden

Es sind die folgenden Methoden definiert:

OpenProject 486
CloseProject 486

10.7.5.2.1 OpenProject

Methode: `OpenProject` (`strFileName` als string) als boolean

Dispatch Id: *3*

Beschreibung:
`OpenProject` lädt die Datei `strFileName` als das neue Projekt in das Control. Wenn das Platzhalterfenster eine `PlaceholderWindowID` 485 hat, die nicht mit `XMLSpyXProjectWindow (=3)` übereinstimmt, schlägt die Methode fehl.

10.7.5.2.2 CloseProject

Methode: `CloseProject` ()

Dispatch Id: *4*

Beschreibung:
`CloseProject` schließt das in das Control geladene Projekt. Wenn das Platzhalterfenster eine `PlaceholderWindowID` 485 hat, die nicht mit `MapForceXProjectWindow (=3)` übereinstimmt, schlägt die Methode fehl.

10.7.5.3 Events

Das MapForceControlPlaceholder ActiveX Control stellt die folgenden Verbindungspunkt-Events zur Verfügung:

OnModifiedFlagChanged 486

10.7.5.3.1 OnModifiedFlagChanged

Event: `OnModifiedFlagChanged` (`i_bIsModified` als boolean)

Dispatch Id: *1*

Beschreibung:

 © 2019-2025 Altova GmbH

Dieses Event wird nur bei Platzhalter-Controls mit einer `PlaceholderWindowID` 485 von `MapForceXProjectWindow (=3)` ausgelöst. Das Event wird immer dann ausgelöst, wenn der Projektinhalt sich zwischen dem Status "geändert" und "nicht geändert" ändert. Der Parameter *i_bIsModifed* ist *true,* wenn sich der Projektinhalt vom ursprünglichen Inhalt unterscheidet und *false*, wenn dies nicht der Fall ist.

10.7.5.3.2 OnSetLabel

Event: `OnSetLabel` (i_strLabel als String)

Dispatch Id: *1000*

Beschreibung:
Dieses Event wird immer dann ausgelöst, wenn die Bezeichnung eines Platzhalter-Control-Fensters geändert wird.

10.7.6 Enumerationen

Es sind die folgenden Enumerationen definiert:

ICActiveXIntegrationLevel 487

10.7.6.1 ICActiveXIntegrationLevel

Mögliche Werte für die Eigenschaft IntegrationLevel 473 für das MapForceControl.

```
ICActiveXIntegrationOnApplicationLevel  = 0
ICActiveXIntegrationOnDocumentLevel     = 1
```

11 Menübefehle

In diesem Abschnitt finden Sie eine Beschreibung der MapForce-Menübefehle. Es stehen die folgenden Menübefehle zur Verfügung:

- Datei 489
- Bearbeiten 492
- Einfügen 493
- Komponente 496
- Verbindung 498
- Funktion 499
- Ausgabe 500
- Ansicht 502
- Extras 504
- Fenster 517
- Hilfe 519

© 2019-2025 Altova GmbH

11.1 Datei

Dieses Kapitel enthält eine Liste aller Menübefehle im Menü **Datei**.

- Neu

  Erstellt ein neues Mapping-Dokument. In der Professional und der Enterprise Edition können Sie auch ein Mapping-Projekt (`.mfp`) erstellen.

- Öffnen

  Öffnet eine zuvor gespeicherte Mapping-Design-Datei (`.mfd`). In der Professional und der Enterprise Edition können Sie auch ein Mapping-Projekt (`.mfp`) öffnen.

- Speichern/Speichern unter/Alles speichern

  Mit der Option **Speichern** wird das gerade aktive Mapping unter seinem aktuellen Namen gespeichert. Mit der Option **Speichern unter** können Sie das gerade aktive Mapping unter einem anderen Namen speichern. Mit dem Befehl **Alles speichern** werden alle aktuell geöffneten Mapping-Dateien gespeichert.

- Neu laden

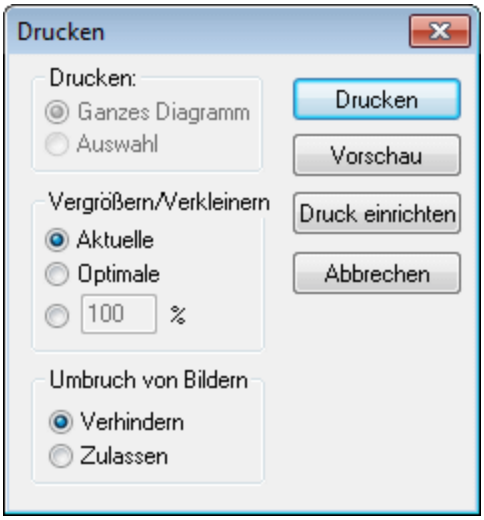
  Durch Neuladen des aktuell aktiven Mappings werden ihre letzten Änderungen rückgängig gemacht.

- Schließen/Alle schließen

  Mit dem Befehl **Schließen** wird das aktuell aktive Mapping geschlossen. Mit dem Befehl **Alle schließen** werden alle gerade offenen Mappings geschlossen. Sie werden gefragt, ob Sie eine der nicht gespeicherten Dateien speichern möchten.

- Drucken/Druckvorschau/Druckereinrichtung

  Mit dem Befehl **Drucken** wird das Dialogfeld **Drucken** (*siehe unten*) aufgerufen, über das Sie Ihr Mapping ausdrucken können. Mit **Aktuelle** wird der aktuell definierte Zoom-Faktor des Mappings beibehalten. Bei Auswahl von **Optimale** wird das Mapping auf Seitengröße vergrößert/verkleinert. Sie können auch einen numerischen Zoom-Faktor angeben. Die Bildlaufleisten der Komponente werden nicht gedruckt. Außerdem können Sie festlegen, ob Grafiken auf mehrere Seiten umbrochen werden sollen oder nicht.



Mit dem Befehl **Druckvorschau** wird dasselbe **Druckdialogfeld** mit denselben Einstellungen wie oben beschrieben, aufgerufen. Mit dem Befehl **Druckereinrichtung** wird das Dialogfeld **Druckereinrichtung** geöffnet, in dem Sie einen Drucker auswählen und die Papiereinstellungen konfigurieren können.

- Mapping validieren

  Mit dem Befehl **Mapping validieren** wird überprüft, ob alle Mappings gültig sind und entsprechende Warn-, Informations- oder Fehlermeldungen angezeigt. Nähere Informationen dazu finden Sie unter Validierung 63.

- Mapping-Einstellungen

  Öffnet das Dialogfeld Mapping-Einstellungen 74, in dem Sie die dokumentspezifischen Einstellungen definieren können.

- Anmeldeinformationen-Manager öffnen (*Enterprise Edition*)

  Öffnet den **Anmeldeinformationen-Manager**, in dem Sie Anmeldeinformationen, die in Mappings mit HTTP-Authentifizierung oder OAuth 2.0-Autorisierung erforderlich sind, verwalten können.

- Code in ausgewählter Sprache generieren/Code generieren in

  Mit dem Befehl **Code in ausgewählter Sprache generieren** wird Code in der in der Symbolleiste ausgewählten Sprache generiert. Mit dem Befehl **Code generieren in <Sprache>** können Sie Code in XSLT 1-3 (*alle Editionen*), XQuery, Java, C# und C++ (*Professional und Enterprise Edition*) generieren. Bei Auswahl eines dieser Befehle wird das Dialogfeld zum **Suchen des Ordners** geöffnet, in dem Sie den Speicherort der generierten Dateien auswählen müssen. Die Namen der generierten-Dateien werden im Dialogfeld "Mapping-Einstellungen" 74 definiert.

  Nähere Informationen über die verfügbaren Transformationssprachen finden Sie unter Transformationssprachen 16. Nähere Informationen dazu finden Sie unter Codegenerierung 65.

- Zu MapForce Server-Ausführungsdatei kompilieren (*Professional und Enterprise Edition*)

 © 2019-2025 Altova GmbH

Generiert eine Datei, die auf MapForce Server ausgeführt werden kann, um die Mapping-Transformation auszuführen

- Auf FlowForce Server bereitstellen (*Professional und Enterprise Edition*)

  Stellt das gerade aktive Mapping auf FlowForce Server bereit.

- Dokumentation generieren (*Professional und Enterprise Edition*)

  Generiert detaillierte Dokumentation zu Ihren Mapping-Projekten in verschiedenen Ausgabeformaten.

- Letzte Dateien

  Zeigt eine Liste der zuletzt geöffneten Dateien an.

- Beenden

  Beendet die Applikation. Sie werden gefragt, ob Sie nicht gespeicherte Dateien speichern möchten.

11.2 Bearbeiten

Dieses Kapitel enthält eine Liste aller Menübefehle im Menü **Bearbeiten**. Die meisten der Befehle in diesem Menü werden erst aktiv, wenn Sie das Ergebnis eines Mappings im **Ausgabe**-Fenster bzw. Code z.B. im **XSLT**-Fenster ansehen.

- Rückgängig

  In MapForce steht eine unbegrenzte Anzahl an "Rückgängig"-Schritten zur Verfügung, mit denen Sie Ihr Mapping Schritt für Schritt wieder rückgängig machen können. Sie können Aktionen auch über die Symbolleisten-Schaltfläche rückgängig machen.

- Wiederherstellen

  Mit Hilfe des Befehls "Wiederherstellen" können Sie zuvor rückgängig gemachte Aktionen wiederholen. Sie können sich innerhalb des Verlaufs der rückgängig gemachten Schritte vorwärts und rückwärts bewegen. Sie können Aktionen auch über die Symbolleisten-Schaltfläche wiederherstellen.

- Suchen

  Dient zum Suchen von bestimmtem Text in den Fenstern **XQuery** (*Professional und Enterprise Edition*), **XSLT**, **XLST2**, **XSLT3** und **Ausgabe**. Sie können die Suche auch über die Symbolleisten-Schaltfläche durchführen.

- Weitersuchen

  Sucht nach der nächsten Stelle, an der der Such-String vorkommt. Sie können dies auch über die Symbolleisten-Schaltfläche tun.

- Vorheriges suchen

  Sucht nach der vorherigen Stelle, an der der Such-String vorkommt. Dies lässt sich auch über die Symbolleisten-Schaltfläche bewerkstelligen.

- Ausschneiden/Kopieren/Einfügen/Löschen

  Mit den Windows-Standardbearbeitungsbefehlen können Sie beliebige im Mapping-Fenster angezeigte Komponenten oder Funktionen ausschneiden, kopieren, einfügen und löschen.

- Alle auswählen

  Markiert alle Komponenten im Fenster **Mapping** oder den Text/Code in den Fenstern **XQuery** (*Professional und Enterprise Edition*), **XSLT**, **XSLT2**, **XSLT3** und **Ausgabe**.

© 2019-2025 Altova GmbH

11.3 Einfügen

Dieses Kapitel enthält eine Liste aller Menübefehle im Menü **Einfügen**.

- XML-Schema/Datei

  Fügt eine XML-Schema- oder Instanzdatei zum Mapping hinzu. Wenn Sie eine XML-Datei ohne Schemareferenz auswählen, kann MapForce ein passendes XML-Schema generieren 130. Wenn Sie ein XML-Schema auswählen, werden Sie aufgefordert optional eine XML-Instanzdatei für die Vorschaudaten zu inkludieren. Sie können eine XML/XSD-Datei auch über die Symbolleisten-Schaltfläche hinzufügen.

- Datenbank (*Professional und Enterprise Edition*)

  Fügt eine Datenbankkomponente hinzu. Sie können eine Datenbankkomponente auch über die Symbolleisten-Schaltfläche hinzufügen. In der MapForce Enterprise Edition können auch NoSQL-Datenbanken als Komponenten hinzugefügt werden.

- EDI (*Enterprise Edition*)

  Fügt eins EDI-Dokument hinzu. Sie können eine EDI-Komponente auch über die Symbolleisten-Schaltfläche hinzufügen.

- Textdatei (*Professional und Enterprise Edition*)

  Fügt ein Flat File-Dokument, wie z.B. eine CSV- oder Textdatei ("FLF" Fixed Length File) hinzu. Sie können eine Textdatei auch über die Symbolleisten-Schaltfläche hinzufügen. In der MapForce Enterprise Edition können Textdateien mit Hilfe von FlexText verarbeitet werden.

- Webservice-Funktion (*Enterprise Edition*)

  Fügt den Aufruf eines Webservice hinzu. Sie können einen Webservice auch über die Symbolleisten-Schaltfläche hinzufügen.

- Excel 2007+-Datei (*Enterprise Edition*)

  Fügt eine Microsoft Excel 2007+ (`.xlsx`)-Datei hinzu. Wenn Sie Excel 2007+ nicht installiert haben, können Sie Dateien trotzdem von/auf Excel 2007+-Dateien mappen. In diesem Fall können Sie zwar keine Vorschau des Ergebnisses im Fenster **Ausgabe** anzeigen, das Ergebnis aber dennoch speichern. Sie können eine Excel-Datei auch über die Symbolleisten-Schaltfläche hinzufügen.

- XBRL-Dokument (*Enterprise Edition*)

  Fügt eine XBRL-Instanz oder ein Taxonomiedokument hinzu. Sie können eine XBRL-Komponente auch über die Symbolleisten-Schaltfläche hinzufügen.

- JSON-Schema/Datei (*Enterprise Edition*)

Fügt ein JSON-Schema- oder eine JSON-Datei hinzu. Sie können eine JSON-Komponente auch über die Symbolleisten-Schaltfläche hinzufügen.

- Protocol Buffers-Datei (*Enterprise Edition*)

  Fügt eine im Protocol Buffers-Format kodierte Binärdatei hinzu. Sie können eine Datei im Protocol Buffers-Format auch über die Symbolleisten-Schaltfläche hinzufügen.

- PDF-Dokument einfügen (*Enterprise Edition*)

  Lädt ein PDF-Dokument. Sie können ein PDF-Dokument auch über die Symbolleisten-Schaltfläche einfügen.

- Input-Komponente einfügen

  Einfache Input-Komponenten können als für das gesamte Mapping relevante Input-Parameter oder nur im Kontext von benutzerdefinierten Funktionen verwendet werden. Nähere Informationen dazu finden Sie unter Einfache Input-Komponente 164 und Parameter in benutzerdefinierten Funktionen 225. Sie können eine einfache Input-Komponente auch über die Symbolleisten-Schaltfläche hinzufügen.

- Output-Komponente einfügen

  Einfache Output-Komponenten können als Output-Komponenten in Mappings und als Output-Parameter von benutzerdefinierten Funktionen verwendet werden. Nähere Informationen dazu finden Sie unter Einfache Output-Komponente 171 und Parameter in benutzerdefinierten Funktionen 225. Sie können eine einfache Output-Komponente auch über die Symbolleisten-Schaltfläche hinzufügen.

- Konstante

  Fügt eine Konstante ein, die Fixdaten für einen Input-Konnektor bereitstellt. Sie können die folgenden Datentypen wählen: `String`, `Zahl` und `alle anderen`. Sie können eine Konstante auch über die Symbolleisten-Schaltfläche einfügen.

- Variable

  Fügt eine Variable 175 ein, die einer regulären (nicht-inline gesetzten) benutzerdefinierten Funktion entspricht. Bei Variablen handelt es sich um eine spezielle Art von Komponente, in der ein Mapping-Zwischenergebnis für die weitere Verarbeitung gespeichert wird. Sie können eine Variable auch über die Symbolleisten-Schaltfläche einfügen.

- Join (*Professional und Enterprise Edition*)

  Mit Hilfe der Join-Komponente können Sie Daten im SQL- und Nicht-SQL-Modus verknüpfen. Sie können eine Join-Komponente auch über die Symbolleisten-Schaltfläche hinzufügen.

- Sortieren: Nodes/Zeilen

  Fügt eine Komponente ein, über die Sie Nodes sortieren können (siehe Sortieren von Daten 188). Sie können eine Sortierkomponente auch über die Symbolleisten-Schaltfläche hinzufügen.

 © 2019-2025 Altova GmbH

- Filter: Nodes/Zeilen

  Fügt eine Filter-Komponente ein, mit der Daten aus jeder anderen von MapForce unterstützten Komponentenstruktur - auch Datenbanken - gefiltert werden können. Nähere Informationen finden Sie unter Filter und Bedingungen 194. Sie können einen Filter auch über die Symbolleisten-Schaltfläche hinzufügen.

- SQL/NoSQL-WHERE/ORDER (*Professional und Enterprise Edition*)

  Fügte eine Komponente ein, mit der Sie Datenbankdaten auf Basis von Bedingungen filtern können. Sie können die SQL/NoSQL-WHERE/ORDER-Komponente auch über die Symbolleisten-Schaltfläche aufrufen.

- Wertezuordnung

  Fügt eine Komponente ein, die einen Input-Wert mittels einer Lookup-Tabelle in einen Output-Wert transformiert. Diese Funktion eignet sich dazu, eine Gruppe von Werten auf eine andere Gruppe von Werten zu mappen (z.B. Zahlen für einen Monat auf Monatsnamen). Nähere Informationen dazu finden Sie unter Wertezuordnungen 200. Sie können eine Wertezuordnung auch über die Symbolleisten-Schaltfläche einfügen.

- IF-Else-Bedingung

  Fügt eine If-Else-Bedingung ein. Dies eignet sich für Szenarien, in denen ein einfacher Wert anhand einer Bedingung verarbeitet werden soll. Nähere Informationen finden Sie unter Filter und Bedingungen 194. Sie können eine If-Else-Bedingung auch über die Symbolleisten-Schaltfläche hinzufügen.

- Ausnahme (*Professional und Enterprise Edition*)

  Mit Hilfe der Ausnahme-Komponente können Sie ein Mapping unterbrechen, wenn eine bestimmte Bedingung erfüllt wird. Sie können eine Ausnahmekomponente auch über die Symbolleisten-Schaltfläche hinzufügen. In der MapForce Enterprise Edition können Sie mit Hilfe dieser Komponente auch Fehlermeldungen in WSDL-Mapping-Projekten definieren.

- Kommentar

  Mit Hilfe dieses Menübefehls können Kommentare im Stil einer Notiz als alleinstehende Komponenten hinzugefügt werden. Nähere Informationen dazu finden Sie unter Kommentare 33. Sie können einen Kommentar auch über die Symbolleisten-Schaltfläche hinzufügen.

11.4 Komponente

Dieses Kapitel enthält eine Liste aller Menübefehle im Menü **Komponente**.

- Root-Element ändern

  Dient zum Ändern des Root-Elements eines XML-Instanzdokuments.

- Schema-Definition in XMLSpy bearbeiten

  Um ein Schema in Altova XMLSpy bearbeiten zu können, müssen Sie auf eine XML-Komponente klicken und die Option **Schema-Definition in XMLSpy bearbeiten** auswählen.

- FlexText-Konfiguration bearbeiten (*Enterprise Edition*)

  Mit diesem Befehl können Sie eine FlexText-Datei bearbeiten.

- Datenbankobjekte hinzufügen/entfernen/bearbeiten (*Professional und Enterprise Edition*)

  Dient zum Hinzufügen, Entfernen oder Ändern von Datenbankobjekten in einer Datenbankkomponente.

- Mapping auf EDI X12 997 erstellen (*Enterprise Edition*)

  Das X12 997 Functional Acknowledgment gibt über den Status des EDI-Datenaustauschs Auskunft. Alle Fehler, die bei der Verarbeitung des Dokuments auftreten, werden hier ausgegeben. MapForce kann automatisch ein X12 997-Dokument generieren, das Sie an den Empfänger senden können.

- Mapping auf EDI X12 999 erstellen (*Enterprise Edition*)

  Das X12 999 Implementation Acknowledgment Transaction Set meldet die Nichterfüllung von HIPAA-Implementierungsrichtlinien oder Applikationsfehler. MapForce kann automatisch eine X12 999-Komponente generieren und die erforderlichen Mapping-Verbindungen erstellen.

- Aktualisieren (*Professional und Enterprise Edition*)

  Lädt die Struktur der gerade aktiven Datenbankkomponente neu.

- Duplikat davor/danach einfügen

  Fügt eine Kopie des ausgewählten Datenelements vor/nach dem aktuell ausgewählten ein. Ein duplizierter Input kann nicht als Datenquelle verwendet werden. Nähere Informationen dazu finden Sie unter Input duplizieren 39.

- Duplikat löschen

  Entfernt ein dupliziertes Datenelement.

- Kommentar/Processing Instruction

  Mit dieser Option können Sie Kommentare und Processing Instructions 140 in XML-Komponenten einfügen.

- Inhalt als CDATA-Abschnitt schreiben

 © 2019-2025 Altova GmbH

Mit diesem Befehl wird ein CDATA-Abschnitt 140 erstellt. CDATA-Abschnitte dienen dazu, Abschnitte eines Dokuments, die normalerweise als Markup interpretiert würden, als Zeichendaten darzustellen.

- Datenbankaktionen (*Professional und Enterprise Edition*)

  Dient zum Konfigurieren von Datenbankeinfüge-, -aktualisierungs- und -löschaktionen und anderer Optionen für Datenbankdatensätze.

- Datenbank abfragen (*Professional und Enterprise Edition*)

  Erstellt anhand der Tabelle bzw. des Felds, die bzw. das Sie in der Datenbankkomponente ausgewählt haben, eine SELECT -Anweisung. Wenn Sie auf eine Tabelle/ein Feld klicken, wird dieser Befehl aktiv und die SELECT -Anweisung wird automatisch in das **Select**-Fenster platziert.

- Links ausrichten

  Richtet die Struktur einer Komponenten linksbündig aus.

- Rechts ausrichten

  Richtet die Struktur einer Komponenten rechtsbündig aus.

- Kommentar bearbeiten

  Wenn Ihr Mapping eine Kommentarkomponente enthält, können Sie diese durch Klick auf die Komponente und Auswahl des Befehls **Kommentar bearbeiten** bearbeiten. Alternativ dazu können Sie in die Kommentarkomponente doppelklicken und den Text direkt im Kommentarfeld bearbeiten. Nähere Informationen zu Kommentarkomponenten und ihren Typen finden Sie unter Kommentar 33.

- Eigenschaften

  Zeigt die Eigenschaften der aktuell ausgewählten Komponente an. Siehe auch Ändern der Komponenteneinstellungen 38.

11.5 Verbindung

Dieses Kapitel enthält eine Liste aller Menübefehle im Menü **Verbindung**.

- Idente Sub-Einträge automatisch verbinden

  Aktiviert bzw. deaktiviert die Option **Idente Sub-Einträge automatisch verbinden**. Nähere Informationen zu Verbindungen und Verbindungsarten finden Sie unter Verbindungen 45.

- Einstellungen für 'Idente Sub-Einträge verbinden'

  Damit können Sie idente Sub-Einträge definieren. Nähere Informationen dazu finden Sie unter Verbindungen mit identen Sub-Einträgen 52.

- Idente Sub-Einträge verbinden

  Mit diesem Befehl können Sie sowohl in Quell- als auch im Zielkomponenten mehrere Verbindungen für Datenelemente **desselben Namens** erzeugen. Die Einstellungen in diesem Dialogfeld werden angewendet, wenn die Symbolleisten-Schaltfläche (**Idente Sub-Einträge automatisch verbinden**) aktiv ist. Nähere Informationen dazu finden Sie unter Verbindungen mit identen Sub-Einträgen 52.

- Zielorientiert (Standard)

  Ändert den Konnektortyp in ein Standard-Mapping. Nähere Informationen dazu finden Sie unter Zielorientierte im Vergleich zu quellorientierten Verbindungen 49.

- Alles kopieren (Sub-Einträge kopieren)

  Erstellt Verbindungen für alle identen Subeinträge. Der Hauptvorteil von "Alles kopieren"-Verbindungen ist, dass der Mapping-Arbeitsbereich dadurch übersichtlicher wird: Anstelle mehrerer Verbindungen wird eine einzige durch eine dicke Linie dargestellte Verbindung erstellt. Nähere Informationen dazu finden Sie unter "Alles kopieren"-Verbindungen 54.

- Quellorientiert (Mixed Content)

  Ändert den Verbindungstyp in eine quellorientierte Verbindung. Dadurch kann gemischter Inhalt (Mixed Content) (Text- und Child-Nodes) automatisch in derselben Reihenfolge, wie er in der XML-*Quelldatei* vorkommt, gemappt werden. Nähere Informationen dazu finden Sie unter Quellorientierte Verbindungen 49 .

- Eigenschaften

  Öffnet das Dialogfeld **Verbindungseinstellungen**, in dem Sie Verbindungsarten und Annotationseinstellungen definieren können. Nähere Informationen dazu finden Sie unter Verbindungseinstellungen 56.

© 2019-2025 Altova GmbH

11.6 Funktion

Dieses Kapitel enthält eine Liste aller Menübefehle im Menü **Funktion**.

- Benutzerdefinierte Funktion erstellen

  Erstellt eine neue benutzerdefinierte Funktion 219 (UDF). Sie können eine benutzerdefinierte Funktion auch über die Symbolleisten-Schaltfläche erstellen.

- Benutzerdefinierte Funktion von Auswahl erstellen

  Erstellt auf Basis der aktuell im Mapping-Fenster ausgewählten Elemente eine benutzerdefinierte Funktion. Nähere Informationen dazu finden Sie unter Erstellen benutzerdefinierter Funktionen 222. Sie können eine benutzerdefinierte Funktion anhand der Auswahl auch über die Symbolleisten-Schaltfläche erstellen.

- Funktionseinstellungen

  Öffnet das Dialogfeld **Benutzerdefinierte Funktion bearbeiten**, wo Sie die aktuellen Einstellungen für die benutzerdefinierte Funktion ändern können. Nähere Informationen dazu finden Sie unter Bearbeiten benutzerdefinierter Funktionen 224.

- Funktion entfernen

  Löscht die aktuell aktive benutzerdefinierte Funktion, wenn Sie in einem Kontext arbeiten, der dies erlaubt.

- Input-Komponente einfügen

  Einfache Input-Komponenten können als für das gesamte Mapping relevante Input-Parameter oder nur im Kontext von benutzerdefinierten Funktionen verwendet werden. Nähere Informationen dazu finden Sie unter Einfache Input-Komponente 164 und Parameter in benutzerdefinierten Funktionen 225. Sie können eine einfache Input-Komponente auch über die Symbolleisten-Schaltfläche hinzufügen.

- Output-Komponente einfügen

  Einfache Output-Komponenten können als Output-Komponenten in Mappings und als Output-Parameter von benutzerdefinierten Funktionen verwendet werden. Nähere Informationen dazu finden Sie unter Einfache Output-Komponente 171 und Parameter in benutzerdefinierten Funktionen 225. Sie können eine einfache Output-Komponente auch über die Symbolleisten-Schaltfläche hinzufügen.

11.7 Ausgabe

Dieses Kapitel enthält eine Liste aller Menübefehle im Menü **Ausgabe**.

- XSLT 1.0/XSLT 2.0/XSLT 3.0/XQuery/Java/C#/C++/Built-In

  Dient zum Definieren der Transformationssprache, in der das Mapping ausgeführt werden soll. Welche Transformationssprachen zur Auswahl stehen, hängt von Ihrer MapForce Edition ab. Nähere Informationen dazu finden Sie unter Transformationssprachen 16. Die Transformationssprache kann auch über die Symbolleiste ausgewählt werden.

- Ausgabedatei validieren

  Validiert die XML-Ausgabedatei anhand des referenzierten Schemas. Siehe Validierung 63.

- Ausgabedatei speichern

  Speichert die aktuell im Fenster **Ausgabe** angezeigten Daten in einer Datei.

- Alle Ausgabedateien speichern

  Speichert alle von dynamischen Mappings 422 generierten Ausgabedateien. Siehe auch Tutorial 4 120.

- Ausgabedatei neu generieren

  Lädt die Daten im Fenster **Ausgabe** neu.

- SQL/NoSQL-Script ausführen (*Professional und Enterprise Edition*)

  Wenn im Fenster **Ausgabe** gerade ein SQL/NoSQL-Script angezeigt wird, wird das Mapping auf die Zieldatenbank unter Berücksichtigung der definierten Tabellenaktionen ausgeführt.

- Lesezeichen einfügen/löschen

  Fügt im Fenster **Ausgabe** an der Cursorposition ein Lesezeichen ein bzw. löscht dieses.

- Nächstes/Vorhergehendes Lesezeichen

  Navigiert im Fenster **Ausgabe** zum nächsten/vorhergehenden Lesezeichen.

- Alle Lesezeichen löschen

  Entfernt im Fenster **Ausgabe** alle derzeit definierten Lesezeichen.

- Pretty-Print

  Formatiert Ihr XML-Dokument im Fenster **Ausgabe** neu, sodass Sie eine strukturierte Übersicht über das Dokument haben. Jedes Subelement ist einen Tabstopp vom übergeordneten Element eingerückt. Die im Dialogfeld Einstellungen für Textansicht 67 definierten Tabulatoreinstellungen (Gruppe "Tabulatoren") werden im Fenster **Ausgabe** angewendet.

- Einstellungen für Textansicht

© 2019-2025 Altova GmbH

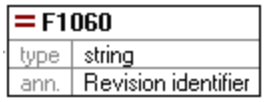
Ruft das Dialogfeld **Einstellungen für Textansicht** auf. In diesem Dialogfeld können Sie die Textansichtseinstellungen in den Fenstern **XQuery** (*Professional und Enterprise Edition*), **Ausgabe** und **XSLT** anpassen. Außerdem sehen Sie darin die aktuell für das Fenster geltenden Tastaturkürzel. Nähere Informationen dazu finden Sie unter Funktionalitäten der Textansicht 67.

11.8 Ansicht

Dieses Kapitel enthält eine Liste aller Menübefehle im Menü **Ansicht**.

- Annotationen anzeigen

  Zeigt Annotationen in der Komponente an. Sie können diese Option auch über die Symbolleisten-Schaltfläche aktivieren. Wenn auch das Symbol **Datentypen anzeigen** aktiv ist, werden beide Informationen in Tabellenform angezeigt (*siehe Abbildung unten*). Sie können Verbindungen auch mit Hilfe von Annotationen beschriften. Nähere Informationen dazu finden Sie unter Verbindungseinstellungen 57.



- Datentypen anzeigen

  Zeigt die Datentypen in einer Komponente an. Sie können diese Option auch über die Symbolleisten-Schaltfläche aktivieren. Wenn auch das Symbol **Datentypen anzeigen** aktiv ist, werden beide Informationen in Tabellenform angezeigt (*siehe "Annotationen anzeigen" oben*).

- Bibliothek in Funktionstitelleiste anzeigen

  Zeigt den Namen der Bibliothek in der Funktionsüberschrift an. Sie können diese Option auch über die Symbolleisten-Schaltfläche aktivieren.

- Tipps anzeigen

  Zeigt einen Tooltip mit erklärendem Text an, wenn Sie den Mauszeiger über eine Funktionsüberschrift platzieren. Wenn die Option **Tipps anzeigen** aktiv ist, werden auch Informationen über Datentypen in einer Komponente angezeigt.

- XBRL-Anzeigeoptionen (*Enterprise Edition*)

  Sie können in MapForce die folgenden XBRL-Einstellungen konfigurieren:

  - die Label-Sprache für die XBRL Items und ihre Annotationen
  - die bevorzugten Label-Roles für XBRL Item-Namen
  - den spezifischen Typ von Label-Roles von Annotationen für XBRL Items
  - Benutzerdefinierte XBRL-Taxonomiepakete

- Ausgewählte Komponentenkonnektoren anzeigen/Quell- und Zielkonnektoren anzeigen

  Mit Hilfe dieser Optionen können Verbindungen selektiv hervorgehoben werden. Eine Erläuterung dazu finden Sie unter Verbindungen 47.

- Vergrößern/Verkleinern

  Öffnet das Dialogfeld **Vergrößern/Verkleinern**, Sie können entweder einen numerischen Zoom-Faktor eingeben oder den Zoom-Faktor mit Hilfe des Schiebereglers interaktiv einstellen.

 © 2019-2025 Altova GmbH

- Zurück/Vorwärts

  Mit den Befehlen **Zurück** und **Vorwärts** können Sie zum vorherigen und nächsten Mapping, an dem Sie relativ zu aktuell geöffneten Mapping gearbeitet haben, wechseln.

- Statusleiste

  Blendet die unterhalb des Fensters **Meldungen** angezeigte **Statusleiste** ein oder aus.

- Bibliotheken/Bibliotheken verwalten

  Klicken Sie auf **Bibliotheken**, um das **Bibliotheksfenster** ein- bzw. auszublenden. Klicken Sie auf **Bibliotheken verwalten**, um das Fenster **Bibliotheken verwalten** ein- oder auszublenden.

- Meldungen

  Blendet das Fenster "Meldungen" 24 ein oder aus. Bei Generierung von Code wird das Fenster **Meldungen** automatisch aktiviert, um das Ergebnis der Validierung anzuzeigen.

- Übersicht

  Blendet das Übersichtsfenster 23 ein oder aus. Ziehen Sie das Rechteck mit der Maus, um in Ihrem Mapping zu navigieren.

- Projektfenster (*Professional und Enterprise Edition*)

  Blendet das **Projektfenster** ein oder aus.

- Debug-Fenster (*Professional und Enterprise Edition*)

  Im Debug-Modus können Sie den Kontext, in dem ein bestimmter Wert erzeugt wird, analysieren. Diese Informationen stehen direkt im Mapping und in den Fenstern **Werte**, **Kontext** und **Breakpoints** zur Verfügung.

11.9 Extras

Dieses Kapitel enthält eine Liste aller Menübefehle im Menü **Extras**.

Globale Ressourcen

Öffnet das Dialogfeld **Globale Ressourcen verwalten**, in dem Sie Einstellungen, die für mehrere Altova-Applikationen verwendet werden können, hinzufügen, bearbeiten und löschen können (siehe Globale Altova-Ressourcen 432).

Aktive Konfiguration

Dient zum Auswählen der aktuell aktiven globalen Ressourcenkonfiguration aus einer Liste von Konfigurationen. Informationen zum Erstellen und Konfigurieren verschiedener Arten von globalen Ressourcen finden Sie unter Globale Altova-Ressourcen 432.

Umgekehrtes Mapping erstellen

Erstellt anhand des gerade in MapForce aktiven Mappings ein "umgekehrtes" Mapping, d.h. die Quellkomponente wird zur Zielkomponente und die Zielkomponente zur Quellkomponente. Beachten Sie, dass im umgekehrten Mapping nur direkte Verbindungen zwischen Komponenten beibehalten werden. Das neue Mapping wird wahrscheinlich nicht gültig sein und kann wahrscheinlich nicht im Fenster **Ausgabe** angezeigt werden. Das neue Mapping müsste vorher manuell bearbeitet werden.

Die folgenden Daten bleiben erhalten:

- direkte Verbindungen zwischen Komponenten
- direkte Verbindungen zwischen Komponenten in einem verketteten Mapping
- die Art der Verbindung 49: Standard, gemischter Inhalt, Alles kopieren
- Einstellungen für die Weiterleitung einer Komponente
- Datenbankkomponenten (*Professional und Enterprise Edition*)

Die folgenden Daten bleiben *nicht* erhalten:

- Verbindungen über Funktionen, Filter, usw.
- Benutzerdefinierte Funktionen
- Webservice-Komponenten (*Enterprise Edition*)

XBRL-Taxonomie-Manager (*Enterprise Edition*)

Der XBRL-Taxonomie-Manager ist ein Tool, mit dem Sie XBRL-Taxonomien installieren und verwalten können.

XML-Schema-Manager

Der XML-Schema-Manager ist ein Altova-Tool, mit dem Sie XML-Schemas (DTDs für XML-Dateien und XML-Schemas) zentral installieren und verwalten können, um diese in allen XML-fähigen Applikationen von Altova verwenden zu können. Nähere Informationen dazu finden Sie unter Schema-Manager 147.

Anpassen

Über diese Option können Sie die grafische Benutzeroberfläche von MapForce anpassen. So können Sie hier etwa Symbolleisten ein- und ausblenden und die Menüs 505 und Tastaturkürzel 506 bearbeiten.

© 2019-2025 Altova GmbH

- Symbolleisten und Fenster wiederherstellen

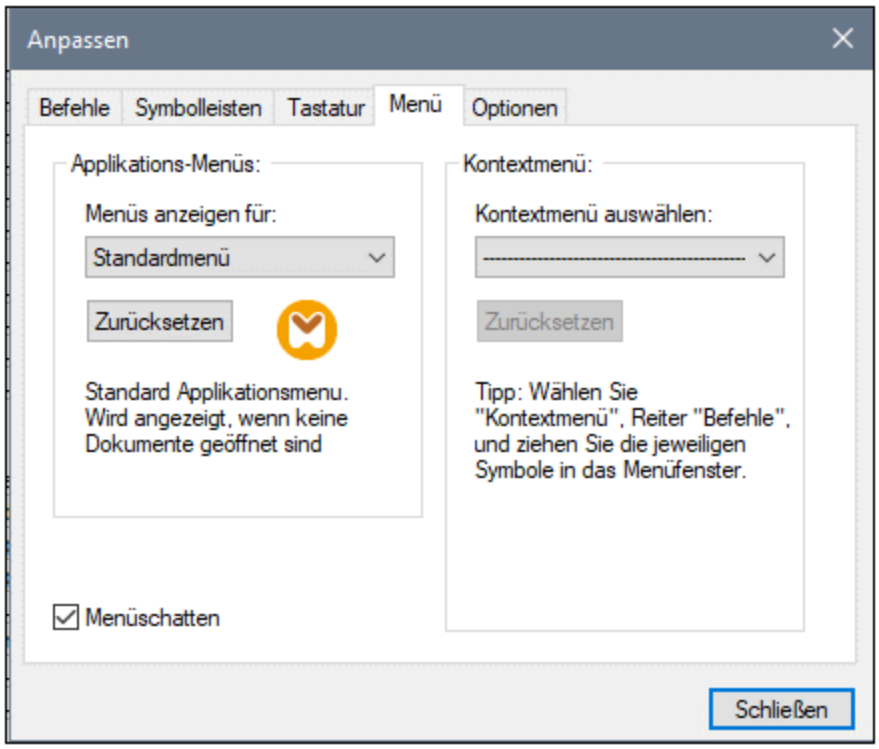
  Setzt die Symbolleisten, Eingabehilfefenster, angedockten Fenster usw. wieder auf ihre Standardeinstellung zurück. MapForce muss neu gestartet werden, damit die Änderungen wirksam werden.

- Optionen

  Ruft das Dialogfeld **Optionen** auf, in dem Sie die Standardeinstellungen von MapForce ändern können. Nähere Informationen dazu finden Sie unter Optionen 508.

11.9.1 Anpassen von Menüs

Sie können MapForce-Standardmenüs und Kontextmenüs anpassen (um z.B. Befehle hinzufügen, ändern oder entfernen). Sie können Ihre Änderungen auch in den Standardzustand zurücksetzen (**Zurücksetzen**). Um Menüs anzupassen, klicken Sie auf **Extras | Anpassen** und anschließend auf das Register **Menü** (*siehe Abbildung unten*).



Standardmenü vs. MapForce Design
Das *Standardmenü* ist die Menüleiste, die angezeigt wird, wenn kein Dokument im Hauptfenster geöffnet ist. Das *MapForce Design*-Menü ist die Menüleiste, die angezeigt wird, wenn ein oder mehrere Mappings geöffnet sind. Jede der beiden Menüleisten kann separat angepasst werden. Änderungen, die an einer Menüleiste vorgenommen wurden, haben keine Auswirkung auf die andere Menüleiste.

Um eine Menüleiste anzupassen, wählen Sie diese in der Auswahlliste *Menüs anzeigen für* aus. Klicken Sie anschließend auf das Register **Befehle** und ziehen Sie die Befehle aus dem Listenfeld *Befehle* in die Menüleiste oder in eines der Menüs.

Löschen von Befehlen aus Menüs
So löschen Sie ein ganzes Menü oder einen Befehl in einem Menü:

1. Wählen Sie aus der Dropdown-Liste *Menüs anzeigen für Standardmenü* oder *MapForce* Design aus.
2. Während das Dialogfeld **Anpassen** geöffnet ist, wählen Sie einen Symbolleisten-Befehl aus, den Sie löschen möchten oder wählen Sie einen Befehl aus, den Sie aus einem der Menüs löschen möchten.
3. Ziehen Sie den Symbolleisten-Befehl mit der Maus aus der Symbolleiste bzw. aus dem Menü. Klicken Sie alternativ dazu mit der rechten Maustaste auf den Symbolleisten-Befehl oder den Menübefehl und wählen Sie **Löschen** aus.

Sie können jede Menüleiste in den Standardzustand zurücksetzen. Wählen Sie sie dazu aus der Dropdown-Liste *Menüs anzeigen für* aus und klicken Sie anschließend auf die Schaltfläche **Zurücksetzen**.

Anpassen von Kontextmenüs
Kontextmenüs sind die Menüs, die angezeigt werden, wenn Sie mit der rechten Maustaste auf bestimmte Objekte auf der Benutzeroberfläche der Applikation klicken. Jedes dieser Kontextmenüs kann folgendermaßen angepasst werden:

1. Wählen Sie das Kontextmenü in der Dropdown-Liste *Kontextmenü auswählen* aus. Daraufhin wird das entsprechende Kontextmenü geöffnet.
2. Klicken Sie auf das Register **Befehle** und ziehen Sie den gewünschten Befehl aus dem Listenfeld *Befehle* in das Kontextmenü.
3. Um einen Befehl aus dem Kontextmenü zu löschen, klicken Sie mit der rechten Maustaste darauf und wählen Sie den Befehl **Löschen**. Ziehen Sie den Befehl alternativ dazu mit der Maus aus dem Kontextmenü heraus.

Sie können jedes Kontextmenü in den Standardzustand zurücksetzen. Wählen Sie es dazu aus der Dropdown-Liste *Kontextmenü auswählen* aus und klicken Sie anschließend auf die Schaltfläche **Zurücksetzen**.

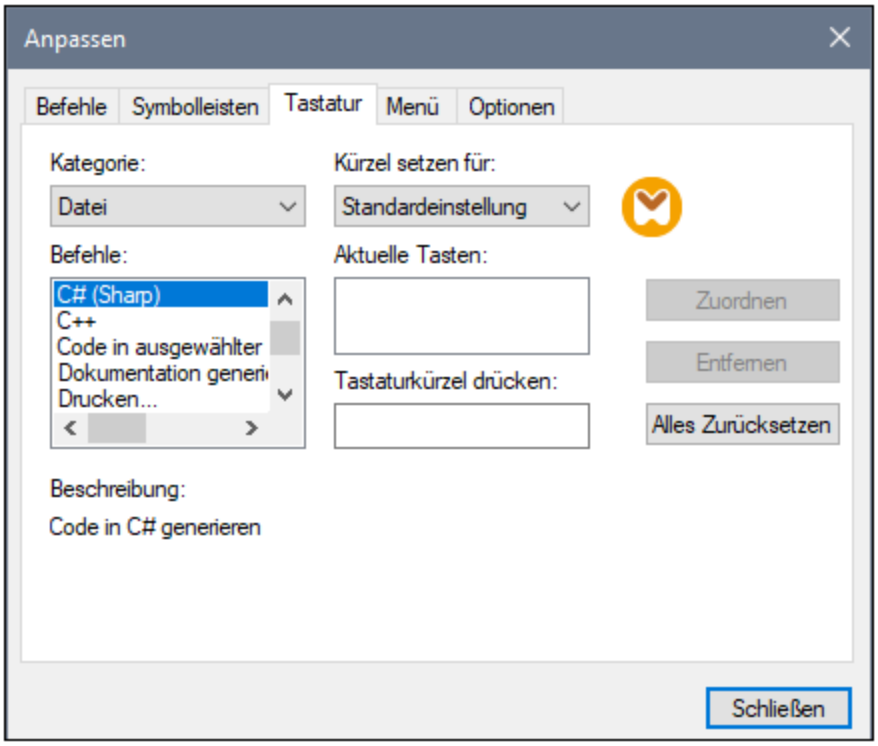
Menüschatten
Aktivieren Sie das Kontrollkästchen *Menüschatten*, wenn Menüs mit Schatten dargestellt werden sollen.

11.9.2 Anpassen von Tastaturkürzeln

Sie können Tastaturkürzel in MapForce folgendermaßen definieren oder ändern: Wählen Sie den Befehl **Extras | Anpassen** und klicken Sie auf das Register **Tastatur**. Um einem Befehl ein neues Tastaturkürzel zuzuweisen, gehen Sie folgendermaßen vor:

1. Wählen Sie den Befehl **Extras | Anpassen** und klicken Sie auf das Register **Tastatur** (*siehe Abbildung unten*).
2. Klicken Sie auf die Auswahlliste *Kategorie*, um den Menünamen auszuwählen.
3. Wählen Sie in der Liste *Befehle* den Befehl aus, dem Sie ein neues Tastenkürzel zuweisen möchten.
4. Geben Sie in das Textfeld *Tastaturkürzel drücken* die neuen Tastaturkürzel ein und klicken Sie auf **Zuordnen**.

 © 2019-2025 Altova GmbH



Um den Eintrag im Textfeld *Tastaturkürzel drücken* zu löschen, drücken Sie eine der Steuerungstasten **Strg**, **Alt** oder **Umschalt**. Um eine Tastenzuweisung zu löschen, klicken Sie unter *Aktuelle Tasten* auf das gewünschte Tastaturkürzel und anschließend auf **Entfernen**.

Anmerkung: Die Option *Kürzel setzen für* hat derzeit keine Funktion.

Tastaturkürzel

MapForce bietet standardmäßig die folgenden Tastaturkürzel:

F1	Hilfe
F2	Nächstes Lesezeichen (im Fenster "Ausgabe")
F3	Weitersuchen
F10	Menüleiste aktivieren
Num +	Aktuellen Node erweitern
Num -	Node reduzieren
Num *	Alle unterhalb des aktuellen Node erweitern
STRG + TAB	Wechselt zwischen offenen Mappings
STRG + F6	Wechselt zwischen offenen Fenstern
STRG + F4	Schließt das aktive Mapping-Dokument
Alt + F4	Schließt MapForce
Alt + F, F, 1	Öffnet die letzte Datei
Alt + F, T, 1	Öffnet das letzte Projekt

STRG + N	Datei neu
STRG + O	Datei öffnen
STRG + S	Datei speichern
STRG + P	Datei drucken
STRG + A	Alles markieren
STRG + X	Ausschneiden
Strg + C	Kopieren
STRG + V	Einfügen
STRG + Z	Rückgängig machen
STRG + Y	Wiederherstellen
Entf	Komponente löschen (mit Betätigungsmeldung)
Umschalt + Entf	Komponente löschen (ohne Bestätigungsmeldung)
STRG + F	Suchen
F3	Weitersuchen
Umschalt+F3	Vorheriges suchen
Pfeiltasten	
(nach oben / nach unten)	Nächstes Datenelement der Komponente auswählen
Esc	Bearbeitungen verwerfen/Dialogfeld schließen
Rückgabewert	Bestätigt eine Auswahl
Tastaturkürzel im Fenster "Ausgabe"	
STRG + F2	Lesezeichen einfügen/löschen
F2	Nächstes Lesezeichen
Umschalt + F2	Vorheriges Lesezeichen
Strg + Umschalt + F2	Alle Lesezeichen löschen
Zoom-Tasten	
Strg + Mausrad vorwärts	Vergrößern
Strg + Mausrad rückwärts	Verkleinern
Strg + 0 (Null)	Zoom zurücksetzen

11.9.3 Optionen

Sie können die allgemeinen Einstellungen und anderen Einstellungen in MapForce mit dem Befehl **Extras | Optionen** ändern. Weiter unten finden Sie eine Beschreibung der verfügbaren Optionen.

Allgemein

Im Abschnitt *Allgemein* können Sie die folgenden Optionen definieren:

- *Logo anzeigen | Beim Start anzeigen:* Hier können Sie definieren, ob ein Bild (Willkommensbildschirm) beim Start von MapForce angezeigt werden soll oder nicht.

 © 2019-2025 Altova GmbH

- Im Abschnitt *Mapping-Ansicht* können Sie die folgenden Parameter definieren:
  - Sie können die Anzeige des schattierten Hintergrunds im Mapping-Fenster aktivieren/deaktivieren (*Schattierten Hintergrund anzeigen*).
  - Sie können die Anzeige von Annotationen auf N Zeilen beschränken (*Anzeige der Annotation einschränken auf*). Wenn Sie diese Option z.B. auf 2 gesetzt haben und Ihr Annotationstext 3 Zeilen enthält, werden nur die ersten zwei Zeilen des Annotationstexts im Mapping angezeigt. Diese Einstellung gilt auch für SELECT-Anweisungen, die in einer Komponente angezeigt werden.
  - Sie können auch die Anzeige von Komponentenkommentaren [31] auf N Zeilen beschränken (*Anzeige des Kommentars einschränken auf*). Wenn Sie die Anzeige von Kommentaren etwa auf 1 Zeile beschränkt haben und Ihr Kommentar mehr als eine Zeile enthält, wird in der Kommentarzeile nur die ersten Zeile angezeigt. Wenn Sie die Eigenschaft auf `0` setzen, wird die Anzeige von Komponentenkommentaren komplett unterdrückt. Beachten Sie, dass die Option *Anzeige des Kommentars einschränken auf* keinen Einfluss auf Kommentarkomponenten [33] hat.

- *Standardkodierung für neue Komponenten.*

  Kodierungsname: Die Standardkodierung für neue XML-Dateien kann durch Auswahl einer Option aus der Dropdown-Liste festgelegt werden. Wird eine 2- oder 4-Byte-Kodierung als standardmäßige Kodierung ausgewählt (z.B. UTF-16, UCS-2 oder UCS-4) können Sie weiters wählen, ob die Byteordnung für XML-Dokumente Little Endian oder Big Endian sein soll. Diese Einstellung kann auch für jede Komponente einzeln geändert werden (siehe Ändern der Komponenteneinstellungen [38]).

  Bytefolge: Wenn ein Dokument mit einer 2-Byte- oder 4-Byte-Zeichenkodierung gespeichert wird, kann es entweder mit der Byteordnung Little-Endian oder Big-Endian gespeichert werden. Außerdem kann festgelegt werden, ob eine Bytefolge-Markierung inkludiert werden soll.

- *Einstellungen Vorschau:* Mit der Option *Timeout benutzen* wird ein Ausführungs-Timeout für die Vorschau auf das Mapping-Ergebnis im Fenster **Ausgabe** definiert.

- *Bei Aktivieren des Ausgabefensters:* Sie können die Ausgabe in temporären Dateien generieren oder sie direkt in eine Ausgabedatei schreiben (*siehe unten*).

  Output-Datei als temporäre Datei generieren: Dies ist die Standardoption. Wenn der Ausgabepfad Ordner enthält, die noch nicht vorhanden sind, erstellt MapForce diese. Für die Professional und Enterprise Edition: Falls Sie beabsichtigen, das Mapping für die Ausführung auf einem Server bereitzustellen, müssen die Verzeichnisse im Pfad auf dem Server vorhanden sein, da bei der Ausführung sonst ein Fehler auftritt.

  Direkt in die endgültigen Output-Dateien schreiben Wenn der Ausgabepfad Ordner enthält, die noch nicht vorhanden sind, kommt es zu einem Fehler. Mit dieser Option werden vorhandene Ausgabedateien ohne vorherige Bestätigung überschrieben.

- *Text in Schritten von N Millionen Zeichen anzeigen:* Definiert die maximale Größe von Text, der im Fenster **Ausgabe** angezeigt wird, wenn eine Vorschau auf Mappings, mit denen große XML- und Textdateien generiert werden, angezeigt wird. Wenn der Ausgabetext diesen Wert übersteigt, müssen Sie auf die Schaltfläche **Mehr laden** klicken, um den nächsten Block zu laden. Nähere Informationen dazu finden Sie unter Vorschau und Validieren der Ausgabe [63].

Bearbeiten

Im Abschnitt *Bearbeiten* können Sie Optionen für die Anzeige von Mappings definieren:

- *Komponenten beim Ziehen mit der Maus aneinander ausrichten:* Hier wird definiert, ob Komponenten oder Funktionen im Mapping-Fenster beim Ziehen mit der Maus an anderen Komponenten ausgerichtet werden sollen (siehe Ausrichten von Komponenten 39).

- *Intelligente Komponentenlöschung:* Sie können Verbindungen in MapForce auch nach Löschung einiger Transformationskomponenten 31 beibehalten. Vor allem die Beibehaltung von Verbindungen mit mehreren Child-Verbindungen kann sich als nützlich erweisen, da Sie dadurch nach Löschung einer Transformationskomponente nicht jede einzelne Child-Verbindung manuell wiederherstellen müssen. Nähere Informationen dazu finden Sie unter Beibehalten von Konnektoren nach Löschen von Komponenten 61.

Meldungen

Im Abschnitt *Meldungen* können Sie die Anzeige von Meldungen wie z.B. den Vorschlag übergeordnete Datenelemente zu verbinden, eine Meldung über die Erstellungen mehrerer Zielkomponenten anzuzeigen, usw. aktivieren.

Code-Generierung (*Professional und Enterprise Edition*)

Im Abschnitt *Code-Generierung* können Sie die Einstellungen für die Generierung von Programmcode und MapForce Server-Ausführungsdateien definieren.

Java

Wenn Sie eine Java Virtual Machine verwenden, die keinen Installer hat und keine Registry-Einträge erstellt (z.B. OpenJDK von Oracle), müssen Sie eventuell einen benutzerdefinierten Java VM-Pfad angeben. Auch wenn Sie automatisch von MapForce ermittelte Java VM-Pfade außer Kraft setzen müssen, müssen Sie diesen Pfad eventuell definieren. Nähere Informationen dazu finden Sie unter Java 512.

XBRL (*Enterprise Edition*)

Sie können in MapForce die folgenden allgemeinen applikationsweiten XBRL-Einstellungen konfigurieren:

- die Label-Sprache für die XBRL Items und ihre Annotationen
- die bevorzugten Label-Roles für XBRL Item-Namen
- den spezifischen Typ von Label-Roles von Annotationen für XBRL Items
- Benutzerdefinierte XBRL-Taxonomiepakete

Debugger (*Professional und Enterprise Edition*)

Im Abschnitt *Debugger* können Sie die folgenden Debugeinstellungen definieren:

- *Maximale Länge gespeicherter Werte:* Definiert die String-Länge von Werten, die im Fenster **Werte** angezeigt werden (mindestens 15 Zeichen). Beachten Sie, dass es zu Arbeitsspeicherproblemen kommen kann, wenn Sie die Länge der gespeicherten Werte auf einen hohen Wert setzen.

© 2019-2025 Altova GmbH

- *Vollständigen Verlauf der Ablaufverfolgung behalten:* Wenn Sie diese Option aktivieren, speichert MapForce während des Debuggens den Verlauf aller Werte, die von allen Konnektoren aller Komponenten im Mapping verarbeitet wurden. In diesem Fall werden alle von Beginn des Debuggens an verarbeiteten Werte im Arbeitsspeicher gespeichert, so dass diese für Ihre Analyse im Fenster **Werte** zur Verfügung stehen, bis Sie den Debug-Vorgang beenden. Es ist nicht empfehlenswert, diese Option beim Debuggen von Mappings mit vielen Daten zu aktivieren, da dies den Debug-Vorgang verlangsamen und zu viel Arbeitsspeicher beanspruchen könnte. Wenn diese Option deaktiviert ist, speichert MapForce nur den jüngsten Debug-Verlauf für Nodes, die mit der aktuellen Ausführungsposition in Zusammenhang stehen.

Datenbank (*Professional und Enterprise Edition*)

Im Abschnitt *Datenbank* können Sie Datenbankabfrageeinstellungen definieren.

Netzwerk-Proxy

Im Abschnitt *Netzwerk-Proxy* können Sie die benutzerdefinierten Proxy-Einstellungen konfigurieren. Diese Einstellungen legen fest, wie sich die Applikation mit dem Internet verbindet. Standardmäßig werden die Proxy-Einstellungen des Systems verwendet, d.h. die Einstellungen funktionieren, ohne dass der Benutzer etwas daran ändern muss. Nähere Informationen dazu finden Sie unter Netzwerk-Proxy 514.

Hilfe

MapForce bietet eine Hilfe (Benutzerhandbuch) in zwei Formaten:

- eine Online-Hilfe im HTML-Format. Diese steht auf der Altova-Website zur Verfügung. Um die Online-Hilfe aufrufen zu können, benötigen Sie Internet-Zugriff.
- eine Hilfedatei im PDF-Format, die bei der Installation von MapForce auf Ihrem Rechner installiert wird. Sie hat den Namen `MapForce.pdf` und befindet sich im Applikationsordner (im Ordner "Programme"). Wenn Sie keinen Internet-Zugriff haben, können Sie immer diese lokal gespeicherte Hilfedatei öffnen.

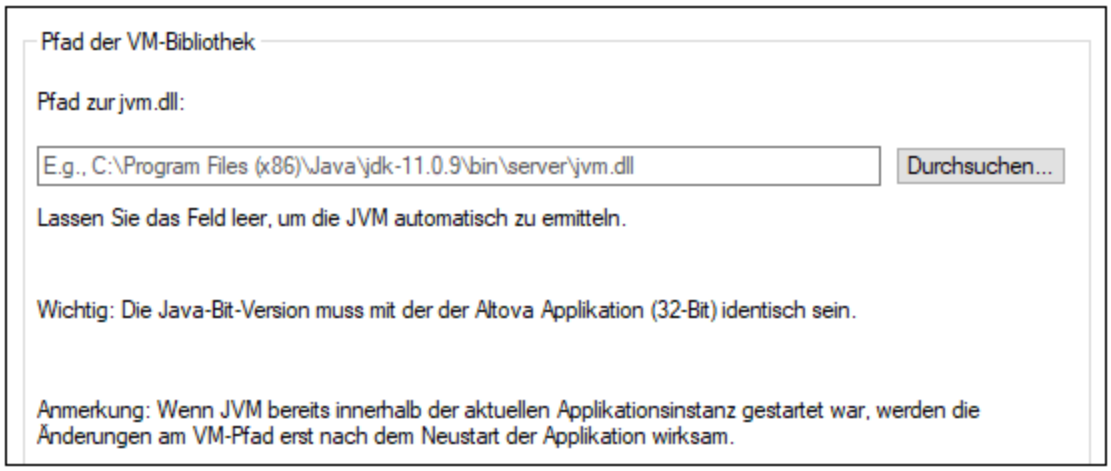
Über die Option Hilfe (*Abbildung unten*) können Sie auswählen, welches der beiden Formate geöffnet werden soll, wenn Sie im Menü **Hilfe** auf den Befehl **Hilfe (F1)** klicken.

Sie können diese Option jederzeit ändern. Über die Links in diesem Abschnitt (*siehe Abbildung oben*) können Sie das entsprechende Hilfeformat öffnen.

11.9.3.1 Java

Im Abschnitt *Java* (*siehe Abbildung unten*) haben Sie die Möglichkeit, den Pfad zu einer Java VM (Virtual Machine) auf Ihrem Dateisystem einzugeben. Beachten Sie, dass dies nicht immer notwendig ist. MapForce versucht standardmäßig den Java VM-Pfad automatisch zu ermitteln. Dazu wird zuerst die Windows Registry und anschließend die JAVA_HOME-Umgebungsvariable gelesen. Ein in dieses Dialogfeld eingegebener benutzerdefinierter Pfad hat Vorrang vor allen automatisch ermittelten Java VM-Pfaden.

Wenn Sie eine Java Virtual Machine verwenden, die keinen Installer hat und keine Registry-Einträge erstellt (z.B. OpenJDK von Oracle), müssen Sie eventuell einen benutzerdefinierten Java VM-Pfad angeben. Auch wenn Sie automatisch von MapForce ermittelte Java VM-Pfade aus irgendeinem Grund außer Kraft setzen müssen, müssen Sie diesen Pfad eventuell definieren.



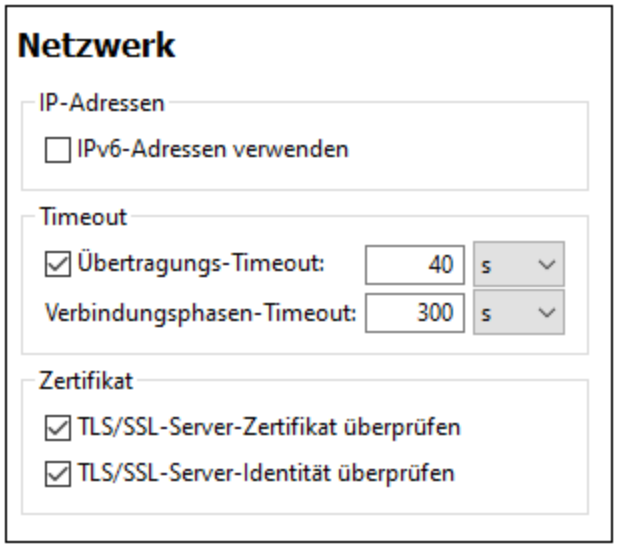
Beachten Sie dazu Folgendes:

- Der Java VM-Pfad wird gemeinsam von allen Altova Desktop-Applikationen (nicht aber den Server-Applikationen) verwendet. Wenn Sie den Pfad daher in einer Applikation ändern, gilt dies automatisch auch für alle anderen Altova-Applikationen.
- Der Pfad muss auf die Datei **`jvm.dll`** im Verzeichnis **`\bin\server`** oder **`\bin\client`** (relativ zum Verzeichnis, in dem JDK installiert ist) verweisen.
- Die MapForce-Plattform (32-Bit, 64-Bit) muss mit der des JDK identisch sein.
- Nachdem Sie den Java VM-Pfad geändert haben, müssen Sie MapForce eventuell neu starten, damit die neuen Einstellungen wirksam werden.

 © 2019-2025 Altova GmbH

11.9.3.2 Netzwerk

Im Abschnitt **Netzwerk** (*Abbildung unten*) können Sie wichtige Netzwerkeinstellungen konfigurieren.



IP-Adressen
Wenn Host-Namen in gemischten IPv4/IPv6-Netzwerken zu mehr als einer Adresse aufgelöst werden, werden bei Auswahl dieser Option die IPv6-Adressen verwendet. Wenn die Option in solchen Umgebungen nicht aktiviert ist und IPv4-Adressen zur Verfügung stehen, werden IPv4-Adressen verwendet.

Timeout

- *Übertragungs-Timeout:* Wenn bei der Übertragung zweier beliebiger aufeinander folgender Datenpakete einer Übertragung (bei Sendung oder Empfang) dieses Limit erreicht wird, wird die gesamte Übertragung abgebrochen. Die Werte können in Sekunden [s] oder Millisekunden [ms] angegeben werden, wobei der Standardwert 40 Sekunden beträgt. Wenn die Option nicht aktiviert ist, gibt es keinen Grenzwert, ab dem eine Übertragung abgebrochen wird.
- *Verbindungsphasen-Timeout:* Dies ist das Zeitlimit, innerhalb dessen die Verbindung (inklusive Sicherheitshandshake) hergestellt worden sein muss. Die Werte können in Sekunden [s] oder Millisekunden [ms] angegeben werden, wobei der Standardwert 300 Sekunden beträgt. Dieses Timeout kann nicht deaktiviert werden.

Zertifikat

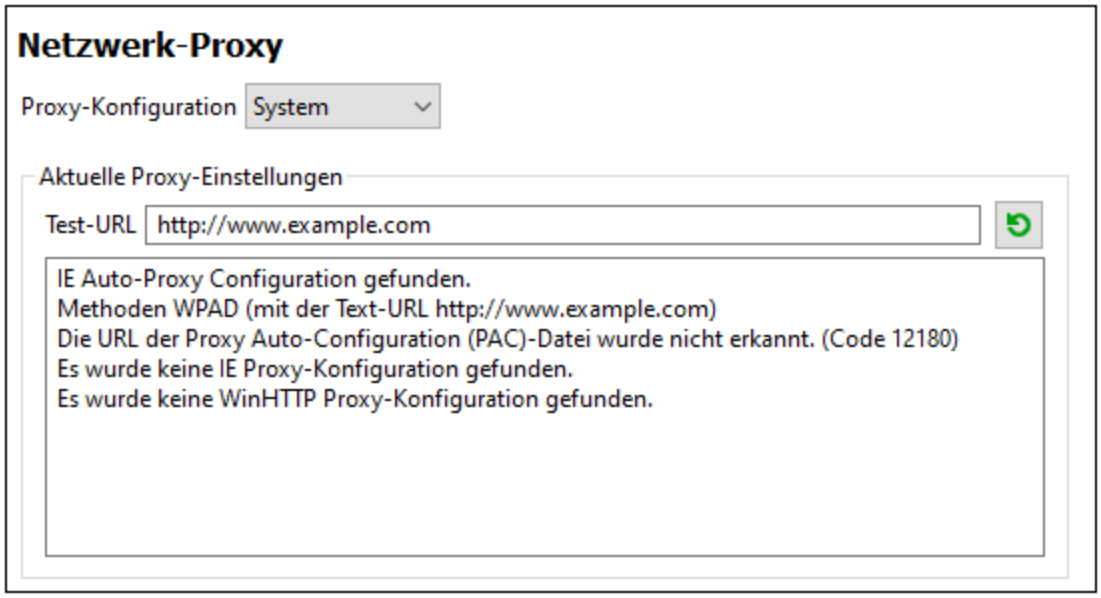
- *TLS/SSL-Server-Zertifikat überprüfen* Wenn diese Option aktiviert ist, wird die Authentizität des Server-Zertifikats überprüft, indem die digitale Signaturkette überprüft wird, bis ein vertrauenswürdiges Root-Zertifikat erreicht wird. Diese Option ist standardmäßig aktiviert. Wenn diese Option nicht aktiviert wird, ist die Kommunikation nicht sicher. Angriffe (z.B. ein Man-in-the-Middle-Angriff) würden nicht erkannt. Beachten Sie, dass mit dieser Option nicht überprüft wird, ob das Zertifikat tatsächlich das Zertifikat für den Server, mit dem kommuniziert wird, ist. Um eine umfassende Sicherheit zu gewährleisten, müssen sowohl das Zertifikat als auch die Identität überprüft werden (*siehe nächste Option*).
- *TLS/SSL-Server-Identität überprüfen* Wenn diese Option ausgewählt ist, wird überprüft, ob das Server-Zertifikat zu dem Server gehört, mit dem kommuniziert werden soll. Dazu wird überprüft, ob der Server-Name in der URL mit dem Namen im Zertifikat übereinstimmt. Diese Option ist standardmäßig aktiviert. Wenn diese Option nicht aktiviert ist, wird die Identität des Servers nicht überprüft. Beachten

Sie, dass das Zertifikat des Servers mit dieser Option nicht überprüft wird. Um eine umfassende Sicherheit zu gewährleisten, müssen sowohl das Zertifikat als auch die Identität überprüft werden (*siehe vorhergehende Option*).

11.9.3.3 Netzwerk-Proxy

Im Abschnitt *Netzwerk-Proxy* können Sie die benutzerdefinierten Proxy-Einstellungen konfigurieren. Diese Einstellungen beeinflussen, wie die Applikation eine Verbindung mit dem Internet herstellt (z.B. zur XML-Validierung). Standardmäßig werden die Proxy-Einstellungen des Systems verwendet, d.h. die Einstellungen funktionieren, ohne dass der Benutzer etwas daran ändern muss. Falls nötig, können Sie jedoch einen anderen Netzwerk-Proxy-Server definieren. Wählen Sie dazu in der Auswahlliste *Proxy-Konfiguration* entweder die Option *Automatisch* oder *Manuell*, um die Einstellungen entsprechend zu konfigurieren.

Anmerkung: Die Nethwerk-Proxy-Einstellungen werden von allen Altova MissionKit-Applikationen gemeinsam verwendet. Wenn Sie daher die Einstellungen in einer Applikation ändern, wirkt sich dies automatisch auf alle anderen Applikationen aus.



System-Proxy-Einstellungen verwenden
Dadurch werden die über die System-Proxy-Einstellungen konfigurierbaren Internet Explorer (IE)-Einstellungen verwendet. Führt außerdem eine Abfrage der mit **`netsh.exe winhttp`** konfigurierten Einstellungen durch.

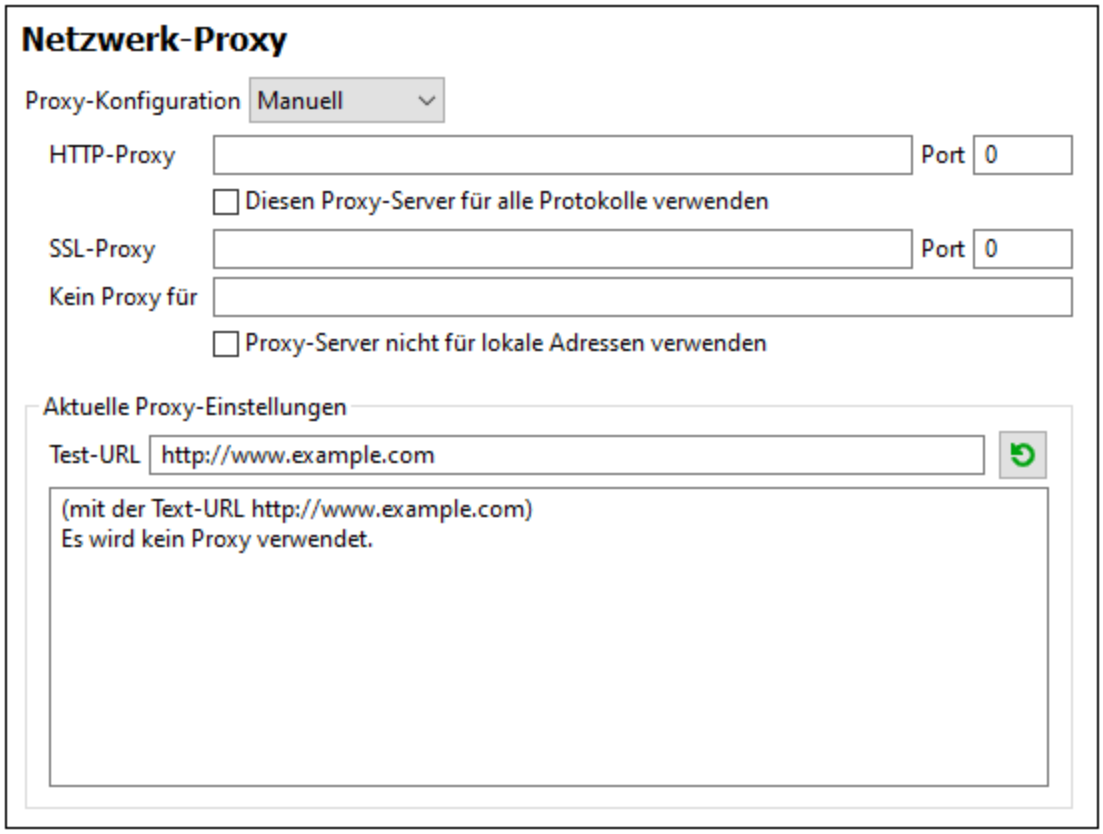
Automatische Proxy-Konfiguration
Es stehen die folgenden Optionen zur Verfügung:

- *Einstellungen automatisch ermitteln:* **`Verwendet ein WPAD-Skript`** **(**http://wpad.LOCALDOMAIN/wpad.dat) über DHCP oder DNS, um die Einrichtung des Proxy-Servers zu konfigurieren.
- *Skript-URL:* Definieren Sie eine HTTP URL zu einem automatischen Proxy-Konfigurationsskript (**`.pac`**), mit dem der Proxy-Server eingerichtet wird.
- *Neu laden:* Setzt die aktuelle automatische Proxy-Konfiguration zurück und lädt sie neu. Dafür ist

 © 2019-2025 Altova GmbH

Windows 8 oder neuer erforderlich. Die Rücksetzung kann bis zu 30 Sekunden dauern.

Manuelle Proxy-Konfiguration
Definieren Sie den vollständig qualifizierten Host-Namen und Port für die Proxy-Server der jeweiligen Protokolle manuell. Im Host-Namen kann ein unterstütztes Schema inkludiert werden (z.B.: **`http://hostname`**). Das Schema muss nicht mit dem entsprechenden Protokoll übereinstimmen, wenn der Proxy-Server das Schema unterstützt.



Es stehen die folgenden Optionen zur Verfügung:

- *HTTP-Proxy:* Verwendet den angegebenen Host-Namen und Port für das HTTP-Protokoll. Wenn *Diesen Proxy-Server für alle Protokolle verwenden* aktiviert ist, wird der angegebene HTTP-Proxy-Server für alle Protokolle verwendet.
- *SSL-Proxy:* Verwendet den angegebenen Host-Namen und Port für das SSL-Protokoll.
- *Kein Proxy für:* eine durch Semikola (**`;`**) getrennte Liste von voll qualifizierten Host-Namen, Domain-Namen oder IP-Adressen für Hosts, die ohne einen Proxy-Server verwendet werden sollen. IP-Adressen dürfen nicht abgeschnitten werden und IPv6-Adressen müssen innerhalb von eckige Klammern gesetzt werden (z.B.: **`[2606:2800:220:1:248:1893:25c8:1946]`**). Domain-Namen muss ein Punkt vorangestellt werden (z.B: **`.example.com`**).
- *Proxy-Server nicht für lokale Adressen verwenden:* Falls dieses Kontrollkästchen aktiviert ist, wird **`<local>`** zur *Kein Proxy für*-Liste hinzugefügt. Falls diese Option ausgewählt ist, wird für die folgenden Adressen kein Proxy-Server verwendet: (i) **`127.0.0.1`**, (ii) [::1], (iii) alle Host-Namen, die kein Punktzeichen (**`.`**) enthalten.

Anmerkung: Wenn ein Proxy-Server definiert wurde und Sie ein Mapping auf Altova FlowForce Server

bereitstellen möchten, müssen Sie die Option *Proxy-Server nicht für lokale Adressen verwenden* aktivieren

Aktuelle Proxy-Einstellungen
Stellt ein ausführliches Protokoll der Proxy-Ermittlung bereit. Es kann über die Schaltfläche **Aktualisieren** rechts vom Feld *Test-URL* aktualisiert werden (z.B. bei Wechsel zu einer anderen Test-URL oder wenn die Proxy-Einstellungen geändert wurden).

- *Test-URL:* Anhand einer Test-URL kann ermittelt werden, welcher Proxy-Server für diese bestimmte URL verwendet wird. Mit dieser URL erfolgt kein I/O. Dieses Feld darf nicht leer sein, wenn die automatische Proxy-Konfiguration verwendet wird (entweder über *System-Proxy-Einstellungen verwenden* oder *Automatische Proxy-Konfiguration*).

 © 2019-2025 Altova GmbH

11.10 Fenster

Dieses Kapitel enthält eine Liste aller Menübefehle im Menü **Fenster**.

- Überlappend

  Mit diesem Befehl werden alle offenen Dokumentenfenster so angeordnet, dass sie einander überlappend angezeigt werden.

- Horizontal anordnen

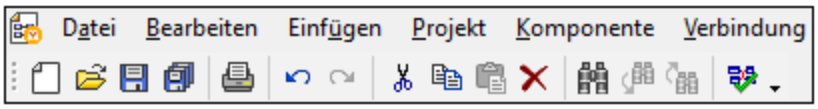
  Mit diesem Befehl ordnen Sie alle offenen Dokumentfenster horizontal nebeneinander an, sodass alle gleichzeitig sichtbar sind.

- Vertikal anordnen

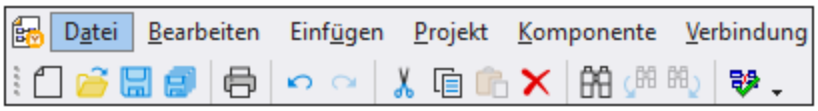
  Mit diesem Befehl werden alle offenen Dokumentenfenster vertikal übereinander angeordnet, sodass alle gleichzeitig sichtbar sind.

- Klassisches/Helles/Dunkles Design

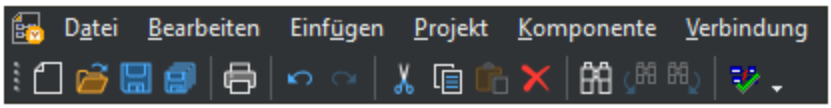
  In MapForce stehen die folgenden Designs zur Auswahl: *Klassisch*, *Hell* und *Dunkel*. Beispiele für diese Designs finden Sie in den Abbildungen weiter unten. Die Standardoption ist das klassische Design.



Klassisches Design



Helles Design



Dunkles Design

- 1 <MappingName>

  Bezieht sich auf das erste offene Mapping-Design. Wenn mehrere Mappings gleichzeitig offen sind, werden diese ebenfalls im Kontextmenü aufgelistet.

- Fenster

  In der Liste sehen Sie alle offenen Fenster und Sie können jederzeit zwischen diesen Fenstern wechseln. Um zwischen den Fenstern zu wechseln, können Sie auch die Tastaturkürzel **Strg+Tab** oder **Strg+F6** verwenden.

© 2019-2025 Altova GmbH

11.11 Hilfe

Dieses Kapitel enthält eine Liste aller Menübefehle im Menü **Hilfe**.

Hilfe (F1)

Mit dem Befehl **Hilfe (F1)** wird die Hilfe-Dokumentation (das Benutzerhandbuch) der Applikation geöffnet. Standardmäßig wird die Online-Hilfe im HTML-Format auf der Altova Website aufgerufen.

Falls Sie keinen Internet-Zugriff haben oder die Online-Hilfe aus einem anderen Grund nicht aufrufen möchten, können Sie die lokal gespeicherte Version des Benutzerhandbuchs verwenden. Dabei handelt es sich um eine PDF-Datei namens `MapForce.pdf`, die sich im Applikationsordner (im Ordner "Programme") befindet.

Im Abschnitt "Hilfe" des Dialogfelds "Optionen" (Menübefehl **Extras | Optionen**) können Sie das gewünschte Standardformat wechseln (Online-Hilfe oder lokale PDF-Datei).

Software-Aktivierung

Lizenzieren Ihres Produkts
Nachdem Sie Ihre Altova-Software heruntergeladen haben, können Sie sie entweder mit Hilfe eines kostenlosen Evaluierungs-Keycode oder eines käuflich erworbenen permanenten Lizenzkeycode lizenzieren oder aktivieren.

- ***Kostenlose Evaluierungs-Lizenz.*** Wenn Sie die Software zum ersten Mal starten, wird das Dialogfeld **Software-Aktivierung** angezeigt. Es enthält eine Schaltfläche, über die Sie eine kostenlose Evaluierungs-Lizenz anfordern können. Klicken Sie darauf, um Ihre Lizenz abzurufen. Wenn Sie auf diese Schaltfläche klicken, wird ein Hash Ihrer Rechner-ID erzeugt und über HTTPS an Altova gesendet. Die Lizenzinformationen werden per HTTP-Response an den Rechner zurückgesendet. Wenn die Lizenz erfolgreich erstellt wurde, wird in Ihrer Altova-Applikation ein entsprechendes Dialogfeld angezeigt. Wenn Sie in diesem Dialogfeld auf **OK** klicken, wird die Software für einen Zeitraum von 30 Tagen **auf diesem bestimmten Rechner aktiviert**.

- ***Permanenter Lizenz-Keycode.*** Über das Dialogfeld **Software-Aktivierung** können Sie einen permanenten Lizenz-Keycode erwerben. Wenn Sie auf diese Schaltfläche klicken, gelangen Sie zum Altova Online Shop, in dem Sie einen permanenten Lizenzschlüssel für Ihr Produkt erwerben können. Ihre Lizenz wird Ihnen in Form einer Lizenzdatei, die Ihre Lizenzdaten enthält, per E-Mail zugesendet.

  Es gibt drei Arten von permanenten Lizenzen: *Einzelplatzlizenzen*, *Parallellizenzen* und *Named User-Lizenzen* (benutzerdefinierte Nutzung). Mit einer Einzelplatzlizenz wird die Software auf einem einzigen Rechner freigeschaltet. Wenn Sie eine Einzelplatzlizenz für *N* Rechner erwerben, gestattet Ihnen die Lizenz, die Software auf bis zu *N* Rechnern zu verwenden. Mit einer Parallellizenz für *N* Parallelbenutzer dürfen *N* Benutzer die Software gleichzeitig ausführen. (Die Software darf auf 10N Rechnern installiert sein.) Mit einer Named User-Lizenz darf ein bestimmter Benutzer die Software auf bis zu 5 verschiedenen Rechnern verwenden. Um Ihre Software zu aktivieren, klicken Sie auf **Neue Lizenz hochladen** und geben Sie im daraufhin angezeigten Dialogfeld den Pfad zur Lizenzdatei ein und klicken Sie auf **OK**.

Anmerkung: Bei Mehrplatzlizenzen wird jeder Benutzer aufgefordert, seinen eigenen Namen einzugeben.

Ihre Lizenz-E-Mail und die verschiedenen Methoden, Ihr Altova-Produkt zu lizenzieren
Die Lizenz-E-Mail, die Sie von Altova erhalten, enthält Ihre Lizenzdatei im Anhang. Die Lizenzdatei hat die Dateierweiterung **`.altova_licenses`**.

Sie haben folgende Möglichkeiten, Ihr Altova-Produkt zu aktivieren:

- Speichern Sie die Lizenzdatei (**`.altova_licenses`**) in einem geeigneten Ordner, doppelklicken Sie auf die Lizenzdatei, geben Sie etwaige erforderliche Informationen in das Dialogfeld ein, das daraufhin angezeigt wird und beenden Sie den Vorgang durch Klicken auf **Lizenzschlüssel anwenden**.
- Speichern Sie die Lizenzdatei (**`.altova_licenses`**) in einem geeigneten Ordner. Wählen Sie in Ihrem Altova-Produkt den Menübefehl **Hilfe | Software-Aktivierung** und klicken Sie anschließend auf **Neue Lizenz hochladen.** Navigieren Sie zur Lizenzdatei oder geben Sie den Pfad dazu ein und klicken Sie auf **OK**.
- Speichern Sie die Lizenzdatei (**`.altova_licenses`**) in einem geeigneten Ordner und laden Sie diese von dort aus in den Lizenz-Pool Ihres Altova LicenseServer hoch. Sie können die Lizenz anschließend (i) entweder von Ihrem Altova-Produkt über das Dialogfeld "Software-Aktivierung" abrufen (*siehe unten*) oder (ii) dem Produkt die Lizenz von Altova LicenseServer aus zuweisen. *Nähere Informationen zur Lizenzierung über LicenseServer finden Sie weiter unten in diesem Kapitel.*

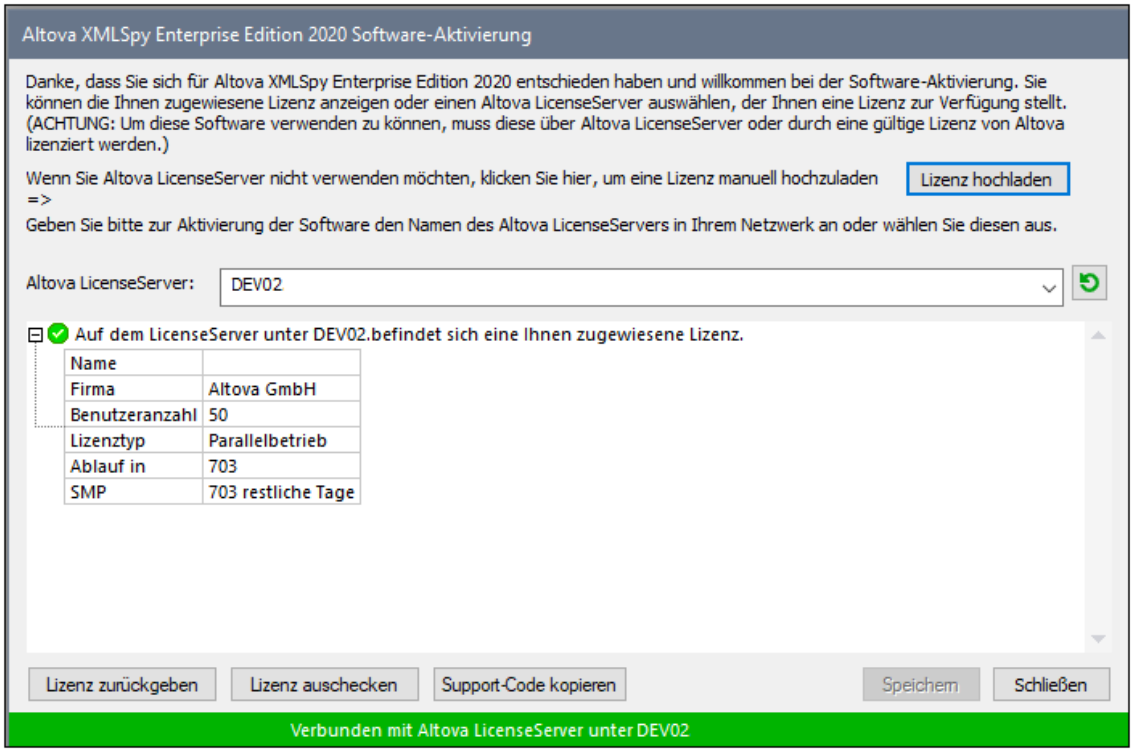
Das Dialogfeld **Software-Aktivierung** (*Abbildung unten*) kann über den Befehl **Hilfe | Software-Aktivierung** aufgerufen werden.

Aktivieren Ihrer Software
Sie können die Software durch Registrieren der Lizenz im Dialogfeld "Software-Aktivierung" oder durch Lizenzierung über Altova LicenseServer (*nähere Informationen siehe unten*) aktivieren.

- *Registrierung der Lizenz im Dialogfeld "Software-Aktivierung".* Klicken Sie im Dialogfeld auf **Neue Lizenz hochladen** und navigieren Sie zur Lizenzdatei. Klicken Sie auf **OK,** um den Pfad zur Lizenzdatei und alle eingegebenen Daten (im Fall einer Mehrplatzlizenz Ihren Namen) zu bestätigen und abschließend auf **Speichern**.

- *Lizenzierung über einen Altova LicenseServer in Ihrem Netzwerk:* Um eine Lizenz über einen Altova LicenseServer in Ihrem Netzwerk abzurufen, (klicken Sie am unteren Rand des Dialogfelds **Software-Aktivierung** auf **Altova LicenseServer verwenden**). Wählen Sie den Rechner aus, auf dem der gewünschte LicenseServer installiert wurde. Beachten Sie, dass die automatische Ermittlung von License Servern durch die Aussendung eines Signals ins LAN erfolgt. Da diese Aussendung auf ein Subnetz beschränkt ist, muss sich der LicenseServer im selben Subnetz wie der Client-Rechner befinden, damit die Ermittlung von License Servern funktioniert. Falls die automatische Ermittlung nicht funktioniert, geben Sie den Namen des Servers ein. Der Altova LicenseServer muss in seinem Lizenzpool eine Lizenz für Ihre Altova-Produkt haben. Wenn im LicenseServer-Pool eine Lizenz verfügbar ist, wird dies im Dialogfeld **Software-Aktivierung** angezeigt (*siehe Abbildung unten, in der Sie das Dialogfeld in Altova XMLSpy sehen*) und Sie können auf **Speichern** klicken, um die Lizenz abzurufen.

 © 2019-2025 Altova GmbH



Eine rechnerspezifische Lizenz (Einzelplatzlizenz) kann erst nach Ablauf von sieben Tagen wieder an LicenseServer zurückgegeben werden. Danach können Sie die rechnerspezifische Lizenz durch Klick auf **Lizenz zurückgeben** an den Server zurückgeben, sodass sie von einem anderen Client vom LicenseServer abgerufen werden kann. Ein LicenseServer-Administrator kann die Zuweisung einer abgerufenen Lizenz jedoch über die Web-Benutzeroberfläche von LicenseServer jederzeit aufheben. Beachten Sie, dass eine Rückgabe von Lizenzen nur bei rechnerspezifischen Lizenzen, nicht aber bei Parallellizenzen möglich ist.

Lizenz-Check-Out
Über den Lizenzpool können Sie eine Lizenz für einen Zeitraum von bis zu 30 Tagen auschecken, sodass die Lizenz auf dem lokalen Rechner gespeichert wird. Dadurch können Sie offline arbeiten, was nützlich ist, wenn Sie z.B. in einer Umgebung arbeiten möchten, in der Sie keinen Zugriff auf Ihren Altova LicenseServer haben (z.B. wenn Ihr Altova-Produkt auf einem Laptop installiert ist und Sie gerade unterwegs sind). Solange die Lizenz ausgecheckt ist, zeigt LicenseServer die Lizenz als in Verwendung an. Diese Lizenz kann dann von keinem anderen Rechner verwendet werden. Die Lizenz wird nach Ablauf des Check-Out-Zeitraums automatisch wieder eingecheckt. Alternativ dazu kann eine ausgecheckte Lizenz jederzeit über die Schaltfläche **Einchecken** des Dialogfelds **Software-Aktivierung** wieder eingecheckt werden.

Um eine Lizenz auszuchecken, gehen Sie folgendermaßen vor: (i) Klicken Sie im Dialogfeld **Software-Aktivierung** auf **Lizenz auschecken** (*siehe Abbildung oben*); (ii) Wählen Sie im daraufhin angezeigten Dialogfeld **Lizenz-Check-Out** den gewünschten Check-Out-Zeitraum aus und klicken Sie auf **Auschecken**. Daraufhin wird die Lizenz ausgecheckt. Nachdem Sie eine Lizenz ausgecheckt haben, geschehen zwei Dinge: (i) Die Check-Out-Informationen und das Ende des Check-Out-Zeitraums werden im Dialogfeld **Software-Aktivierung** angezeigt; (ii) Die Schaltfläche **Lizenz auschecken** im Dialogfeld ändert sich nun in **Einchecken**. Sie können die Lizenz jederzeit durch Klicken auf **Einchecken** einchecken. Da die Lizenz nach Ablauf des

Check-Out-Zeitraums automatisch wieder in den Zustand "Eingecheckt" zurück wechselt, sollte der von Ihnen ausgewählte Zeitraum für das Check-Out den gewünschten Zeitraum, in dem Sie offline arbeiten möchten, entsprechend abdecken.

Wenn es sich bei der ausgecheckten Lizenz um eine Einzelplatzlizenz oder Parallellizenz handelt, wird sie auf dem Rechner ausgecheckt und steht dem Benutzer, der die Lizenz ausgecheckt hat, zur Verfügung. Wenn es sich bei der Lizenz um eine Named User-Lizenz handelt, wird die Lizenz an das Windows-Konto des jeweiligen Benutzers (Named User) ausgecheckt. Lizenz Check-outs funktionieren auf einer virtuellen Maschine, nicht aber auf einem virtuellen Desktop (in einer VDI). Anmerkung:Wenn eine Named User-Lizenz ausgecheckt wird, werden die Daten zur Identifikation des Check-outs im Profil des Benutzers gespeichert. Damit Lizenz-Check-outs funktionieren, muss das Profil des Benutzers auf dem lokalen Rechner, der offline verwendet werden soll, gespeichert sein. Wenn das Profil des Benutzers nicht lokal (z.B. auf einem freigegebenen Laufwerk) gespeichert ist, wird der Check-out als ungültig gemeldet, sobald der Benutzer versucht, die Altova-Applikation zu verwenden.

Wenn eine Lizenz wieder eingecheckt wird, muss diese Lizenz für dieselbe Hauptversion eines Altova-Produkts ausgestellt sein, wie die Lizenz, die ausgecheckt wurde. Stellen Sie daher sicher, dass die Lizenz eingecheckt ist, bevor Sie für Ihr Altova-Produkt ein Upgrade auf die nächste Hauptversion installieren.

Anmerkung: Damit Lizenzen ausgecheckt werden können, muss die Check-Out-Funktion auf dem LicenseServer aktiviert werden. Wenn diese Funktion nicht aktiviert wurde, erhalten Sie eine entsprechende Fehlermeldung, wenn Sie versuchen die Lizenz auszuchecken. Wenden Sie sich in diesem Fall an Ihren LicenseServer-Administrator.

Support-Code kopieren
Klicken Sie auf **Support-Code kopieren,** um Lizenzinformationen in die Zwischenablage zu kopieren. Dies sind die Daten, die Sie bei einer Support-Anfrage über das Online Support-Formular benötigen.

Altova LicenseServer bietet IT-Administratoren einen Echtzeitüberblick über alle Altova-Lizenzen in einem Netzwerk. Dazu werden die Einzelheiten zu jeder Lizenz sowie Client-Zuweisungen und die Verwendung von Lizenzen durch Clients angezeigt. Der Vorteil der Verwendung von LicenseServer liegt in seinen Funktionen zur Verwaltung großer Altova-Lizenzpools. Altova LicenseServer steht kostenlos auf der Altova Website zur Verfügung. Nähere Informationen zu Altova LicenseServer und der Lizenzierung mittels Altova LicenseServer finden Sie in der Dokumentation zu Altova LicenseServer.

Bestellformular

Sobald Sie eine lizenzierte Version des Software-Produkts bestellen möchten, klicken Sie im Dialogfeld **Software-Aktivierung** (*siehe oben*) auf die Schaltfläche **Permanenten Key-Code erwerben...** oder wählen Sie den Befehl **Bestellformular**, um zum sicheren Online-Shop von Altova weitergeleitet zu werden.

Registrierung

Bei Aufruf dieses Befehls wird die Altova-Produktregistrierungsseite auf einem Register Ihres Browsers geöffnet. Durch Registrierung Ihrer Altova-Software stellen Sie sicher, dass Sie immer die neuesten Produktinformationen erhalten.

Auf Updates überprüfen

 © 2019-2025 Altova GmbH

Überprüft, ob am Altova Server eine neuere Version Ihres Produkts vorhanden ist und zeigt eine entsprechende Meldung an.

- Support Center

Der Befehl "Support Center" ist ein Link zum Altova Support Center im Internet. Im Support Center finden Sie Antworten auf häufig gestellt Fragen, Diskussionsforen, in denen Sie Software-Probleme besprechen können und ein Formular, um unsere Mitarbeiter vom technischen Support zu kontaktieren.

- Komponenten und Gratistools downloaden

Dieser Befehl ist ein Link zum Komponenten Download Center von Altova im Internet. Von hier können Sie Software-Komponenten verschiedener anderer Anbieter herunterladen, die Sie mit Altova Produkten verwenden können. Dabei handelt es sich um XSLT- und XSL-FO-Prozessoren, Applikationsserverplattformen usw. Die im Komponenten Download Center verfügbare Software ist normalerweise kostenlos.

- MapForce im Internet

Der Befehl MapForce im Internet ist ein Link zur Altova Website im Internet. Hier erfahren Sie mehr über MapForce und verwandte Technologien und Produkte auf der Altova Website.

- MapForce Training

Ein Link zur Online Training-Seite der Altova Website. Hier können Sie zwischen den von Altova-Experten gehaltenen Online-Kursen wählen.

- Über MapForce

Mit dem Befehl Über MapForce wird das Willkommensfenster und die Versionsnummer Ihres Produkts angezeigt. Wenn Sie die 64-Bit-Verson von MapForce verwenden, wird dies durch das Suffix (x64) nach dem Applikationsnamen angezeigt. Die 32-Bit-Version hat kein Suffix.

12 Anhänge

Diese Anhänge enthalten technische Informationen über MapForce, seine Aspekte und die Lizenzierung. Des Weiteren finden Sie darin eine Liste von Schlüsselbegriffen in MapForce und mit MapForce in Zusammenhang stehenden Produkten. Dieser Abschnitt ist in die folgenden Unterabschnitte gegliedert:

- Anmerkungen zur Unterstützung 525
- Prozessoren 527
- Technische Daten 633
- Lizenzinformationen 636

© 2019-2025 Altova GmbH

12.1 Anmerkungen zur Unterstützung

MapForce® ist eine 32/64-Bit-Windows-Applikation, die auf den folgenden Betriebssystemen läuft:

- Windows 10, Windows 11
- Windows Server 2016 oder höher

64-Bit-Unterstützung steht für die Enterprise und die Professional Edition zur Verfügung.

12.1.1 Unterstützte Quellen und Ziele

Wenn Sie die Transformationssprache eines MapForce-Mappings ändern, kann es vorkommen, dass bestimmte Funktionalitäten für diese spezifische Sprache nicht zur Verfügung stehen. Die folgende Tabelle enthält eine Übersicht über die Kompatibilität von Mapping-Formaten und Transformationssprachen in **MapForce Basic Edition**.

Anmerkungen:

- *Built-in* bedeutet, dass Sie das Mapping durch Klick auf das Fenster **Ausgabe** in MapForce oder mit MapForceServer ausführen können.

12.1.2 Unterstützte Funktionalitäten im generierten Code

In der folgenden Tabelle finden Sie eine Liste der Funktionalitäten, die für die Codegenerierung relevant sind und Informationen dazu, inwieweit diese in der **MapForce Basic Edition** in der jeweiligen Sprache unterstützt werden.

Funktion	XSLT 1.0	XSLT 2.0	XSLT 3.0
Bereitstellung von Parametern für das Mapping 164	●	●	●
Dynamische Bereitstellung der Input-Dateinamen über das Mapping 422	●	●	●
Dynamische Bereitstellung von Platzhalterdateinamen als Mapping-Input 422 1		●	●
Dynamische Generierung der Ausgabedateinamen anhand des Mappings 422		●	●
Rückgabe von String-Werten aus einem Mapping 171	●	●	●
Variablen 175		●	●
Sortierkomponenten 188		●	●
Gruppierungsfunktionen 297		●	●

Funktion	XSLT 1.0	XSLT 2.0	XSLT 3.0
Filter 194	●	●	●
Wertezuordnungskomponenten 200	●	●	●
Dynamische Node-Namen 405		●	●

Fußnoten:

1. Für XSLT 2.0, XSLT 3.0 und XQuery wird die Funktion `fn:collection` verwendet. In der Implementierung im Altova XSLT 2.0-, XSLT 3.0- und XQuery-Prozessor werden Platzhalter aufgelöst. Andere Prozessoren verhalten sich eventuell anders.

© 2019-2025 Altova GmbH

12.2 Informationen zu den Prozessoren

Dieser Abschnitt enthält Informationen über implementierungsspezifische Funktionen des Altova XML Validators, des Altova XSLT 1.0-Prozessors, des Altova XSLT 2.0-Prozessors und des Altova XQuery-Prozessors.

12.2.1 Informationen zum XSLT- und XQuery-Prozessor

Der XSLT- und der XQuery-Prozessor von MapForce hält sich genau an die W3C-Spezifikationen und ist daher strenger als die früheren Altova-Prozessoren, wie z.B. die in frühere Versionen von XMLSpy integrierten . Infolgedessen werden auch leichte Fehler, die von früheren Prozessoren ignoriert wurden, von MapForce als Fehler gekennzeichnet.

Zum Beispiel:

- Wenn das Ergebnis eines Pfad-Operators sowohl Nodes als auch Nicht-Nodes enthält, wird ein Typfehler (`err:XPTY0018`) ausgegeben.
- Wenn `E1` in einem Pfadausdruck `E1/E2` nicht zu einer Node-Sequenz ausgewertet wird, wird ein Typfehler (`err:XPTY0019`) ausgegeben.

Ändern Sie bei Auftreten eines solchen Fehlers je nach Bedarf, entweder das XSLT/XQuery-Dokument oder das Instanzdokument.

In diesem Abschnitt sind implementierungsspezifische Funktionalitäten der Prozessoren geordnet nach Spezifikation beschrieben:

- XSLT 1.0 527
- XSLT 2.0 528
- XQuery 1.0 529

12.2.1.1 XSLT 1.0

Der XSLT 1.0-Prozessor von MapForce entspricht der XSLT 1.0 Recommendation vom 16. November 1999 und der XPath 1.0 Recommendation vom 16. November 1999 des World Wide Web Consortium (W3C). Beachten Sie die folgenden Informationen zur Implementierung.

Anmerkungen zur Implementierung

Wenn das `method`-Attribut von `xsl:output` auf HTML gesetzt ist oder wenn standardmäßig die HTML-Ausgabe ausgewählt ist, werden Sonderzeichen in der XML- oder XSLT-Datei als HTML-Zeichenreferenzen in das HTML-Ausgabedokument eingefügt. So wird z.B. das Zeichen U+00A0 (die hexadezimale Zeichenreferenz für ein geschütztes Leerzeichen) entweder als Zeichenreferenz (` ` or ` `) oder als Entity-Referenz ` ` in den HTML-Code eingefügt.

12.2.1.2 XSLT 2.0

In diesem Abschnitt:

- Prozessorkonformität 528
- Rückwärtskompatibilität 528
- Namespaces 528
- Schema-Fähigkeit 529
- Implementierungsspezifisches Verhalten 529

Standardkonformität

Der XSLT 2.0-Prozessor von MapForce entspricht der XSLT 2.0 Recommendation vom 23. Jänner 2007 und der XPath 2.0 Recommendation vom 14. Dezember 2010 des World Wide Web Consortium (W3C).

Rückwärtskompatibilität

Der XSLT 2.0-Prozessor ist rückwärtskompatibel. Normalerweise kommt die Rückwärtskompatibilität des XSLT 2.0.-Prozessors nur dann zum Einsatz, wenn Sie den XSLT 2.0-Prozessor zur Verarbeitung eines XSLT 1.0 Stylesheets oder einer XSLT 1.0-Anweisung verwenden. Beachten Sie, dass sich das Ergebnis des XSLT 1.0-Prozessors und des rückwärtskompatiblen XSLT 2.0.Prozessors unter Umständen unterscheiden kann.

Namespaces

In Ihrem XSLT 2.0 Stylesheet sollten die folgenden Namespaces deklariert sein, damit Sie die in XSLT 2.0 verfügbaren Typ-Konstruktoren und Funktionen verwenden können. Normalerweise werden die unten aufgelisteten Präfixe verwendet; bei Bedarf können Sie auch andere Präfixe verwenden.

Namespace Name	Präfix	Namespace URI
XML Schema-Typen	`xs:`	`http://www.w3.org/2001/XMLSchema`
XPath 2.0-Funktionen	`fn:`	`http://www.w3.org/2005/xpath-functions`

Normalerweise werden diese Namespaces im Element `xsl:stylesheet` oder `xsl:transform` deklariert, wie unten gezeigt:

```
<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>
```

Beachten Sie die folgenden Punkte:

- Der XSLT 2.0-Prozessor verwendet als **Standard-Funktions-Namespace** den Namespace für XPath 2.0- und XQuery 1.0-Funktionen (siehe Tabelle oben). Sie können daher XPath 2.0- und XSLT 2.0-Funktionen in Ihrem Stylesheet ohne Präfix verwenden. Wenn Sie den Namespace für XPath 2.0-Funktionen in Ihrem Stylesheet mit einem Präfix deklarieren, können Sie zusätzlich dazu das in der Deklaration zugewiesene Präfix verwenden.

 © 2019-2025 Altova GmbH

- Bei Verwendung von Typ-Konstruktoren und Typen aus dem XML Schema-Namespace, muss bei Aufruf des Typ-Konstruktors (z.B. `xs:date`) das in der jeweiligen Namespace-Deklaration verwendeten Präfix verwendet werden.
- Einige XPath 2.0-Funktionen haben denselben Namen wie XML Schema-Datentypen. So gibt es z.B. für die XPath-Funktionen `fn:string` und `fn:boolean` XML-Schema-Datentypen mit demselben lokalen Namen: `xs:string` und `xs:boolean`. Wenn Sie daher den XPath-Ausdruck `string('Hello')` verwenden, wird der Ausdruck als `fn:string('Hello')` ausgewertet und nicht als `xs:string('Hello')`.

Schemafähigkeit

Der XSLT 2.0-Prozessor ist schemafähig. Sie können daher benutzerdefinierte Schematypen und die `xsl:validate`-Anweisung verwenden.

Implementierungsspezifisches Verhalten

Im Folgenden finden Sie eine Beschreibung, wie der XSLT 2.0-Prozessor implementierungsspezifische Aspekte von bestimmten XSLT 2.0-Funktionen behandelt.

`xsl:result-document`
Zusätzlich werden die folgenden Kodierungen unterstützt: `x-base16tobinary` und `x-base64tobinary`.

`function-available`
Die Funktion überprüft, ob in-scope-Funktionen (XSLT, XPath und Erweiterungsfunktionen) verfügbar sind.

`unparsed-text`
Das Attribut `href` akzeptiert (i) relative Pfade für Dateien im Basis-URI-Ordner und (ii) absolute Pfade mit oder ohne das `file://`-Protokoll. Zusätzlich werden die folgenden (Altova-spezifischen) Kodierungen unterstützt: `binarytobase16` und `binarytobase64`. Beispiel: `xs:base64Binary(unparsed-text('chart.png', 'x-binarytobase64'))`.

`unparsed-text-available`
Das Attribut `href` akzeptiert (i) relative Pfade für Dateien im Basis-URI-Ordner und (ii) absolute Pfade mit oder ohne das `file://`-Protokoll. Zusätzlich werden die folgenden (Altova-spezifischen) Kodierungen unterstützt: `binarytobase16` und `binarytobase64`.

Anmerkung: Die folgenden Kodierungswerte, die in früheren Versionen von AltovaXML, dem Vorgängerprodukt von RaptorXML, verwendet wurden, werden nun nicht mehr verwendet:`base16tobinary`, `base64tobinary`, `binarytobase16` und `binarytobase64`.

12.2.1.3 XQuery 1.0

In diesem Abschnitt:

- Prozessorkonformität 530
- Schema-Fähigkeit 530
- Kodierung 530
- Namespaces 528
- XML-Quelle und Validierung 531
- Statische und dynamische Typüberprüfung 531

- Bibliotheksmodule 531
- Externe Funktionen 532
- Collations 532
- Präzision von numerischen Daten 532
- Unterstützung für XQuery-Anweisungen 532
- Implementierungsspezifisches Verhalten 532

Standardkonformität

Der XQuery 1.0-Prozessor von MapForce entspricht der XQuery 1.0 Recommendation vom 14. Dezember 2010 des W3C. Der Query-Standard stellt bei vielen Funktionen frei, wie viele diese zu implementieren sind. Im Folgenden finden Sie eine Liste, wie der Altova XQuery 1.0-Prozessor diese Funktionen implementiert.

Schema-Fähigkeit

Der Altova XQuery 1.0-Prozessor ist **schemafähig.**

Kodierung

Die UTF-8 und die UTF-16 Zeichen-Kodierungen werden unterstützt.

Namespaces

Die folgenden Namespace-URIs und die damit verknüpften Bindings sind vordefiniert.

Namespace Name	Präfix	Namespace URI
XML Schema-Typen	`xs:`	`http://www.w3.org/2001/XMLSchema`
Schema-Instanz	`xsi:`	`http://www.w3.org/2001/XMLSchema-instance`
Vordefinierte Funktionen	`fn:`	`http://www.w3.org/2005/xpath-functions`
Lokale Funktionen	`local:`	`http://www.w3.org/2005/xquery-local-functions`

Beachten Sie die folgenden Punkte:

- Der Altova XQuery 1.0-Prozessor ist so konfiguriert, dass die oben aufgelisteten Präfixe an die entsprechenden Namespaces gebunden sind.
- Da der oben angeführte Namespace für vordefinierte Funktionen (siehe `fn:`) der Standard-Funktions-Namespace in XQuery ist, muss beim Aufruf von vordefinierten Funktionen das Präfix `fn:` nicht verwendet werden (`string("Hello")` ruft z.B. die Funktion `fn:string` auf). Das Präfix `fn:` kann jedoch verwendet werden, um eine vordefinierte Funktion aufzurufen, ohne die Namespace im Abfrage-Prolog deklarieren zu müssen (z.B.: `fn:string("Hello")`).
- Sie können den Standard-Funktions-Namespace durch Deklarierung des `default function namespace`-Ausdrucks im Abfrageprolog ändern.
- Bei Verwendung von Typen aus dem XML Schema-Namespace kann das Präfix `xs:` verwendet werden, ohne dass Sie den Namespace explizit deklarieren müssen und dieses Präfix im Abfrageprolog daran binden müssen. (Beispiel: `xs:date` und `xs:yearMonthDuration`.) Wenn Sie ein anderes Präfix für den XML-Schema-Namespace verwenden möchten, muss dieses im Abfrageprolog explizit deklariert werden. (Beispiel: `declare namespace alt = "http://www.w3.org/2001/XMLSchema"; alt:date("2004-10-04")`.)

 © 2019-2025 Altova GmbH

- Beachten Sie, dass die Datentypen `untypedAtomic`, `dayTimeDuration` und `yearMonthDuration` mit den Candidate Recommendations vom 23 January 2007 aus dem XPath Datentypen-Namespace in den XML-Schema Namespace verschoben wurden, d.h. `xs:yearMonthDuration`.

Wenn Namespaces für Funktionen, Typ-Konstruktoren, Node Tests usw. falsch zugewiesen wurden, wird ein Fehler ausgegeben. Beachten Sie jedoch, dass einige Funktionen denselben Namen wie Schema-Datentypen haben, z.B. `fn:string` und `fn:boolean`. (Sowohl `xs:string` als auch `xs:boolean` ist definiert.) Das Namespace-Präfix legt fest, ob die Funktion oder der Typ-Konstruktor verwendet wird.

XML-Quelldokument und Validierung

XML-Dokumente, die bei der Ausführung eines XQuery-Dokuments mit dem Altova XQuery 1.0-Prozessor verwendet werden, müssen wohlgeformt sein. Sie müssen jedoch nicht gemäß einem XML-Schema gültig sein. Wenn die Datei nicht gültig ist, wird die ungültige Datei ohne Schemainformationen geladen. Wenn die XML-Datei mit einem externen Schema verknüpft ist und gemäß diesem Schema gültig ist, werden für die XML-Daten nachträglich Validierungsinformationen generiert und für die Auswertung der Abfrage verwendet.

Statische und dynamische Typüberprüfung

In der statischen Analysephase werden Aspekte der Abfrage überprüft wie z.B. die Syntax, ob externe Referenzen (z.B. für Module) vorhanden sind, ob aufgerufene Funktionen und Variablen definiert sind, usw. Wenn in dieser Phase ein Fehler gefunden wird, wird eine Meldung ausgegeben und die Ausführung wird gestoppt.

Die dynamische Typ-Überprüfung wird zur Laufzeit durchgeführt, während die Abfrage ausgeführt wird. Wenn ein Typ mit den Anforderungen einer Operation nicht kompatibel ist, wird ein Fehler ausgegeben. So gibt z.B. der Ausdruck `xs:string("1") + 1` einen Fehler zurück, weil die Operation "Addition" nicht an einem Operanden vom Typ `xs:string` ausgeführt werden kann.

Bibliotheksmodule

Bibliotheksmodule dienen zum Speichern von Funktionen und Variablen, damit diese wiederverwendet werden können. Der Altova XQuery 1.0-Prozessor unterstützt Module, die in einer **einzigen externen XQuery-Datei** gespeichert sind. Eine solche Moduldatei muss im Prolog eine `module`-Deklaration enthalten, in der ein Target Namespace zugewiesen wird. Hier ein Beispielmodul:

```
module namespace libns="urn:module-library";
declare variable $libns:company := "Altova";
declare function libns:webaddress() { "http://www.altova.com" };
```

Alle im Modul deklarierten Funktionen und Variablen gehören zu dem mit dem Modul verknüpften Namespace. Das Modul wird durch Import in eine XQuery-Datei mittels der `import module`-Anweisung im Abfrageprolog verwendet. Die `import module`-Anweisung importiert nur Funktionen und Variablen, die direkt in der Bibliotheksmodul-Datei deklariert sind:

```
import module namespace modlib = "urn:module-library" at "modulefilename.xq";
if        ($modlib:company = "Altova")
then       modlib:webaddress()
else       error("No match found.")
```

External functions

Externe Funktionen, d.h. diejenigen Funktionen, die das Schlüsselwort `external` verwenden, werden nicht unterstützt:

```
declare function hoo($param as xs:integer) as xs:string external;
```

Collations

Die Standard-Collation ist die Unicode Codepoint Collation, die Strings auf Basis ihrer Unicode-Codepunkte vergleicht. Andere unterstützte Collations sind die hier 533 aufgelisteten ICU-Collations. Um eine bestimmte Collation zu verwenden, geben Sie ihre in der Liste der unterstützten Collations 533 angeführte URI an. String-Vergleiche, wie die Funktionen `fn:max` und `fn:min` werden anhand der angegebenen Collation durchgeführt. Wenn die Collation-Option nicht definiert ist, wird die Standard-Unicode Codepoint Collation verwendet.

Präzision von numerischen Typen

- Der Datentyp `xs:integer` hat eine beliebige Präzision, d.h. er kann beliebig viele Stellen haben.
- Der Datentyp `xs:decimal` kann nach dem Dezimalpunkt maximal 20 Stellen haben.
- Die Datentypen `xs:float` und `xs:double` sind auf 15 Stellen beschränkt.

Unterstützung für XQuery-Anweisungen

Die `Pragma`-Anweisung wird nicht unterstützt. Gegebenenfalls wird sie ignoriert und der Fallback-Ausdruck wird evaluiert.

Implementierungsspezifisches Verhalten

Im Folgenden wird beschrieben, wie der XQuery- und der XQuery Update 1.0-Prozessor implementierungsspezifische Aspekte bestimmter Funktionen behandeln.

`unparsed-text`

Das Attribut `href` akzeptiert (i) relative Pfade für Dateien im Basis-URI-Ordner und (ii) absolute Pfade mit oder ohne das `file://`-Protokoll. Zusätzlich werden die folgenden (Altova-spezifischen) Kodierungen unterstützt: `binarytobase16` und `binarytobase64`. Beispiel: `xs:base64Binary(unparsed-text('chart.png', 'x-binarytobase64'))`.

`unparsed-text-available`

Das Attribut `href` akzeptiert (i) relative Pfade für Dateien im Basis-URI-Ordner und (ii) absolute Pfade mit oder ohne das `file://`-Protokoll. Zusätzlich werden die folgenden (Altova-spezifischen) Kodierungen unterstützt: `binarytobase16` und `binarytobase64`.

Anmerkung: Die folgenden Kodierungswerte, die in früheren Versionen von AltovaXML, dem Vorgängerprodukt von RaptorXML, verwendet wurden, werden nun nicht mehr verwendet:`base16tobinary`, `base64tobinary`, `binarytobase16` und `binarytobase64`.

 © 2019-2025 Altova GmbH

12.2.2 XSLT- und XPath/XQuery-Funktionen

Dieser Abschnitt enthält eine Liste von Altova-Erweiterungsfunktionen und anderen Erweiterungsfunktionen, die in XPath und/oder XQuery-Ausdrücken verwendet werden können. ltova-Erweiterungsfunktionen können mit dem XSLT- und XQuery-Prozessor von Altova verwendet werden und bieten zusätzliche Funktionalitäten zu den in den W3C-Standards definierten Funktionsbibliotheken.

In diesem Abschnitt werden hauptsächlich XPath/XQuery-Erweiterungsfunktionen, die von Altova für zusätzliche Operationen sowie andere Erweiterungsfunktionen 614 erstellt wurden, beschrieben. Diese Erweiterungsfunktionen 534 können vom Altova-XSLT- und XQuery-Prozessor gemäß den in diesem Abschnitt beschriebenen Regeln verarbeitet werden. Informationen zu den regulären XPath/XQuery-Funktionen finden Sie in der Altova XPath/XQuery-Funktionsreferenz

Allgemeine Punkte

Beachten Sie bitte die folgenden allgemeinen Punkte:

- Funktionen aus den in den W3C-Spezifikationen definierten core-Funktionsbibliotheken können ohne Präfix aufgerufen werden, da der Altova-XSLT- und XQuery-Prozessor Funktionen, die kein Präfix haben, als Funktionen des in der XPath/XQuery Functions-Spezifikation Standard-Funktions-Namespace `http://www.w3.org/2005/xpath-functions` liest. Wenn dieser Namespace in einem XSLT- oder XQuery-Dokument explizit deklariert ist, kann das in der Namespace-Deklaration definierte Präfix optional auch in Funktionsnamen verwendet werden.
- Wenn bei einer Funktion eine Sequenz von einem Datenelement als Argument erwartet wird und eine Sequenz von mehr als einem Datenelement gesendet wird, wird ein Fehler zurückgegeben.
- Alle String-Vergleiche werden unter Verwendung der Unicode Codepoint Collation ausgeführt.
- Ergebnisse, bei denen es sich um QNames handelt, werden in der Form `[prefix:]localname` serialisiert.

Präzision von xs:decimal

Die Präzision bezieht sich auf die Anzahl der Stellen in einer Zahl. Laut Spezifikation sind mindestens 18 Stellen erforderlich. Bei Divisionen, bei denen ein Ergebnis vom Typ `xs:decimal` erzeugt wird, beträgt die Präzision 19 Kommastellen ohne Runden.

Implizite Zeitzone

Beim Vergleich zweier `date`, `time`, oder `dateTime`-Werte muss die Zeitzone der verglichenen Werte bekannt sein. Wenn die Zeitzone in einem solchen Wert nicht explizit angegeben ist, wird die implizite Zeitzone verwendet. Als implizite Zeitzone wird die der Systemuhr verwendet. Der Wert kann mit Hilfe der Funktion `implicit-timezone()` überprüft werden.

Collations

Die Standard-Collation ist die Unicode Codepoint Collation, die Strings auf Basis ihrer Unicode-Codepunkte vergleicht. Der Prozessor verwendet den Unicode Collation-Algorithmus. Andere unterstützte Collations sind die hier aufgelisteten ICU-Collations. Um eine bestimmte Collation zu verwenden, geben Sie Ihre URI an (siehe Tabelle unten). String-Vergleiche, wie die Funktionen `fn:max` und `fn:min` werden anhand der angegebenen Collation durchgeführt. Wenn die Collation-Option nicht definiert ist, wird die Standard-Unicode Codepoint Collation verwendet.

Sprache	URIs
da: Dänisch	da_DK
de: Deutsch	de_AT, de_BE, de_CH, de_DE, de_LI, de_LU
en: Englisch	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: Spanisch	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: Französisch	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: Italienisch	it_CH, it_IT
ja: Japanisch	ja_JP
nb: Norwegisch (Bokmal)	nb_NO
nl: Holländisch	nl_AW, nl_BE, nl_NL
nn: Norwegisch (Nynorsk)	nn_NO
pt: Portugiesisch	pt_AO, pt_BR, pt_GW, pt_MZ, pt_PT, pt_ST
ru: Russisch	ru_MD, ru_RU, ru_UA
sv: Schwedisch	sv_FI, sv_SE

Namespace-Achse
Die Namespace-Achse wird in XPath 2.0 nicht mehr verwendet, wird aber weiterhin unterstützt. Um Namespace-Informationen mit XPath 2.0-Mechanismen aufzurufen, verwenden Sie die Funktionen `in-scope-prefixes()`, `namespace-uri()` und `namespace-uri-for-prefix()`.

12.2.2.1 Altova-Erweiterungsfunktionen

Altova-Erweiterungsfunktionen können in XPath/XQuery-Ausdrücken verwendet werden. Dadurch stehen neben den Funktionen in der Standardbibliothek der XPath-, XQuery- und XSLT-Funktionen zusätzliche Funktionen zur Verfügung. Die Altova-Erweiterungsfunktionen befinden sich im **Altova-Erweiterungsfunktions-Namespace, `http://www.altova.com/xslt-extensions`** und sind in diesem Abschnitt mit dem Präfix **`altova:`**, das als an diesen Namespace gebunden angenommen wird, gekennzeichnet. Beachten Sie, dass manche Funktionen in zukünftigen Versionen Ihres Produkts eventuell nicht mehr unterstützt werden oder dass sich das Verhalten einzelner Funktionen ändern kann. Um zu sehen, welche Altova-Erweiterungsfunktionen unterstützt werden, lesen Sie bitte die Dokumentation zur jeweiligen Release.

 © 2019-2025 Altova GmbH

Die in der "XPath/XQuery Functions"-Spezifikation des W3C definierten Funktionen können (i) in einem XSLT-Kontext in XPath-Ausdrücken und (ii) in einem XQuery-Dokument in XQuery-Ausdrücken verwendet werden. In dieser Dokumentation sind die Funktionen, die im Zusammenhang mit XPath in XSLT verwendet werden können, mit einem **XP**-Symbol und Funktionen, die im Zusammenhang mit XQuery verwendet werden können, mit einem **XQ**-Symbol markiert; sie fungieren als XQuery-Funktionen. In den XSLT-Spezifikationen des W3C (nicht in den "XPath/XQuery Functions"-Spezifikationen) sind außerdem Funktionen definiert, die in XSLT-Dokumenten in XPath-Ausdrücken verwendet werden können. Diese Funktionen sind mit dem Symbol **XSLT** gekennzeichnet und werden als XSLT-Funktionen bezeichnet. In welcher XPath/XQuery- und XSLT-Version eine Funktion verwendet werden kann, wird in der Beschreibung der Funktion (*siehe Symbole unten)* angegeben. Funktionen aus der XPath/XQuery- und XSLT-Funktionsbibliothek werden ohne Präfix aufgelistet. Erweiterungsfunktionen aus anderen Bibliotheken wie z.B. Altova-Erweiterungsfunktionen werden mit einem Präfix angegeben.

XPath-Funktionen (in XPath-Ausdrücken in XSLT verwendet):	**XP1** **XP2** **XP3.1**
XSLT-Funktionen (in XPath-Ausdrücken in XSLT verwendet):	**XSLT1** **XSLT2** **XSLT3**
XQuery-Funktionen (in XQuery-Ausdrücken in XQuery verwendet):	**XQ1** **XQ3.1**

Verwendung von Altova-Erweiterungsfunktionen

Um Altova-Erweiterungsfunktionen verwenden zu können, müssen Sie den Altova-Erweiterungsfunktions-Namespace deklarieren (*erster markierter Bereich im Codefragment unten*) und die Erweiterungsfunktionen anschließend so verwenden, dass sie als zu diesem Namespace gehörig aufgelöst werden (*siehe zweiter markierter Bereich*). Im Beispiel unten wird die Altova-Erweiterungsfunktion **age** verwendet.

```
<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>
```

XSLT-Funktionen [536]

XSLT-Funktionen können in XPath-Ausdrücken nur im XSLT-Kontext verwendet werden (ähnlich wie die XSLT 2.0-Funktionen `current-group()` oder `key()`). Diese Funktionen sind nicht für Nicht-XSLT-Kontext gedacht und funktionieren in einem solchen Kontext (z.B. in einem XQuery-Kontext) nicht. Beachten Sie, dass XSLT-Funktionen für XBRL nur mit Altova Produkten verwendet werden können, die XBRL unterstützen.

XPath/XQuery-Funktionen

XPath/XQuery-Funktionen können sowohl in XPath-Ausdrücken im XSLT-Kontext als auch in XQuery-Ausdrücken verwendet werden.

- Datum/Uhrzeit 539
- Standort 557
- Bildbezogene 569
- Numerisch 574
- Schema 576
- Sequenz 596
- String 605
- Verschiedenes 611

12.2.2.1.1 XSLT-Funktionen

XSLT-Erweiterungsfunktionen können in XPath-Ausdrücken in einem XSLT-Kontext verwendet werden. In einem Nicht-XSLT-Kontext (z.B. in einem XQuery-Kontext) funktionieren sie nicht.

Anmerkung zur Benennung von Funktionen und zur Anwendbarkeit der Sprache

Altova-Erweiterungsfunktionen können in XPath/XQuery-Ausdrücken verwendet werden. Dadurch stehen neben den Funktionen in der Standardbibliothek der XPath-, XQuery- und XSLT-Funktionen zusätzliche Funktionen zur Verfügung. Die Altova-Erweiterungsfunktionen befinden sich im **Altova-Erweiterungsfunktions-Namespace, `http://www.altova.com/xslt-extensions`** und sind in diesem Abschnitt mit dem Präfix **`altova:`**, das als an diesen Namespace gebunden angenommen wird, gekennzeichnet. Beachten Sie, dass manche Funktionen in zukünftigen Versionen Ihres Produkts eventuell nicht mehr unterstützt werden oder dass sich das Verhalten einzelner Funktionen ändern kann. Um zu sehen, welche Altova-Erweiterungsfunktionen unterstützt werden, lesen Sie bitte die Dokumentation zur jeweiligen Release.

XPath-Funktionen (in XPath-Ausdrücken in XSLT verwendet):	XP1 XP2 XP3.1
XSLT-Funktionen (in XPath-Ausdrücken in XSLT verwendet):	XSLT1 XSLT2 XSLT3
XQuery-Funktionen (in XQuery-Ausdrücken in XQuery verwendet):	XQ1 XQ3.1

Allgemeine Funktionen

distinct-nodes [altova:]

`altova:distinct-nodes(`*`node()*`*`) als node()*` XSLT1 XSLT2 XSLT3

Erhält eine Gruppe von einem oder mehreren Nodes als Input und gibt dieselbe Gruppe ohne Nodes mit doppelt vorhandenen Werten zurück. Der Vergleich wird mittels der XPath/XQuery-Funktion `fn:deep-equal` durchgeführt.

Beispiele

- **`altova:distinct-nodes`**`(country)` gibt alle Child `country` Nodes ohne diejenigen mit doppelt vorhandenen Werten zurück.

 © 2019-2025 Altova GmbH

evaluate [altova:]

altova:evaluate(XPathExpression *as xs:string***[, ValueOf$p1, ... ValueOf$pN])** XSLT1 XSLT2 XSLT3

Erhält einen XPath-Ausdruck als obligatorisches Argument, der als String übergeben wird, und gibt das Resultat des ausgewerteten Ausdrucks zurück. Beispiel: **altova:evaluate('//Name[1]')** gibt den Inhalt des ersten `Name` Elements im Dokument zurück. Beachten Sie, dass der Ausdruck `//Name[1]` durch Einschließen in einfache Anführungszeichen als String übergeben wird.

Die Funktion `altova:evaluate` kann zusätzliche (optionale) Argumente erhalten. Diese Argumente sind die Werte der einzelnen im Geltungsbereich befindlichen Variablen und haben die Namen `p1`, `p2`, `p3`... `pN`. Beachten Sie zur Verwendung die folgenden Punkte: (i) Die Variablennamen müssen die Form `pX` haben, wobei `X` eine Ganzzahl ist; (ii) die Argumente der Funktion `altova:evaluate` (*siehe Signatur oben*) liefern vom zweiten Argument an die Werte der Variablen, wobei die Reihenfolge der Argumente der numerisch geordneten Variablensequenz entspricht: `p1` bis `pN`. Das zweite Argument wird der Wert der Variablen `p1`, das dritte Argument der der Variablen `p2`, usw.; (iii) Die Werte der Variablen müssen vom Typ `item*` sein

Beispiel

```
<xsl:variable name="xpath" select="'$p3, $p2, $p1'" />
<xsl:value-of select="altova:evaluate($xpath, 10, 20, 'hi')" />
```

gibt aus "hi 20 10"

Beachten Sie im obigen Beispiel folgende Punkte:

- Das zweite Argument des Ausdrucks `altova:evaluate` ist der der Variablen `$p1` zugewiesene Wert, das dritte Argument ist das der Variablen `$p2` zugewiesene usw.
- Beachten Sie, dass das vierte Argument der Funktion ein String-Wert ist. Als String-Wert wird dieser innerhalb von Anführungszeichen gesetzt.
- Das `select` Attribut des Elements `xs:variable` liefert den XPath-Ausdruck. Da dieser Ausdruck den Typ `xs:string`, haben muss, wird er in einfache Anführungszeichen gesetzt.

Weitere Beispiele für die Verwendung der Variablen

- ```
  <xsl:variable name="xpath" select="'$p1'" />
  <xsl:value-of select="altova:evaluate($xpath, //Name[1])" />
  ```
  Gibt den Wert des ersten `Name` *Elements zurück.*

- ```
  <xsl:variable name="xpath" select="'$p1'" />
  <xsl:value-of select="altova:evaluate($xpath, '//Name[1]')" />
  ```
  Gibt "`//Name[1]`*" aus*

Die `altova:evaluate()` Erweiterungsfunktion ist in Situationen nützlich, in denen ein XPath-Ausdruck im XSLT-Stylesheet einen oder mehrere Teile enthält, die dynamisch ausgewertet werden müssen. Angenommen ein Benutzer gibt seinen Request für das Sortierkriterium ein und das Sortierkriterium ist im Attribut `UserReq/@sortkey`, gespeichert. Im Stylesheet könnten Sie den folgenden Ausdruck haben: `<xsl:sort select="`**altova:evaluate(../UserReq/@sortkey)**`" order="ascending"/>` Die `altova:evaluate()` Funktion liest das `sortkey` Attribut des `UserReq` Child-Elements des Parent des Kontext-Node. Angenommen der Wert des `sortkey` Attributs ist `Price`, dann wird von der `altova:evaluate()` Funktion `Price` zurückgegeben und wird zum Wert des `select` Attributs:

`<xsl:sort select="Price" order="ascending"/>`. Wenn diese `sort` Anweisung im Kontext eines Elements namens `Order` vorkommt, dann werden die `Order` Elemente nach den Werten Ihrer `Price` Children sortiert. Alternativ dazu, wenn der Wert von `@sortkey` z.B. `Date` ist, werden die `Order` Elemente nach den Werten ihrer `Date` Children sortiert. Das Sortierkriterium für `Order` wird also zur Laufzeit aus dem `sortkey` Attribut ausgewählt. Diese hätte man mit einem Ausdruck wie dem folgenden nicht bewerkstelligen können: `<xsl:sort select="../UserReq/@sortkey" order="ascending"/>`. Im oben gezeigten Beispiel wäre das Sortierkriterium das `sortkey` Attribut selbst, nicht `Price` oder `Date` (oder jeder beliebige andere Inhalt von `sortkey`)

Hinweis:
Der statische Kontext enthält Namespaces, Typen und Funktionen - aber keine Variablen - aus der aufrufenden Umgebung. Die Basis-URI und der Standard-Namespace werden vererbt.

Weitere Beispiele

- Statische Variablen: `<xsl:value-of select="$i3, $i2, $i1" />`
  Gibt die Werte von drei Variablen aus.

- Dynamischer XPath-Ausdruck mit dynamischen Variablen:
  ```
  <xsl:variable name="xpath" select="'$p3, $p2, $p1'" />
  <xsl:value-of select="altova:evaluate($xpath, 10, 20, 30)" />
  ```
  Gibt "30 20 10" aus

- Dynamischer XPath-Ausdruck ohne dynamische Variable:
  ```
  <xsl:variable name="xpath" select="'$p3, $p2, $p1'" />
  <xsl:value-of select="altova:evaluate($xpath)" />
  ```
  Gibt einen Fehler aus.: Es wurde keine Variable für `$p3` definiert.

encode-for-rtf [altova:]

`altova:encode-for-rtf(input` *`als xs:string`*`, preserveallwhitespace` *`als xs:boolean`*`, preservenewlines` *`als xs:boolean`*`) als xs:string` XSLT2 XSLT3
Konvertiert den Input-String in Code für RTF. Whitespaces und neue Zeilen werden gemäß dem für die entsprechenden Parameter definierten Booleschen Wert beibehalten.

[Nach oben 536]

XBRL-Funktionen

Altova XBRL-Funktionen können nur mit Editionen von Altova-Produkten verwendet werden, die XBRL unterstützen.

xbrl-footnotes [altova:]

`altova:xbrl-footnotes(`*`node()`*`)` als **`node()*`** XSLT2 XSLT3
Erhält einen Node als Input-Argument und gibt die durch den Input-Node referenzierte Gruppe der XBRL-Fußnoten-Nodes zurück.

 © 2019-2025 Altova GmbH

- xbrl-labels [altova:]

  altova:xbrl-labels(*xs:QName, xs:string***)** als **node()*** XSLT2 XSLT3
  Erhält zwei Input-Argumente: einen Node-Namen und den Pfad der Taxonomiedatei, die den Node enthält. Die Funktion gibt die XBRL Labels zurück, die mit dem Input-Node verknüpft sind.

[Nach oben 536]

12.2.2.1.2 XPath/XQuery-Funktionen: Datum und Uhrzeit

Die Datums- und Uhrzeit-Erweiterungsfunktionen von Altova können im Zusammenhang mit XPath- und XQuery-Ausdrücken verwendet werden und bieten zusätzliche Funktionalitäten für die Verarbeitung von Daten, die in Form von XML-Schema-Datums- und Uhrzeit-Datentypen zur Verfügung stehen. Die Funktionen in diesem Abschnitt können mit dem **XPath 3.0** - und dem **XQuery 3.0** -Prozessor von Altova verwendet werden. Sie stehen in verschiedenen XPath/XQuery-Kontexten zur Verfügung.

Anmerkung zur Benennung von Funktionen und zur Anwendbarkeit der Sprache

Altova-Erweiterungsfunktionen können in XPath/XQuery-Ausdrücken verwendet werden. Dadurch stehen neben den Funktionen in der Standardbibliothek der XPath-, XQuery- und XSLT-Funktionen zusätzliche Funktionen zur Verfügung. Die Altova-Erweiterungsfunktionen befinden sich im **Altova-Erweiterungsfunktions-Namespace, http://www.altova.com/xslt-extensions** und sind in diesem Abschnitt mit dem Präfix **altova:**, das als an diesen Namespace gebunden angenommen wird, gekennzeichnet. Beachten Sie, dass manche Funktionen in zukünftigen Versionen Ihres Produkts eventuell nicht mehr unterstützt werden oder dass sich das Verhalten einzelner Funktionen ändern kann. Um zu sehen, welche Altova-Erweiterungsfunktionen unterstützt werden, lesen Sie bitte die Dokumentation zur jeweiligen Release.

XPath-Funktionen (in XPath-Ausdrücken in XSLT verwendet):	XP1 XP2 XP3.1
XSLT-Funktionen (in XPath-Ausdrücken in XSLT verwendet):	XSLT1 XSLT2 XSLT3
XQuery-Funktionen (in XQuery-Ausdrücken in XQuery verwendet):	XQ1 XQ3.1

- Nach Funktionalität gruppiert

- Hinzufügen einer Zeitdauer zu xs:dateTime und Rückgabe von xs:dateTime 540
- Hinzufügen einer Zeitdauer zu xs:date und Rückgabe von xs:date 542
- Hinzufügen einer Zeitdauer zu xs:time und Rückgabe von xs:time 544
- Formatieren und Abrufen einer Zeitdauer 543
- Entfernen der Zeitzone aus Funktionen, die das aktuelle Datum/die aktuelle Uhrzeit generieren 545
- Rückgabe von Tagen, Stunden, Minuten und Sekunden anhand einer Zeitdauer 546
- Rückgabe des Wochentags anhand des Datums als Ganzzahl 548
- Rückgabe eines Wochentags als Ganzzahl anhand eines Datums 548
- Erstellen des Datums, der Uhrzeit oder des Zeitdauertyps anhand der lexikalischen Komponenten der einzelnen Typen 551
- Konstruieren des Typs "Datum", "Datum und Uhrzeit" oder "Uhrzeit" anhand eines String Input 552
- Funktionen zur Berechnung des Alters 554
- Epochen-Zeit (Unix-Zeit)-Funktionen 555

▼ in alphabetischer Reihenfolge

altova:add-days-to-date 542
altova:add-days-to-dateTime 540
altova:add-hours-to-dateTime 540
altova:add-hours-to-time 544
altova:add-minutes-to-dateTime 540
altova:add-minutes-to-time 544
altova:add-months-to-date 542
altova:add-months-to-dateTime 540
altova:add-seconds-to-dateTime 540
altova:add-seconds-to-time 544
altova:add-years-to-date 542
altova:add-years-to-dateTime 540
altova:age 554
altova:age-details 554
altova:build-date 551
altova:build-duration 551
altova:build-time 551
altova:current-dateTime-no-TZ 545
altova:current-date-no-TZ 545
altova:current-time-no-TZ 545
altova:date-no-TZ 545
altova:dateTime-from-epoch 555
altova:dateTime-from-epoch-no-TZ 555
altova:dateTime-no-TZ 545
altova:days-in-month 546
altova:epoch-from-dateTime 555
altova:hours-from-dateTimeDuration-accumulated 546
altova:minutes-from-dateTimeDuration-accumulated 546
altova:seconds-from-dateTimeDuration-accumulated 546
altova:format-duration 543
altova:parse-date 552
altova:parse-dateTime 552
altova:parse-duration 543
altova:parse-time 552
altova:time-no-TZ 545
altova:weekday-from-date 548
altova:weekday-from-dateTime 548
altova:weeknumber-from-date 549
altova:weeknumber-from-dateTime 549

[**Nach oben** 539]

Hinzufügen einer Zeitdauer zu xs:dateTime XP3.1 XQ3.1

Mit diesen Funktionen werden Zeitdauerwerte zu **`xs:dateTime`** hinzugefügt, bevor **`xs:dateTime`** zurückgegeben wird. Der Typ `xs:dateTime` hat das Format `JJJJ-MM-TTZhh:mm:ss.sss`. Es handelt sich hierbei um eine Verkettung des `xs:date` und `xs:time` Formats, getrennt durch den Buchstaben `Z`. Ein Zeitzonensuffix (wie z.B. **`+01:00`**) ist optional.

▼ add-years-to-dateTime [altova:]

`altova:add-years-to-dateTime(DateTime` *`als xs:dateTime`*, **`Years`** *`als xs:integer`*) **`als`**

 © 2019-2025 Altova GmbH

xs:dateTime XP3.1 XQ3.1
Fügt eine Zeitdauer in Jahren zu einem `xs:dateTime` Wert (*siehe Beispiele unten*) hinzu. Beim zweiten Argument handelt es sich um die Anzahl der Jahre, die zu dem im ersten Parameter angegebenen `xs:dateTime` Wert hinzugefügt werden sollen. Das Ergebnis ist vom Typ `xs:dateTime`.

Beispiele

- **altova:add-years-to-dateTime**(xs:dateTime("2014-01-15T14:00:00"), 10) gibt 2024-01-15T14:00:00 zurück
- **altova:add-years-to-dateTime**(xs:dateTime("2014-01-15T14:00:00"), -4) gibt 2010-01-15T14:00:00 zurück

add-months-to-dateTime [altova:]

altova:add-months-to-dateTime(DateTime *als xs:dateTime*, **Months** *als xs:integer*) als **xs:dateTime** XP3.1 XQ3.1
Fügt eine Zeitdauer in Monaten zu einem `xs:dateTime` Wert (*siehe Beispiele unten*) hinzu. Beim zweiten Argument handelt es sich um die Anzahl der Monate, die zu dem im ersten Argument angegebenen `xs:dateTime` Wert hinzugefügt werden sollen. Das Ergebnis ist vom Typ `xs:dateTime`.

Beispiele

- **altova:add-months-to-dateTime**(xs:dateTime("2014-01-15T14:00:00"), 10) gibt 2014-11-15T14:00:00 zurück
- **altova:add-months-to-dateTime**(xs:dateTime("2014-01-15T14:00:00"), -2) gibt 2013-11-15T14:00:00 zurück

add-days-to-dateTime [altova:]

altova:add-days-to-dateTime(DateTime *als xs:dateTime*, **Days** *als xs:integer*) als **xs:dateTime** XP3.1 XQ3.1
Fügt eine Zeitdauer in Tagen zu einem `xs:dateTime` Wert (*siehe Beispiel unten*) hinzu. Beim zweiten Argument handelt es sich um die Anzahl der Tage, die zu dem im ersten Argument angegebenen `xs:dateTime` Wert hinzugefügt werden sollen. Das Ergebnis ist vom Typ `xs:dateTime`.

Beispiele

- **altova:add-days-to-dateTime**(xs:dateTime("2014-01-15T14:00:00"), 10) gibt 2014-01-25T14:00:00 zurück
- **altova:add-days-to-dateTime**(xs:dateTime("2014-01-15T14:00:00"), -8) gibt 2014-01-07T14:00:00 zurück

add-hours-to-dateTime [altova:]

altova:add-hours-to-dateTime(DateTime *als xs:dateTime*, **Hours** *als xs:integer*) als **xs:dateTime** XP3.1 XQ3.1
Fügt eine Zeitdauer in Stunden zu einem `xs:dateTime` Wert (*siehe Beispiel unten*) hinzu. Beim zweiten Argument handelt es sich um die Anzahl der Stunden, die zu dem im ersten Argument angegebenen `xs:dateTime` Wert hinzugefügt werden sollen. Das Ergebnis ist vom Typ `xs:dateTime`.

Beispiele

- **altova:add-hours-to-dateTime**(xs:dateTime("2014-01-15T13:00:00"), 10) gibt 2014-01-

15T23:00:00 zurück
- **altova:add-hours-to-dateTime**(xs:dateTime("2014-01-15T13:00:00"), -8) gibt 2014-01-15T05:00:00 zurück

add-minutes-to-dateTime [altova:]

altova:add-minutes-to-dateTime(DateTime *als xs:dateTime*, **Minutes** *als xs:integer*) als **xs:dateTime** XP3.1 XQ3.1
Fügt eine Zeitdauer in Minuten zu einem xs:dateTime Wert (*siehe Beispiele unten)* hinzu. Beim zweiten Argument handelt es sich um die Anzahl der Minuten, die zu dem im ersten Argument angegebenen xs:dateTime Wert hinzugefügt werden sollen. Das Ergebnis ist vom Typ xs:dateTime.

Beispiele

- **altova:add-minutes-to-dateTime**(xs:dateTime("2014-01-15T14:10:00"), 45) gibt 2014-01-15T14:55:00 zurück
- **altova:add-minutes-to-dateTime**(xs:dateTime("2014-01-15T14:10:00"), -5) gibt 2014-01-15T14:05:00 zurück

add-seconds-to-dateTime [altova:]

altova:add-seconds-to-dateTime(DateTime *als xs:dateTime*, **Seconds** *als xs:integer*) als **xs:dateTime** XP3.1 XQ3.1
Fügt eine Zeitdauer in Sekunden zu einem xs:dateTime Wert (*siehe Beispiele unten*) hinzu. Beim zweiten Argument handelt es sich um die Anzahl der Sekunden, die zu dem im ersten Argument angegebenen xs:dateTime Wert hinzugefügt werden sollen. Das Ergebnis ist vom Typ xs:dateTime.

Beispiele

- **altova:add-seconds-to-dateTime**(xs:dateTime("2014-01-15T14:00:10"), 20) gibt 2014-01-15T14:00:30 zurück
- **altova:add-seconds-to-dateTime**(xs:dateTime("2014-01-15T14:00:10"), -5) gibt 2014-01-15T14:00:05 zurück

[Nach oben 539]

Hinzufügen einer Zeitdauer zu xs:date XP3.1 XQ3.1

Mit diesen Funktionen werden Zeitdauerwerte zu **xs:date** hinzugefügt, bevor **xs:date** zurückgegeben wird. Der Typ xs:date hat das Format JJJJ-MM-TT.

add-years-to-date [altova:]

altova:add-years-to-date(Date *als xs:date*, **Years** *als xs:integer*) als **xs:date** XP3.1 XQ3.1
Fügt eine Zeitdauer in Jahren zu einem Datumswert hinzu. Beim zweiten Argument handelt es sich um die Anzahl der Jahre, die zu dem im ersten Argument angegebenen xs:date Wert hinzugefügt werden sollen. Das Ergebnis ist vom Typ xs:date.

Beispiele

- **altova:add-years-to-date**(xs:date("2014-01-15"), 10) gibt 2024-01-15 zurück

 © 2019-2025 Altova GmbH

- **altova:add-years-to-date**(xs:date("2014-01-15"), -4) gibt 2010-01-15 zurück

add-months-to-date [altova:]

altova:add-months-to-date(Date *als xs:date*, **Months** *als xs:integer*) als **xs:date** XP3.1 XQ3.1
Fügt eine Zeitdauer in Monaten zu einem Datumswert hinzu. Beim zweiten Argument handelt es sich um die Anzahl der Monate, die zu dem im ersten Argument angegebenen xs:date Wert hinzugefügt werden sollen. Das Ergebnis ist vom Typ xs:date.

Beispiele

- **altova:add-months-to-date**(xs:date("2014-01-15"), 10) gibt 2014-11-15 zurück
- **altova:add-months-to-date**(xs:date("2014-01-15"), -2) gibt 2013-11-15 zurück

add-days-to-date [altova:]

altova:add-days-to-date(Date *als xs:date*, **Days** *als xs:integer*) als **xs:date** XP3.1 XQ3.1
Fügt eine Zeitdauer in Tagen zu einem Datumswert hinzu. Beim zweiten Argument handelt es sich um die Anzahl der Tage, die zu dem im ersten Argument angegebenen xs:date Wert hinzugefügt werden sollen. Das Ergebnis ist vom Typ xs:date.

Beispiele

- **altova:add-days-to-date**(xs:date("2014-01-15"), 10) gibt 2014-01-25 zurück
- **altova:add-days-to-date**(xs:date("2014-01-15"), -8) gibt 2014-01-07 zurück

[Nach oben 539]

Formatieren und Abrufen einer Zeitdauer XP3.1 XQ3.1

Mit diesen Funktionen wird ein Input **xs:duration**- oder **xs:string**-Wert geparst und ein **xs:string**- bzw. **xs:duration**-Wert zurückgegeben.

format-duration [altova:]

altova:format-duration(Duration *als xs:duration*, **Picture** *als xs:string*) als **xs:string** XP3.1 XQ3.1
Formatiert eine Zeitdauer, die als erstes Argument bereitgestellt wird, gemäß einem Muster-String, der als zweites Argument bereitgestellt wird. Die Ausgabe ist ein Textstring, der dem Muster-String entsprechend formatiert ist.

Beispiele

- **altova:format-duration**(xs:duration("P2DT2H53M11.7S"), "Days:[D01] Hours:[H01] Minutes:[m01] Seconds:[s01] Fractions:[f0]") gibt "Days:02 Hours:02 Minutes:53 Seconds:11 Fractions:7" gibt
- **altova:format-duration**(xs:duration("P3M2DT2H53M11.7S"), "Months:[M01] Days:[D01] Hours:[H01] Minutes:[m01]") gibt "Months:03 Days:02 Hours:02 Minutes:53" gibt

parse-duration [altova:]

altova:parse-duration(InputString *als xs:string*, **Picture** *als xs:string*) als **xs:duration** XP3.1 XQ3.1
Erhält einen Pattern-String als erstes Argument und eine Muster-String als zweites Argument. Der Input-Strings wird auf Basis des Muster-Strings geparst und ein `xs:duration` wird zurückgegeben.

Beispiele

- **altova:parse-duration**("Days:02 Hours:02 Minutes:53 Seconds:11 Fractions:7"), "Days:[D01] Hours:[H01] Minutes:[m01] Seconds:[s01] Fractions:[f0]") gibt "P2DT2H53M11.7S" zurück
- **altova:parse-duration**("Months:03 Days:02 Hours:02 Minutes:53 Seconds:11 Fractions:7", "Months:[M01] Days:[D01] Hours:[H01] Minutes:[m01]") gibt "P3M2DT2H53M" zurück

[Nach oben 539]

Hinzufügen einer Zeitdauer zu xs:time XP3.1 XQ3.1

Diese Funktionen fügen einen Zeitdauerwert zu **xs:time** hinzu und geben **xs:time** zurück. Der Typ `xs:time` entspricht in seiner lexikalischen Form `hh:mm:ss.sss`. Eine optionale Zeitzone kann angehängt werden. Der Buchstabe `Z` steht für Coordinated Universal Time (UTC). Alle anderen Zeitzonen werden in Form des Zeitunterschieds zur UTC im Format `+hh:mm`, oder `-hh:mm` dargestellt. Wenn kein Wert für die Zeitzone vorhanden ist, wird sie als unbekannt und nicht als UTC angenommen.

add-hours-to-time [altova:]

altova:add-hours-to-time(Time *als xs:time*, **Hours** *als xs:integer*) als **xs:time** XP3.1 XQ3.1
Fügt eine Zeitdauer in Stunden zu einem Uhrzeitwert hinzu. Beim zweiten Argument handelt es sich um die Anzahl der Stunden, die zu dem im ersten Argument angegebenen `xs:time` Wert hinzugefügt werden sollen. Das Ergebnis ist vom Typ `xs:time`.

Beispiele

- **altova:add-hours-to-time**(xs:time("11:00:00"), 10) gibt 21:00:00 zurück
- **altova:add-hours-to-time**(xs:time("11:00:00"), -7) gibt 04:00:00 zurück

add-minutes-to-time [altova:]

altova:add-minutes-to-time(Time *als xs:time*, **Minutes** *als xs:integer*) als **xs:time** XP3.1 XQ3.1
Fügt eine Zeitdauer in Minuten zu einem `xs:time` Wert hinzu. Beim zweiten Argument handelt es sich um die Anzahl der Minuten, die zu dem im ersten Argument angegebenen `xs:time` Wert hinzugefügt werden sollen. Das Ergebnis ist vom Typ `xs:time`.

Beispiele

- **altova:add-minutes-to-time**(xs:time("14:10:00"), 45) gibt 14:55:00 zurück
- **altova:add-minutes-to-time**(xs:time("14:10:00"), -5) gibt 14:05:00 zurück

add-seconds-to-time [altova:]

altova:add-seconds-to-time(Time *as xs:time*, **Minutes** *als xs:integer*) als **xs:time** XP3.1 XQ3.1
Fügt eine Zeitdauer in Sekunden zu einem Uhrzeitwert hinzu. Beim zweiten Argument handelt es sich um

 © 2019-2025 Altova GmbH

die Anzahl der Sekunden, die zu dem im ersten Argument angegebenen `xs:time` Wert hinzugefügt werden sollen. Das Ergebnis ist vom Typ `xs:time`. Die Seconds Komponenten kann sich im Bereich von `0` bis `59.999` befinden.

Beispiele

- **`altova:add-seconds-to-time`**`(xs:time("14:00:00"), 20)` gibt `14:00:20` zurück
- **`altova:add-seconds-to-time`**`(xs:time("14:00:00"), 20.895)` gibt `14:00:20.895` zurück

[Nach oben 539]

Entfernen der Zeitzone aus date/time-Datentypen XP3.1 XQ3.1

Diese Funktionen entfernen die Zeitzone aus den aktuellen **`xs:dateTime`**, **`xs:date`** bzw. **`xs:time`** Werten. Beachten Sie, dass im Unterschied zu `xs:dateTime` bei `xs:dateTimeStamp` die Zeitzone erforderlich ist (während sie im ersteren Fall optional ist). Das Format eines `xs:dateTimeStamp` Werts lautet daher: `JJJJ-MM-TTZhh:mm:ss.sss±hh:mm`. oder `JJJJ-MM-TTZhh:mm:ss.sssZ`. Wenn das Datum und die Uhrzeit von der Systemuhr als `xs:dateTimeStamp` ausgelesen wird, können Sie die Zeitzone, falls erforderlich, mit der Funktion `current-dateTime-no-TZ()` entfernen.

current-date-no-TZ [altova:]

`altova:current-date-no-TZ()` als **`xs:date`** XP3.1 XQ3.1

Die Funktion hat kein Argument. Sie entfernt die Zeitzone aus dem `current-date()` Wert (welcher das aktuelle Datum laut Systemuhr ist) und gibt einen `xs:date` Wert zurück.

Beispiele

Wenn das aktuelle Datum **`2014-01-15+01:00`** lautet:

- **`altova:current-date-no-TZ`**`()` gibt `2014-01-15` zurück

current-dateTime-no-TZ [altova:]

`altova:current-dateTime-no-TZ()` als **`xs:dateTime`** XP3.1 XQ3.1

Die Funktion hat kein Argument. Sie entfernt die Zeitzone aus dem `current-dateTime()` Wert (welcher das aktuelle Datum und die aktuelle Uhrzeit laut Systemuhr ist) und gibt einen `xs:dateTime` Wert zurück.

Beispiele

Wenn der aktuelle Datums- und Uhrzeitwert **`2014-01-15T14:00:00+01:00`** lautet:

- **`altova:current-dateTime-no-TZ`**`()` gibt `2014-01-15T14:00:00` zurück

current-time-no-TZ [altova:]

`altova:current-time-no-TZ()` als **`xs:time`** XP3.1 XQ3.1

Die Funktion hat kein Argument. Sie entfernt die Zeitzone aus dem `current-time()` Wert (welcher die aktuelle Uhrzeit laut Systemuhr ist) und gibt einen `xs:time` Wert zurück.

Beispiele

Wenn der aktuelle Uhrzeitwert **`14:00:00+01:00`** lautet:

- **altova:current-time-no-TZ**() gibt 14:00:00 zurück

date-no-TZ [altova:]

altova:date-no-TZ(InputDate *as xs:date***)** als **xs:date** XP3.1 XQ3.1
Diese Funktion verwendet ein xs:date Argument, entfernt den Zeitzonenteil daraus und gibt einen xs:date Wert zurück. Beachten Sie, dass das Datum nicht geändert wird..

Beispiele

- **altova:date-no-TZ**(xs:date("2014-01-15+01:00")) gibt 2014-01-15 zurück

dateTime-no-TZ [altova:]

altova:dateTime-no-TZ(InputDateTime *als xs:dateTime***)** als **xs:dateTime** XP3.1 XQ3.1
Diese Funktion verwendet ein xs:dateTime Argument, entfernt den Zeitzonenteil daraus und gibt einen xs:dateTime Wert zurück. Beachten Sie, dass weder Datum noch Uhrzeit geändert werden.

Beispiele

- **altova:dateTime-no-TZ**(xs:date("2014-01-15T14:00:00+01:00")) gibt 2014-01-15T14:00:00 zurück

time-no-TZ [altova:]

altova:time-no-TZ(InputTime *als xs:time***)** als **xs:time** XP3.1 XQ3.1
Diese Funktion verwendet ein xs:time Argument, entfernt den Zeitzonenteil daraus und gibt einen xs:time Wert zurück. Beachten Sie, dass die Uhrzeit nicht geändert wird.

Beispiele

- **altova:time-no-TZ**(xs:time("14:00:00+01:00")) gibt 14:00:00 zurück

[Nach oben 539]

Rückgabe der Anzahl von Tagen, Stunden, Minuten, Sekunden anhand einer Zeitdauer XP3.1 XQ3.1

Diese Funktionen geben die Anzahl der Tage in einem Monat bzw. die Anzahl der Stunden, Minuten und Sekunden anhand einer Zeitdauer zurück.

days-in-month [altova:]

altova:days-in-month(Year *als xs:integer*, **Month** *als xs:integer***)** als **xs:integer** XP3.1 XQ3.1
Gibt die Anzahl der Tage im angegebenen Monat zurück. Der Monat wird mit Hilfe der Argumente Year und Month angegeben.

Beispiele

- **altova:days-in-month**(2018, 10) gibt 31 zurück
- **altova:days-in-month**(2018, 2) gibt 28 zurück
- **altova:days-in-month**(2020, 2) gibt 29 zurück

© 2019-2025 Altova GmbH

hours-from-dayTimeDuration-accumulated

altova:hours-from-dayTimeDuration-accumulated(DayAndTime *als xs:duration***)** als **xs:integer**
XP3.1 **XQ3.1**

Gibt die Gesamtzahl der Stunden in der durch das Argument `DayAndTime` bereitgestellten Zeitdauer (vom Typ `xs:duration`) zurück. Die Stunden in den Komponenten `Day` und `Time` werden addiert, um ein Ergebnis in Form einer Ganzzahl zu erhalten. Nur für volle 60 Minuten wird eine neue Stunde berechnet. Negative Zeitdauerergebnisse ergeben einen negativen Stundenwert.

Beispiele

- **altova:hours-from-dayTimeDuration-accumulated**(xs:duration("P5D")) gibt 120 zurück, d.h. die Gesamtanzahl der Stunden in 5 Tagen.
- **altova:hours-from-dayTimeDuration-accumulated**(xs:duration("P5DT2H")) gibt 122 zurück, d.h. die Gesamtzahl der Stunden in 5 Tagen plus 2 Stunden.
- **altova:hours-from-dayTimeDuration-accumulated**(xs:duration("P5DT2H60M")) gibt 123 zurück, d.h. die Gesamtzahl der Stunden in 5 Tagen plus 2 Stunden und 60 Minuten.
- **altova:hours-from-dayTimeDuration-accumulated**(xs:duration("P5DT2H119M")) gibt 123 zurück, d.h. die Gesamtzahl der Stunden in 5 Tagen plus 2 Stunden und 119 Minuten.
- **altova:hours-from-dayTimeDuration-accumulated**(xs:duration("P5DT2H120M")) gibt 124 zurück, d.h. die Gesamtzahl der Stunden in 5 Tagen plus 2 Stunden und 120 Minuten.
- **altova:hours-from-dayTimeDuration-accumulated**(xs:duration("-P5DT2H")) gibt -122 zurück.

minutes-from-dayTimeDuration-accumulated

altova:minutes-from-dayTimeDuration-accumulated(DayAndTime *als xs:duration***)** als **xs:integer** **XP3.1** **XQ3.1**

Gibt die Gesamtzahl der Minuten in der durch das Argument `DayAndTime` bereitgestellten Zeitdauer (vom Typ `xs:duration`) zurück. Die Minuten in den Komponenten `Day` und `Time` werden addiert, um ein Ergebnis in Form einer Ganzzahl zu erhalten. Negative Zeitdauerergebnisse ergeben einen negativen Minutenwert.

Beispiele

- **altova:minutes-from-dayTimeDuration-accumulated**(xs:duration("PT60M")) gibt 60 zurück.
- **altova:minutes-from-dayTimeDuration-accumulated**(xs:duration("PT1H")) gibt 60 zurück, d.h. die Gesamtzahl der Minuten in 1 Stunde.
- **altova:minutes-from-dayTimeDuration-accumulated**(xs:duration("PT1H40M") gibt 100 zurück.
- **altova:minutes-from-dayTimeDuration-accumulated**(xs:duration("P1D")) gibt 1440 zurück, d.h. die Gesamtzahl der Minuten an einem Tag.
- **altova:minutes-from-dayTimeDuration-accumulated**(xs:duration("-P1DT60M")) gibt -1500 zurück.

seconds-from-dayTimeDuration-accumulated

altova:seconds-from-dayTimeDuration-accumulated(DayAndTime *als xs:duration***)** als **xs:integer** **XP3.1** **XQ3.1**

Gibt die Gesamtzahl der Sekunden in der durch das Argument `DayAndTime` bereitgestellten Zeitdauer (vom Typ `xs:duration`) zurück. Die Sekunden in den Komponenten `Day` und `Time` werden addiert, um ein Ergebnis in Form einer Ganzzahl zu erhalten. Negative Zeitdauerergebnisse ergeben einen negativen Sekundenwert.

Examples

- **altova:seconds-from-dayTimeDuration-accumulated**(xs:duration("PT1M")) gibt 60 zurück, d.h. die Gesamtzahl der Sekunden in 1 Minute.
- **altova:seconds-from-dayTimeDuration-accumulated**(xs:duration("PT1H")) gibt 3600 zurück, d.h. die Gesamtzahl der Sekunden in 1 Stunde.
- **altova:seconds-from-dayTimeDuration-accumulated**(xs:duration("PT1H2M")) gibt 3720 zurück.
- **altova:seconds-from-dayTimeDuration-accumulated**(xs:duration("P1D"))gibt 86400 zurück, d.h. die Gesamtzahl der Sekunden an 1 Tag.
- **altova:seconds-from-dayTimeDuration-accumulated**(xs:duration("-P1DT1M")) gibt -86460 zurück.

Rückgabe des Wochentages anhand von xs:dateTime oder xs:date XP3.1 XQ3.1

Diese Funktionen geben anhand des `xs:dateTime` oder `xs:date` Werts den Wochentag in Form einer Ganzzahl zurück. Die Tage der Woche sind (im amerikanischen Format) von `1` bis `7` nummeriert, wobei `Sonntag=1`. Im europäischen Format beginnt die Woche am Montag (`=1`). Das amerikanische Format, in dem `Sonntag=1`, kann mittels der Ganzzahl `0` definiert werden, wenn das Format mittels einer Ganzzahl angegeben werden kann.

weekday-from-dateTime [altova:]

altova:weekday-from-dateTime(DateTime *als xs:dateTime***) als xs:integer** XP3.1 XQ3.1

Erhält ein Datum mit einer Uhrzeit als einziges Argument und gibt den Tag der Woche dieses Datums in Form einer Ganzzahl zurück. Die Wochentage sind beginnend mit `Sonntag=1` nummeriert. Wenn das europäische Format benötigt wird (wo `Montag=1`), verwenden Sie die andere Signatur dieser Funktion (*siehe nächste Signatur unten*).

Beispiele

- **altova:weekday-from-dateTime**(xs:dateTime("2014-02-03T09:00:00")) gibt 2 zurück, wobei 2 für Montag steht.

altova:weekday-from-dateTime(DateTime *als xs:dateTime***, Format** *als xs:integer***) als xs:integer** XP3.1 XQ3.1

Erhält ein Datum mit einer Uhrzeit als erstes Argument und gibt den Tag der Woche dieses Datums in Form einer Ganzzahl zurück. Wenn das zweite (Integer)-Argument `0` ist, werden die Wochentage beginnend mit `Sonntag=1` von `1` bis `7` nummeriert. Wenn das zweite Argument eine andere Ganzzahl als `0` ist, so ist `Montag=1`. Wenn es kein zweites Argument gibt, wird die Funktion gelesen, als ob sie die andere Signatur dieser Funktion hätte (*siehe vorherige Signatur*).

Beispiele

- **altova:weekday-from-dateTime**(xs:dateTime("2014-02-03T09:00:00"), 1) gibt 1 zurück, wobei 1 für Montag steht
- **altova:weekday-from-dateTime**(xs:dateTime("2014-02-03T09:00:00"), 4) gibt 1 zurück, wobei 1 für Montag steht

© 2019-2025 Altova GmbH

- `altova:weekday-from-dateTime(xs:dateTime("2014-02-03T09:00:00"), 0)` gibt `2` zurück, wobei 2 für Montag steht.

weekday-from-date [altova:]

altova:weekday-from-date(Date *als xs:date*) **als xs:integer** XP3.1 XQ3.1
Erhält ein Datum als einziges Argument und gibt den Tag der Woche dieses Datums in Form einer Ganzzahl zurück. Die Wochentage sind beginnend mit `Sonntag=1` nummeriert. Wenn das europäische Format benötigt wird (wo `Montag=1`), verwenden Sie die andere Signatur dieser Funktion (*siehe nächste Signatur unten*).

Beispiele

- `altova:weekday-from-date(xs:date("2014-02-03+01:00"))` gibt `2` zurück, d.h. dies wäre ein Montag.

altova:weekday-from-date(Date *als xs:date*, **Format** *als xs:integer*) **als xs:integer** XP3.1 XQ3.1
Erhält ein Datum als erstes Argument und gibt den Tag der Woche dieses Datums in Form einer Ganzzahl zurück. Wenn das zweite Argument (`Format`) `0` ist, werden die Wochentage beginnend mit `Sonntag=1` von `1` bis `7` nummeriert. Wenn das zweite Argument eine andere Ganzzahl als `0` ist, so ist `Montag=1`. Wenn es kein zweites Argument gibt, wird die Funktion gelesen, als ob sie die andere Signatur dieser Funktion hätte (*siehe vorherige Signatur*).

Beispiele

- `altova:weekday-from-date(xs:date("2014-02-03"), 1)` gibt `1` zurück, wobei 1 für Montag steht
- `altova:weekday-from-date(xs:date("2014-02-03"), 4)` gibt `1` zurück, wobei 1 für Montag steht
- `altova:weekday-from-date(xs:date("2014-02-03"), 0)` gibt `2` zurück, wobei 2 für Montag steht.

[Nach oben 539]

Rückgabe der Wochennummer anhand von xs:dateTime oder xs:date XP2 XQ1 XP3.1 XQ3.1

Diese Funktionen geben anhand von `xs:dateTime` oder `xs:date` die Wochennummer als Ganzzahl zurück. Die Wochennummer steht in den Kalenderformaten US, ISO/European und Islamic zur Verfügung. Die Wochennummerierung unterscheidet sich in diesen Kalenderformaten, da die Woche in diesen Formaten an unterschiedlichen Tagen beginnt (Im Format US am Sonntag, im Format ISO/European am Montag und im Format Islamic am Samstag).

weeknumber-from-date [altova:]

altova:weeknumber-from-date(Date *als xs:date*, **Calendar** *als xs:integer*) als **xs:integer** XP2 XQ1 XP3.1 XQ3.1
Gibt die Wochennummer des bereitgestellten **Date** Arguments als Ganzzahl zurück. Das zweite Argument (**Calendar**) definiert das zu verwendende Kalendersystem.
Unterstützte **Calendar** Werte sind:

- **0 = US-Kalender** (Woche beginnt am Sonntag)
- **1 = ISO-Standard, Europäischer Kalender** (Woche beginnt am Montag)
- **2 = Islamischer Kalender** (Woche beginnt am Samstag)

Der Standardwert ist **0**.

Beispiele

- `altova:weeknumber-from-date(xs:date("2014-03-23"), 0)` gibt `13` zurück
- `altova:weeknumber-from-date(xs:date("2014-03-23"), 1)` gibt `12` zurück
- `altova:weeknumber-from-date(xs:date("2014-03-23"), 2)` gibt `13` zurück
- `altova:weeknumber-from-date(xs:date("2014-03-23") )` gibt `13` zurück

Der Tag des Datums in den obigen Beispielen (`2014-03-23`) ist ein Sonntag. Daher ist der US- und der islamische Kalender dem europäischen Kalender an diesem Tag eine Woche voraus.

weeknumber-from-dateTime [altova:]

altova:weeknumber-from-dateTime(DateTime *als xs:dateTime*, **Calendar** *als xs:integer*) **als xs:integer** XP2 XQ1 XP3.1 XQ3.1

Gibt die Wochennummer des bereitgestellten **DateTime** Arguments als Ganzzahl zurück. Das zweite Argument (**Calendar**) definiert das zu verwendende Kalendersystem.
Unterstützte **Calendar** Werte sind:

- **0 = US-Kalender** (*Woche beginnt am Sonntag*)
- **1 = ISO-Standard, Europäischer Kalender** (*Woche beginnt am Montag*)
- **2 = Islamischer Kalender** (*Woche beginnt am Samstag*)

Der Standardwert ist **0**.

Beispiele

- `altova:weeknumber-from-dateTime(xs:dateTime("2014-03-23T00:00:00"), 0)` gibt `13` zurück
- `altova:weeknumber-from-dateTime(xs:dateTime("2014-03-23T00:00:00"), 1)` gibt `12` zurück
- `altova:weeknumber-from-dateTime(xs:dateTime("2014-03-23T00:00:00"), 2)` gibt `13` zurück
- `altova:weeknumber-from-dateTime(xs:dateTime("2014-03-23T00:00:00") )` gibt `13` zurück

Der Tag des Datums- und Uhrzeitwerts in den obigen Beispielen (`2014-03-23T00:00:00`) ist ein Sonntag. Daher ist der US- und der islamische Kalender dem europäischen Kalender an diesem Tag eine Woche voraus.

[Nach oben 539]

© 2019-2025 Altova GmbH

Erstellen des Datums-, Uhrzeit- oder Zeitdauer-Datentyps anhand der lexikalischen Komponenten der einzelnen Typen XP3.1 XQ3.1

Die Funktionen erhalten die lexikalischen Komponenten des `xs:date`, `xs:time` oder `xs:duration`-Datentyps als Input-Argumente und kombinieren diese zum entsprechenden Datentyp.

build-date [altova:]

altova:build-date(Year *als xs:integer*, **Month** *als xs:integer*, **Date** *als xs:integer*) als **xs:date** XP3.1 XQ3.1

Das erste, zweite und dritte Argument steht für das Jahr, bzw. den Monat bzw. das Datum. Sie werden zu einem Wert vom Typ `xs:date` kombiniert. Die Werte der Ganzzahlen müssen sich innerhalb des korrekten Bereichs dieses jeweiligen Datumsteils befinden. So sollte z.B. das zweite Argument nicht größer als `12` sein.

Beispiele

- `altova:build-date(2014, 2, 03)` gibt `2014-02-03` zurück

build-time [altova:]

altova:build-time(Hours *als xs:integer*, **Minutes** *als xs:integer*, **Seconds** *als xs:integer*) als **xs:time** XP3.1 XQ3.1

Das erste, zweite und dritte Argument steht für die Stunde (`0` bis `23`), bzw. die Minuten (`0` bis `59`) bzw. die Sekunden (`0` bis `59`). Sie werden zu einem Wert von Typ `xs:time` kombiniert. Die Werte der Ganzzahlen müssen sich innerhalb des korrekten Bereichs dieses jeweiligen Uhrzeitteils befinden. So sollte z.B. der zweite Parameter nicht größer als `59` sein. Um eine Zeitzone zum Wert hinzuzufügen, verwenden Sie die andere Signatur der Funktion (*siehe nächste Signatur*).

Beispiele

- `altova:build-time(23, 4, 57)` gibt `23:04:57` zurück

altova:build-time(Hours *als xs:integer*, **Minutes** *als xs:integer*, **Seconds** *als xs:integer*, **TimeZone** *als xs:string*) als **xs:time** XP3.1 XQ3.1

Das erste, zweite und dritte Argument steht für die Stunde (`0` bis `23`), bzw. die Minuten (`0` bis `59`) bzw. die Sekunden (`0` bis `59`). Das vierte Argument ist ein String, der den Zeitzonenteil des Werts liefert. Die vier Argumente werden zu einem Wert vom Typ `xs:time` kombiniert. Die Werte der Ganzzahlen müssen sich innerhalb des korrekten Bereichs dieses jeweiligen Uhrzeitteils befinden. So sollte z.B. das zweite Argument (Minuten) nicht größer als `59` sein.

Beispiele

- `altova:build-time(23, 4, 57, '+1')` gibt `23:04:57+01:00` zurück

build-duration [altova:]

altova:build-duration(Years *als xs:integer*, **Months** *als xs:integer*) als **xs:yearMonthDuration** XP3.1 XQ3.1

Setzt aus zwei Argumenten einen Wert vom Typ `xs:yearMonthDuration` zusammen. Das erste Argument liefert den Jahr-Teil des Zeitdauerwerts, während das zweite Argument den Monat-Teil liefert. Wenn der zweite Parameter (Monate) größer oder gleich `12` ist, so wird die Ganzzahl durch 12 dividiert. Der Quotient wird zum ersten Argument hinzugefügt, um den Jahr-Teil des Zeitdauerwerts zu liefern,

während der Rest (der Division) den Monat-Teil liefert. Eine Beschreibung zur Erstellung einer Zeitdauer vom Typ `xs:dayTimeDuration` finden Sie in der nächsten Signatur.

Beispiele

- **altova:build-duration**(2, 10) gibt P2Y10M zurück
- **altova:build-duration**(14, 27) gibt P16Y3M zurück
- **altova:build-duration**(2, 24) gibt P4Y zurück

altova:build-duration(Days *als xs:integer*, **Hours** *als xs:integer*, **Minutes** *als xs:integer*, **Seconds** *als xs:integer*) als **xs:dayTimeDuration** XP3.1 XQ3.1

Kombiniert vier Argumente zu einem Wert vom Typ `xs:dayTimeDuration`. Das erste Argument liefert den Tage-Teil, das zweite die Stunden, das dritte die Minuten und das vierte die Sekunden des Zeitdauerwerts. Die einzelnen Uhrzeitparameter werden in den entsprechenden Wert für die nächsthöhere Einheit konvertiert und das Ergebnis wird zur Berechnung der Gesamtdauer weitergegeben. So werden z.B. 72 Sekunden in `1M(inute)12S(ekunden)` konvertiert. Dieser Wert wird zur Berechnung der Gesamtdauer weitergegeben. Um eine Zeitdauer vom Typ `xs:yearMonthDuration` zu berechnen, verwenden Sie die vorherige Signatur.

Beispiele

- **altova:build-duration**(2, 10, 3, 56) gibt P2DT10H3M56S zurück
- **altova:build-duration**(1, 0, 100, 0) gibt P1DT1H40M zurück
- **altova:build-duration**(1, 0, 0, 3600) gibt P1DT1H zurück

[Nach oben 539]

Konstruieren von Datum, Datum und Uhrzeit und Zeit-Datentypen anhand des String-Input XP2 XQ1 XP3.1 XQ3.1

Diese Funktionen erhalten Strings als Argumente und konstruieren anhand dieser die Datentypen `xs:date`, `xs:dateTime` oder `xs:time`. Der String wird anhand eines bereitgestellten Pattern-Arguments nach Komponenten des Datentyps analysiert.

parse-date [altova:]

altova:parse-date(Date *als xs:string*, **DatePattern** *als xs:string*) als **xs:date** XP2 XQ1 XP3.1 XQ3.1

Gibt den Input-String **Date** als **xs:date** Wert zurück. Das zweite Argument **DatePattern** definiert das Pattern (die Komponentensequenz) des Input-String. **DatePattern** wird durch die unten aufgelisteten Komponenten-Specifier beschrieben. Als Komponententrennzeichen kann jedes beliebige Zeichen verwendet werden. Siehe Beispiele unten.

D	Datum
M	Monat
Y	Jahr

Das Pattern in **DatePattern** muss mit dem Pattern in **Date** übereinstimmen. Da die Ausgabe vom Typ **xs:date** ist, hat sie immer das lexikalische Format **YYYY-MM-DD**.

Beispiele

- **altova:parse-date**(xs:string("09-12-2014"), "[D]-[M]-[Y]") gibt 2014-12-09 zurück
- **altova:parse-date**(xs:string("09-12-2014"), "[M]-[D]-[Y]") gibt 2014-09-12 zurück

 © 2019-2025 Altova GmbH

- `altova:parse-date("06/03/2014", "[M]/[D]/[Y]")` gibt `2014-06-03` zurück
- `altova:parse-date("06 03 2014", "[M] [D] [Y]")` gibt `2014-06-03` zurück
- `altova:parse-date("6 3 2014", "[M] [D] [Y]")` gibt `2014-06-03` zurück

parse-dateTime [altova:]

altova:parse-dateTime(DateTime *als xs:string*, **DateTimePattern** *als xs:string*) als **xs:dateTime** XP2 XQ1 XP3.1 XQ3.1

Gibt den Input-String **DateTime** als **xs:dateTime** Wert zurück. Das zweite Argument **DateTimePattern** definiert das Pattern (die Komponentensequenz) des Input-String. **DateTimePattern** wird durch die unten aufgelisteten Komponenten-Specifier beschrieben. Als Komponententrennzeichen kann jedes beliebige Zeichen verwendet werden. Siehe Beispiele unten.

D	Datum
M	Monat
Y	Jahr
H	Stunde
m	Minuten
s	Sekunden

Das Pattern in **DateTimePattern** muss mit dem Pattern in **DateTime** übereinstimmen. Da die Ausgabe vom Typ **xs:dateTime** ist, hat sie immer das lexikalische Format **YYYY-MM-DDTHH:mm:ss**.

Beispiele

- `altova:parse-dateTime(xs:string("09-12-2014 13:56:24"), "[M]-[D]-[Y] [H]:[m]:[s]")` gibt `2014-09-12T13:56:24` zurück
- `altova:parse-dateTime("time=13:56:24; date=09-12-2014", "time=[H]:[m]:[s]; date=[D]-[M]-[Y]")` gibt `2014-12-09T13:56:24` zurück

parse-time [altova:]

altova:parse-time(Time *als xs:string*, **TimePattern** *als xs:string*) als **xs:time** XP2 XQ1 XP3.1 XQ3.1

Gibt den Input-String **Time** als **xs:time** Wert zurück. Das zweite Argument **TimePattern** definiert das Pattern (die Komponentensequenz) des Input-String. **TimePattern** wird durch die unten aufgelisteten Komponenten-Specifier beschrieben. Als Komponententrennzeichen kann jedes beliebige Zeichen verwendet werden. Siehe Beispiele unten.

H	Stunde
m	Minuten
s	Sekunden

Das Pattern in **TimePattern** muss mit dem Pattern in **Time** übereinstimmen. Da die Ausgabe vom Typ **xs:Time** ist, hat sie immer das lexikalische Format **HH:mm:ss**.

Beispiele

- `altova:parse-time(xs:string("13:56:24"), "[H]:[m]:[s]")` gibt `13:56:24` zurück

- **altova:parse-time**("13-56-24", "[H]-[m]") gibt 13:56:00 zurück
- **altova:parse-time**("time=13h56m24s", "time=[H]h[m]m[s]s") gibt 13:56:24 zurück
- **altova:parse-time**("time=24s56m13h", "time=[s]s[m]m[H]h") gibt 13:56:24 zurück

[Nach oben 539]

Funktionen zur Berechnung des Alters XP3.1 XQ3.1

Diese Funktionen geben das Alter berechnet (i) anhand von einem Input-Argument und dem aktuellen Datum oder (ii) anhand zweier Input-Argumentdaten zurück. Die Funktion **altova:age** gibt das Alter in Jahren zurück, die Funktion **altova:age-details** gibt das Alter als Sequenz von drei Ganzzahlen zurück, die die Jahre, Monate und Tage des Alters angeben.

age [altova:]

altova:age(StartDate *als xs:date***)** als **xs:integer** XP3.1 XQ3.1
Gibt eine Ganzzahl zurück, die das Alter eines Objekts in *Jahren* angibt. Berechnet wird das Alter anhand des durch das Argument gelieferten Startdatums endend mit dem aktuellen Datum (laut Systemuhr). Wenn das Input-Argument eines Datums größer oder gleich einem Jahr in der Zukunft ist, ist der Rückgabewert negativ.

Beispiele

Wenn das aktuelle Datum **2014-01-15** lautet:

- **altova:age**(xs:date("2013-01-15")) gibt 1 zurück
- **altova:age**(xs:date("2013-01-16")) gibt 0 zurück
- **altova:age**(xs:date("2015-01-15")) gibt -1 zurück
- **altova:age**(xs:date("2015-01-14")) gibt 0 zurück

altova:age(StartDate *als xs:date*, **EndDate** *als xs:date***)** als **xs:integer** XP3.1 XQ3.1
Gibt eine Ganzzahl zurück, die das Alter eines Objekts in *Jahren* angibt. Berechnet wird das Alter anhand des durch das erste Argument gelieferten Startdatums endend mit dem als zweites Datum gelieferten Enddatum. Wenn das erste Argument ein Jahr oder mehr nach dem zweiten Argument liegt, ist der Rückgabewert negativ.

Beispiele

Wenn das aktuelle Datum **2014-01-15** lautet:

- **altova:age**(xs:date("2000-01-15"), xs:date("2010-01-15")) gibt 10 zurück
- **altova:age**(xs:date("2000-01-15"), current-date()) gibt 14 zurück, wenn das aktuelle Datum 2014-01-15 ist
- **altova:age**(xs:date("2014-01-15"), xs:date("2010-01-15")) gibt -4 zurück

age-details [altova:]

altova:age-details(InputDate *als xs:date***)** als (**xs:integer**)* XP3.1 XQ3.1
Gibt drei Ganzzahlen zurück. Dabei handelt es sich um die Jahre, Monate bzw. Tage zwischen dem als

Argument angegebenen Datum und dem aktuellen Datum (laut Systemuhr). Die Summe der zurückgegebenen `years+months+days` gibt zusammen die Gesamtzeitdifferenz zwischen den beiden Datumswerten (dem Input-Datum und dem aktuellen Datum) an. Das Input-Datum hat eventuell einen Wert, der vor oder nach dem aktuellen Datum liegt, doch wird dies nicht aus dem Vorzeichen der Rückgabewerte ersichtlich; die Rückgabewerte sind immer positiv.

Beispiele

Wenn das aktuelle Datum **2014-01-15** lautet:

- **altova:age-details**(xs:date("2014-01-16")) gibt (0 0 1) zurück
- **altova:age-details**(xs:date("2014-01-14")) gibt (0 0 1) zurück
- **altova:age-details**(xs:date("2013-01-16")) gibt (1 0 1) zurück
- **altova:age-details**(current-date()) gibt (0 0 0) zurück

altova:age-details(Date-1 *als xs:date*, **Date-2** *als xs:date*) als (**xs:integer**)* XP3.1 XQ3.1
Gibt drei Ganzzahlen zurück. Dabei handelt es sich um die Jahre, Monate bzw. Tage zwischen den beiden Argumentdaten. Die Summe der zurückgegebenen `years+months+days` gibt zusammen die Gesamtzeitdifferenz zwischen den beiden Input-Datumswerten an. Es ist unerheblich, ob das frühere oder spätere Datum als erstes Argument angegeben wird. Die Rückgabewerte geben nicht an, ob das Input-Datum vor oder nach dem aktuellen Datum liegt. Die Rückgabewerte sind immer positiv.

Beispiele

- **altova:age-details**(xs:date("2014-01-16"), xs:date("2014-01-15")) gibt (0 0 1) zurück
- **altova:age-details**(xs:date("2014-01-15"), xs:date("2014-01-16")) gibt (0 0 1) zurück

[Nach oben 539]

Epochen-Zeit (Unix-Zeit)-Funktionen XP3.1 XQ3.1

Die Epochenzeit ist ein auf Unix-Systemen verwendetes Zeitsystem. Darin wird jeder Zeitpunkt als Anzahl der Sekunden seit 00:00:00 UTC des 1. Januar 1970 definiert. Diese Epochenzeitfunktionen konvertieren **xs:dateTime**-Werte in Epochenzeitwerte und umgekehrt.

dateTime-from-epoch [altova:]

altova:dateTime-from-epoch(Epoch *als xs:decimal* **als xs:dateTime** XP3.1 XQ3.1
Die Epochenzeit ist ein auf Unix-Systemen verwendetes Zeitsystem. Darin wird jeder Zeitpunkt als Anzahl der Sekunden seit 00:00:00 UTC des 1. Januar 1970 definiert. Die Funktion **dateTime-from-epoch** gibt das **xs:dateTime**-Äquivalent einer Epochenzeit zurück, passt die lokale Zeitzone an und inkludiert die Zeitzoneninformation im Ergebnis.

Die Funktion erhält ein `xs:decimal`-Argument und gibt einen `xs:dateTime`-Wert, der einen **ZZ**-Teil (Zeitzone) enthält, zurück. Das Ergebnis wird durch Berechnung des UTC **dateTime**-Äquivalents der Epochenzeit und Hinzufügen der (anhand der Systemuhr ermittelten) lokalen Zeitzone ermittelt. Wenn die Funktion z.B. auf einem Rechner, der in der Zeitzone +01:00 (relativ zur UTC) konfiguriert wurde, ausgeführt wird, so wird nach Berechnung des UTC-**dateTime**-Äquivalents eine Stunde zum Ergebnis addiert. Auch die Zeitzoneninformation, die einen optionalen lexikalischen Bestandteil des **xs:dateTime**-Ergebnisses bildet, wird im **dateTime**-Ergebnis ausgegeben. Vergleichen Sie dieses Ergebnis mit dem von **dateTime-from-epoch-no-TZ** und auch der Funktion **epoch-from-dateTime**.

Beispiele

In den Beispielen unten wird eine lokale Zeitzone von UTC +01:00 angenommen. Das UTC **dateTime**-Äquivalent der angegebenen Epochenzeit wird folglich um eine Stunde erhöht. Die Zeitzone wird im Ergebnis ausgegeben.

- **altova:dateTime-from-epoch**(34) gibt 1970-01-01T01:00:34+01:00 zurück.
- **altova:dateTime-from-epoch**(62) gibt 1970-01-01T01:01:02+01:00 zurück.

dateTime-from-epoch-no-TZ [altova:]

altova:dateTime-from-epoch-no-TZ(Epoch *als xs:decimal* **als xs:dateTime** XP3.1 XQ3.1
Die Epochenzeit ist ein auf Unix-Systemen verwendetes Zeitsystem. Darin wird jeder Zeitpunkt als Anzahl der Sekunden seit 00:00:00 UTC des 1. Januar 1970 definiert. Die Funktion **dateTime-from-epoch-no-TZ** gibt das **xs:dateTime**-Äquivalent einer Epochenzeit zurück, passt es an die lokale Zeitzone an, inkludiert die Zeitzoneninformation jedoch nicht im Ergebnis.

Die Funktion erhält ein xs:decimal-Argument und gibt einen xs:dateTime-Wert, der keinen **ZZ**-Teil (Zeitzone) enthält, zurück. Das Ergebnis wird durch Berechnung des UTC **dateTime**-Äquivalents der Epochenzeit und Hinzufügen der (anhand der Systemuhr ermittelten) lokalen Zeitzone ermittelt. Wenn die Funktion z.B. auf einem Rechner, der in der Zeitzone +01:00 (relativ zur UTC) konfiguriert wurde, ausgeführt wird, so wird nach Berechnung des UTC-**dateTime**-Äquivalents eine Stunde zum Ergebnis addiert. Die Zeitzoneninformation, die einen optionalen lexikalischen Bestandteil des **xs:dateTime**-Ergebnisses bildet, wird nicht im **dateTime**-Ergebnis ausgegeben. Vergleichen Sie dieses Ergebnis mit dem von **dateTime-from-epoch** und auch der Funktion **epoch-from-dateTime**.

Beispiele

In den Beispielen unten wird eine lokale Zeitzone von UTC +01:00 angenommen. Das UTC **dateTime**-Äquivalent der angegebenen Epochenzeit wird folglich um eine Stunde erhöht. Die Zeitzone wird nicht im Ergebnis ausgegeben.

- **altova:dateTime-from-epoch**(34) gibt 1970-01-01T01:00:34 zurück.
- **altova:dateTime-from-epoch**(62) gibt 1970-01-01T01:01:02 zurück.

epoch-from-dateTime [altova:]

altova:epoch-from-dateTime(dateTimeValue *als xs:dateTime*) **als xs:decimal** XP3.1 XQ3.1
Die Epochenzeit ist ein auf Unix-Systemen verwendetes Zeitsystem. Darin wird jeder Zeitpunkt als Anzahl der Sekunden seit 00:00:00 UTC des 1. Januar 1970 definiert. Die Funktion **epoch-from-dateTime** gibt das Epochenzeitäquivalent von **xs:dateTime** zurück, welches als Argument der Funktion bereitgestellt wird. Beachten Sie, dass Sie den **xs:dateTime**-Wert eventuell explizit konstruieren müssen. Der angegebene **xs:dateTime**-Wert kann den optionalen **ZZ** (Zeitzonen)-Wert enthalten, muss ihn aber nicht enthalten.

Unabhängig davon, ob der Zeitzonenteil als Bestandteil des Arguments angegeben wird oder nicht, wird der (anhand der Systemuhr ermittelte) lokale Zeitzonenunterschied vom angegebenen **dateTimeValue**-Argument subtrahiert. Dadurch wird das UTC-Zeit-Äquivalent erzeugt, anhand dessen die entsprechende Epochenzeit berechnet wird. Wenn die Funktion z.B. auf einem Rechner, der für die Zeitzone +01:00 (relativ zur UTC) konfiguriert wurde, ausgeführt wird, so wird vor Berechnung des Epochenzeitwerts eine Stunde vom angegebenen **dateTimeValue** subtrahiert. Siehe dazu auch die Funktion **dateTime-from-**

© 2019-2025 Altova GmbH

epoch.

Beispiele

In den Beispielen unten wird eine lokale Zeitzone von UTC +01:00 angenommen. Daher wird vor Berechnung der Epochenzeit eine Stunde vom angegebenen **dateTime**-Wert subtrahiert.

- **altova:epoch-from-dateTime**(xs:dateTime("1970-01-01T01:00:34+01:00")) gibt 34 zurück.
- **altova:epoch-from-dateTime**(xs:dateTime("1970-01-01T01:00:34")) gibt 34 zurück.
- **altova:epoch-from-dateTime**(xs:dateTime("2021-04-01T11:22:33")) gibt 1617272553 zurück.

[Nach oben 539]

12.2.2.1.3 XPath/XQuery-Funktionen: Standort

Die folgenden XPath/XQuery-Erweiterungsfunktionen zu Standortdaten werden in der aktuellen Version von MapForce unterstützt und können in (i) in einem XSLT-Kontext in XPath-Ausdrücken oder (ii) in einem XQuery-Dokument in einem XQuery-Ausdruck verwendet werden.

Anmerkung zur Benennung von Funktionen und zur Anwendbarkeit der Sprache

Altova-Erweiterungsfunktionen können in XPath/XQuery-Ausdrücken verwendet werden. Dadurch stehen neben den Funktionen in der Standardbibliothek der XPath-, XQuery- und XSLT-Funktionen zusätzliche Funktionen zur Verfügung. Die Altova-Erweiterungsfunktionen befinden sich im **Altova-Erweiterungsfunktions-Namespace, http://www.altova.com/xslt-extensions** und sind in diesem Abschnitt mit dem Präfix **altova:**, das als an diesen Namespace gebunden angenommen wird, gekennzeichnet. Beachten Sie, dass manche Funktionen in zukünftigen Versionen Ihres Produkts eventuell nicht mehr unterstützt werden oder dass sich das Verhalten einzelner Funktionen ändern kann. Um zu sehen, welche Altova-Erweiterungsfunktionen unterstützt werden, lesen Sie bitte die Dokumentation zur jeweiligen Release.

XPath-Funktionen (in XPath-Ausdrücken in XSLT verwendet):	XP1 XP2 XP3.1
XSLT-Funktionen (in XPath-Ausdrücken in XSLT verwendet):	XSLT1 XSLT2 XSLT3
XQuery-Funktionen (in XQuery-Ausdrücken in XQuery verwendet):	XQ1 XQ3.1

format-geolocation [altova:]

altova:format-geolocation(Latitude *als xs:decimal*, **Longitude** *als xs:decimal*, **GeolocationOutputStringFormat** *als xs:integer*) als **xs:string** XP3.1 XQ3.1

Erhält als die ersten beiden Argumente die geografische Breite und Länge und gibt den Standort als String zurück. Das dritte Argument, **GeolocationOutputStringFormat**, ist das Format des Ausgabestring für den Standort; darin werden zum Identifizieren des Ausgabestringformats Ganzzahlwerte von 1 bis 4 verwendet (*siehe 'Format des Ausgabestrings für die geografische Position' weiter unten*). Die Werte für die Breite liegen im Bereich von +90 bis -90 (N nach S). Die Werte für die Länge liegen im Bereich von +180 bis -180 (O nach W).

Anmerkung: Zur Bereitstellung der Input-Strings können die Funktion image-exif-data 569 und die

Attribute der Exif-Metadaten verwendet werden.

Beispiele

- **altova:format-geolocation**(33.33, -22.22, 4) gibt xs:string "33.33 -22.22" zurück
- **altova:format-geolocation**(33.33, -22.22, 2) gibt xs:string "33.33N 22.22W" zurück
- **altova:format-geolocation**(-33.33, 22.22, 2) gibt xs:string "33.33S 22.22E" zurück
- **altova:format-geolocation**(33.33, -22.22, 1) gibt xs:string "33°19'48.00"S 22° 13'12.00"E" zurück

Ausgabestringformate für die geografische Position:

Die bereitgestellte Breite und Länge ist in einem der unten aufgelisteten Ausgabeformate formatiert. Das gewünschte Format wird anhand seiner Ganzzahl-ID (1 bis 4) identifiziert. Die Breitenwerte liegen im Bereich von +90 bis -90 (N nach S). Die Längenwerte liegen im Bereich von +180 bis -180 (O nach W).

1
Grad, Minuten, Dezimalsekunden, mit nachgestellter Orientierung (N/S, E/W) **D°M'S.SS"N/S D°M'S.SS"E/W** *Beispiel*: **33°55'11.11"N 22°44'66.66"W**

2
Dezimalgrad, mit nachgestellter Orientierung (N/S, E/W) **D.DDN/S D.DDE/W** *Beispiel*: **33.33N 22.22W**

3
Grad, Minuten, Dezimalsekunden mit Vorzeichen (+/-); Plus-Zeichen für (**N/E**) ist optional **+/-D°M'S.SS" +/-D°M'S.SS"** *Beispiel*: **33°55'11.11" -22°44'66.66"**

4
Dezimalgrad, mit Vorzeichen (+/-); Plus-Zeichen für (**N/E**) ist optional **+/-D.DD +/-D.DD** *Beispiel*: **33.33 -22.22**

Altova Exif-Attribut: Geolocation

Der Altova XPath/XQuery-Prozessor generiert anhand der Exif-Standard-Metadaten-Tags das benutzerdefinierte Attribut **Geolocation**. **Geolocation** ist eine Verkettung von vier Exif-Tags: GPSLatitude, GPSLatitudeRef, GPSLongitude, GPSLongitudeRef mit hinzugefügten Werten (*siehe Tabelle unten*).

GPSLatitude	GPSLatitudeRef	GPSLongitude	GPSLongitudeRef	Geolocation

© 2019-2025 Altova GmbH

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 *als xs:string***)** als **xs:decimal+** XP3.1 XQ3.1

Parst das bereitgestellte `GeolocationInputString`-Argument und gibt die geografische Breite und Länge (in dieser Reihenfolge) als Sequenz aus zwei `xs:decimal` Elementen zurück. Die Formate, in denen der Input-String für die geografische Position bereitgestellt werden kann, sind unten aufgelistet.

Anmerkung: Zur Bereitstellung des Input-String für die geografische Position können die Funktion `image-exif-data` 569 und das `@Geolocation` 569-Attribut der Exif-Metadaten verwendet werden (*siehe Beispiel unten*).

Beispiele

- **altova:parse-geolocation**("33.33 -22.22") gibt die Sequenz bestehend aus zwei `xs:decimals` (33.33, 22.22) Elementen zurück
- **altova:parse-geolocation**("48°51'29.6""N 24°17'40.2""") gibt die Sequenz bestehend aus zwei `xs:decimals` (48.8582222222222, 24.2945) Elementen zurück
- **altova:parse-geolocation**('48°51''29.6"N 24°17''40.2"') gibt die Sequenz bestehend aus zwei `xs:decimals` (48.8582222222222, 24.2945) Elementen zurück
- **altova:parse-geolocation**(**image-exif-data**(//MyImages/Image20141130.01)/**@Geolocation**) gibt die Sequenz bestehend aus zwei `xs:decimals` Elementen zurück

Input-String-Formate der Standortdaten:

Der Input-String für die geografische Position muss die Breite und Länge (in dieser Reihenfolge) getrennt durch ein Leerzeichen enthalten. Beide Werte können jedes der folgenden Formate haben. Auch Kombinationen sind zulässig, d.h. die Breite kann in einem anderen Format als die Länge angegeben werden. Die Breitenwerte liegen im Bereich `+90` bis `-90` (`N` nach `S`). Die Längenwerte liegen im Bereich von `+180` bis `-180` (`O` nach `W`).

Anmerkung: Wenn als Trennzeichen für den Input-String einfache oder doppelte Anführungszeichen verwendet werden, kann dies zu einer Fehlinterpretation der einfachen bzw. doppelten Anführungszeichen als Minuten- bzw. Sekundenwerte führen. In solchen Fällen müssen die zur Angabe der Minuten- und Sekundenwerte verwendeten Anführungszeichen durch Verdoppelung mit einem Escape-Zeichen versehen werden. In den Beispielen in diesem Abschnitt sind Anführungszeichen, die als Trennzeichen für den Input-String dienen, gelb markiert (**"**), während Maßeinheitsangaben blau (**""**) markiert sind.

- Grad, Minuten, Dezimalsekunden, mit nachgestellter Orientierung (`N/S, O/W`)
  D°M'S.SS"N/S D°M'S.SS"W/E
  Beispiel: **33°55'11.11"N 22°44'55.25"W**

- Grad, Minuten, Dezimalsekunden mit Vorzeichen (`+/-`); das Plus-Zeichen für (**N/O**) ist optional

+/-D°M'S.SS" +/-D°M'S.SS"
Beispiel: **33°55'11.11" -22°44'55.25"**

- Grad, Dezimalminuten mit nachgestellter Orientierung (N/S, O/W)
  D°M.MM'N/S D°M.MM'W/E
  Beispiel: **33°55.55'N 22°44.44'W**

- Grad, Dezimalminuten mit Vorzeichen (+/-); das Plus-Zeichen für (**N/O**) ist optional
  +/-D°M.MM' +/-D°M.MM'
  Beispiel: **+33°55.55' -22°44.44'**

- Dezimalgrade, mit nachgestellter Orientierung (N/S, O/W)
  D.DDN/S D.DDW/E
  Beispiel: **33.33N 22.22W**

- Dezimalgrade mit Vorzeichen (+/-); das Plus-Zeichen für (**N/S O/W**) ist optional
  +/-D.DD +/-D.DD
  Beispiel: **33.33 -22.22**

Beispiele für Formatkombinationen:
33.33N -22°44'55.25"
33.33 22°44'55.25"W
33.33 22.45

Altova Exif-Attribut: Geolocation

Der Altova XPath/XQuery-Prozessor generiert anhand der Exif-Standard-Metadaten-Tags das benutzerdefinierte Attribut **Geolocation**. **Geolocation** ist eine Verkettung von vier Exif-Tags: GPSLatitude, GPSLatitudeRef, GPSLongitude, GPSLongitudeRef mit hinzugefügten Werten (*siehe Tabelle unten*).

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 *als xs:string*, **GeolocationInputString-2** *als xs:string*) als **xs:decimal** XP3.1 XQ3.1
Berechnet die Entfernung zwischen zwei geografischen Positionen in Kilometern. Die Formate, in denen der Input-String für die geografischen Position angegeben werden kann, sind unten aufgelistet. Die Breitenwerte liegen im Bereich von +90 bis -90 (N nach S). Die Längenwerte liegen im Bereich von +180 bis -180 (O nach W).

Anmerkung: Zur Bereitstellung des Input-String für die geografische Position können die Funktion image-exif-data 569 und das @Geolocation 569-Attribut der Exif-Metadaten verwendet werden.

Beispiele

 © 2019-2025 Altova GmbH

- **altova:geolocation-distance-km**("33.33 -22.22", "48°51'29.6""N 24°17'40.2""") gibt xs:decimal 4183.08132372392 zurück

Input-String-Formate der Standortdaten:

Der Input-String für die geografische Position muss die Breite und Länge (in dieser Reihenfolge) getrennt durch ein Leerzeichen enthalten. Beide Werte können jedes der folgenden Formate haben. Auch Kombinationen sind zulässig, d.h. die Breite kann in einem anderen Format als die Länge angegeben werden. Die Breitenwerte liegen im Bereich +90 bis -90 (N nach S). Die Längenwerte liegen im Bereich von +180 bis -180 (O nach W).

Anmerkung: Wenn als Trennzeichen für den Input-String einfache oder doppelte Anführungszeichen verwendet werden, kann dies zu einer Fehlinterpretation der einfachen bzw. doppelten Anführungszeichen als Minuten- bzw. Sekundenwerte führen. In solchen Fällen müssen die zur Angabe der Minuten- und Sekundenwerte verwendeten Anführungszeichen durch Verdoppelung mit einem Escape-Zeichen versehen werden. In den Beispielen in diesem Abschnitt sind Anführungszeichen, die als Trennzeichen für den Input-String dienen, gelb markiert (**"**), während Maßeinheitsangaben blau (**""**) markiert sind.

- Grad, Minuten, Dezimalsekunden, mit nachgestellter Orientierung (N/S, O/W)
  D°M'S.SS"N/S D°M'S.SS"W/E
  Beispiel: **33°55'11.11"N 22°44'55.25"W**

- Grad, Minuten, Dezimalsekunden mit Vorzeichen (+/-); das Plus-Zeichen für (**N/O**) ist optional
  +/-D°M'S.SS" +/-D°M'S.SS"
  Beispiel: **33°55'11.11" -22°44'55.25"**

- Grad, Dezimalminuten mit nachgestellter Orientierung (N/S, O/W)
  D°M.MM'N/S D°M.MM'W/E
  Beispiel: **33°55.55'N 22°44.44'W**

- Grad, Dezimalminuten mit Vorzeichen (+/-); das Plus-Zeichen für (**N/O**) ist optional
  +/-D°M.MM' +/-D°M.MM'
  Beispiel: **+33°55.55' -22°44.44'**

- Dezimalgrade, mit nachgestellter Orientierung (N/S, O/W)
  D.DDN/S D.DDW/E
  Beispiel: **33.33N 22.22W**

- Dezimalgrade mit Vorzeichen (+/-); das Plus-Zeichen für (**N/S O/W**) ist optional
  +/-D.DD +/-D.DD
  Beispiel: **33.33 -22.22**

Beispiele für Formatkombinationen:
33.33N -22°44'55.25"
33.33 22°44'55.25"W
33.33 22.45

Altova Exif-Attribut: Geolocation

Der Altova XPath/XQuery-Prozessor generiert anhand der Exif-Standard-Metadaten-Tags das

benutzerdefinierte Attribut **Geolocation**. **Geolocation** ist eine Verkettung von vier Exif-Tags: GPSLatitude, GPSLatitudeRef, GPSLongitude, GPSLongitudeRef mit hinzugefügten Werten (*siehe Tabelle unten*).

GPSLatitude	GPSLatitudeRef	GPSLongitude	GPSLongitudeRef	Geolocation
33 51 21.91	S	151 13 11.73	E	33°51'21.91"S 151° 13'11.73"E

geolocation-distance-mi [altova:]

altova:geolocation-distance-mi(GeolocationInputString-1 *als xs:string*, **GeolocationInputString-2** *als xs:string*) als **xs:decimal** XP3.1 XQ3.1

Berechnet die Entfernung zwischen zwei geografischen Positionen in Meilen. Die Formate, in denen der Input-String für die geografischen Position angegeben werden kann, sind unten aufgelistet. Die Breitenwerte liegen im Bereich von +90 bis -90 (N nach S). Die Längenwerte liegen im Bereich von +180 bis -180 (O nach W).

Anmerkung: Zur Bereitstellung des Input-String für die geografische Position können die Funktion image-exif-data 569 und das @Geolocation 569-Attribut der Exif-Metadaten verwendet werden.

Beispiele

- **altova:geolocation-distance-mi**("33.33 -22.22", "48°51'29.6""N 24°17'40.2""") gibt xs:decimal 2599.40652340653 zurück

Input-String-Formate der Standortdaten:

Der Input-String für die geografische Position muss die Breite und Länge (in dieser Reihenfolge) getrennt durch ein Leerzeichen enthalten. Beide Werte können jedes der folgenden Formate haben. Auch Kombinationen sind zulässig, d.h. die Breite kann in einem anderen Format als die Länge angegeben werden. Die Breitenwerte liegen im Bereich +90 bis -90 (N nach S). Die Längenwerte liegen im Bereich von +180 bis -180 (O nach W).

Anmerkung: Wenn als Trennzeichen für den Input-String einfache oder doppelte Anführungszeichen verwendet werden, kann dies zu einer Fehlinterpretation der einfachen bzw. doppelten Anführungszeichen als Minuten- bzw. Sekundenwerte führen. In solchen Fällen müssen die zur Angabe der Minuten- und Sekundenwerte verwendeten Anführungszeichen durch Verdoppelung mit einem Escape-Zeichen versehen werden. In den Beispielen in diesem Abschnitt sind Anführungszeichen, die als Trennzeichen für den Input-String dienen, gelb markiert (**"**), während Maßeinheitsangaben blau (**""**) markiert sind.

- Grad, Minuten, Dezimalsekunden, mit nachgestellter Orientierung (N/S, O/W)
  D°M'S.SS"N/S D°M'S.SS"W/E
  Beispiel: **33°55'11.11"N 22°44'55.25"W**

- Grad, Minuten, Dezimalsekunden mit Vorzeichen (+/-); das Plus-Zeichen für (**N/O**) ist optional

© 2019-2025 Altova GmbH

+/-D°M'S.SS" +/-D°M'S.SS"
Beispiel: **33°55'11.11" -22°44'55.25"**

- Grad, Dezimalminuten mit nachgestellter Orientierung (N/S, O/W)
  D°M.MM'N/S D°M.MM'W/E
  Beispiel: **33°55.55'N 22°44.44'W**

- Grad, Dezimalminuten mit Vorzeichen (+/-); das Plus-Zeichen für (**N/O**) ist optional
  +/-D°M.MM' +/-D°M.MM'
  Beispiel: **+33°55.55' -22°44.44'**

- Dezimalgrade, mit nachgestellter Orientierung (N/S, O/W)
  D.DDN/S D.DDW/E
  Beispiel: **33.33N 22.22W**

- Dezimalgrade mit Vorzeichen (+/-); das Plus-Zeichen für (**N/S O/W**) ist optional
  +/-D.DD +/-D.DD
  Beispiel: **33.33 -22.22**

Beispiele für Formatkombinationen:
33.33N -22°44'55.25"
33.33 22°44'55.25"W
33.33 22.45

Altova Exif-Attribut: Geolocation

Der Altova XPath/XQuery-Prozessor generiert anhand der Exif-Standard-Metadaten-Tags das benutzerdefinierte Attribut **Geolocation**. **Geolocation** ist eine Verkettung von vier Exif-Tags: GPSLatitude, GPSLatitudeRef, GPSLongitude, GPSLongitudeRef mit hinzugefügten Werten (*siehe Tabelle unten*).

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

geolocations-bounding-rectangle [altova:]

altova:geolocations-bounding-rectangle(Geolocations *als xs:sequence*, **GeolocationOutputStringFormat** *als xs:integer*) als **xs:string** XP3.1 XQ3.1
Erhält als erstes Argument eine Sequenz von Strings, wobei es sich bei jedem String in der Sequenz um eine geografische Position handelt. Die Funktion gibt eine Sequenz von zwei Strings zurück, die die geografischen Positionskoordinaten der linken oberen bzw. rechten unteren Ecke eines Rechtecks bilden, dessen Größe so angepasst ist, dass es alle im ersten Argument angegebenen Positionskoordinaten enthält. Die Formate, in denen der Input-String für die geografischen Position angegeben werden kann, sind unten aufgelistet (*siehe 'Input-String-Formate der Standortdaten'*). Die Breitenwerte liegen im Bereich von +90 bis -90 (N nach S). Die Längenwerte liegen im Bereich von +180 bis -180 (O nach W).

Im zweiten Argument der Funktion ist das Format der beiden Geolocation-Strings in der Ausgabesequenz angegeben. Das Argument erhält einen Ganzzahlwert von `1` bis `4`, wobei die einzelnen Werte ein jeweils unterschiedliches String-Format definieren (*siehe 'Ausgabestringsformate für die geografische Position' weiter unten).*

Anmerkung: Zur Bereitstellung der Input-Strings können die Funktion `image-exif-data` 569 und die Attribute der Exif-Metadaten verwendet werden.

Beispiele

- **altova:geolocations-bounding-rectangle**(("48.2143531 16.3707266", "51.50939 -0.11832"), 1) gibt die Sequenz ("51°30'33.804"N 0°7'5.952"W", "48°12'51.67116"N 16°22'14.61576"E")zurück.
- **altova:geolocations-bounding-rectangle**(("48.2143531 16.3707266", "51.50939 -0.11832", "42.5584577 -70.8893334"), 4) gibt die Sequenz ("51.50939 -70.8893334", "42.5584577 16.3707266") zurück.

Input-String-Formate der Standortdaten:

Der Input-String für die geografische Position muss die Breite und Länge (in dieser Reihenfolge) getrennt durch ein Leerzeichen enthalten. Beide Werte können jedes der folgenden Formate haben. Auch Kombinationen sind zulässig, d.h. die Breite kann in einem anderen Format als die Länge angegeben werden. Die Breitenwerte liegen im Bereich `+90` bis `-90` (`N` nach `S`). Die Längenwerte liegen im Bereich von `+180` bis `-180` (`O` nach `W`).

Anmerkung: Wenn als Trennzeichen für den Input-String einfache oder doppelte Anführungszeichen verwendet werden, kann dies zu einer Fehlinterpretation der einfachen bzw. doppelten Anführungszeichen als Minuten- bzw. Sekundenwerte führen. In solchen Fällen müssen die zur Angabe der Minuten- und Sekundenwerte verwendeten Anführungszeichen durch Verdoppelung mit einem Escape-Zeichen versehen werden. In den Beispielen in diesem Abschnitt sind Anführungszeichen, die als Trennzeichen für den Input-String dienen, gelb markiert (**"**), während Maßeinheitsangaben blau (**""**) markiert sind.

- Grad, Minuten, Dezimalsekunden, mit nachgestellter Orientierung (`N/S, O/W`)
  D°M'S.SS"N/S D°M'S.SS"W/E
  Beispiel: **33°55'11.11"N 22°44'55.25"W**

- Grad, Minuten, Dezimalsekunden mit Vorzeichen (`+/-`); das Plus-Zeichen für (**N/O**) ist optional
  +/-D°M'S.SS" +/-D°M'S.SS"
  Beispiel: **33°55'11.11" -22°44'55.25"**

- Grad, Dezimalminuten mit nachgestellter Orientierung (`N/S`, `O/W`)
  D°M.MM'N/S D°M.MM'W/E
  Beispiel: **33°55.55'N 22°44.44'W**

- Grad, Dezimalminuten mit Vorzeichen (`+/-`); das Plus-Zeichen für (**N/O**) ist optional
  +/-D°M.MM' +/-D°M.MM'
  Beispiel: **+33°55.55' -22°44.44'**

- Dezimalgrade, mit nachgestellter Orientierung (`N/S`, `O/W`)
  D.DDN/S D.DDW/E

 © 2019-2025 Altova GmbH

Beispiel: **33.33N 22.22W**

- Dezimalgrade mit Vorzeichen (+/-); das Plus-Zeichen für (**N/S O/W**) ist optional
  +/-D.DD +/-D.DD
  Beispiel: **33.33 -22.22**

Beispiele für Formatkombinationen:
33.33N -22°44'55.25"
33.33 22°44'55.25"W
33.33 22.45

Ausgabestringformate für die geografische Position:

Die bereitgestellte Breite und Länge ist in einem der unten aufgelisteten Ausgabeformate formatiert. Das gewünschte Format wird anhand seiner Ganzzahl-ID (1 bis 4) identifiziert. Die Breitenwerte liegen im Bereich von +90 bis -90 (N nach S). Die Längenwerte liegen im Bereich von +180 bis -180 (O nach W).

1
Grad, Minuten, Dezimalsekunden, mit nachgestellter Orientierung (N/S, E/W) **D°M'S.SS"N/S D°M'S.SS"E/W** *Beispiel:* **33°55'11.11"N 22°44'66.66"W**

2
Dezimalgrad, mit nachgestellter Orientierung (N/S, E/W) **D.DDN/S D.DDE/W** *Beispiel:* **33.33N 22.22W**

3
Grad, Minuten, Dezimalsekunden mit Vorzeichen (+/-); Plus-Zeichen für (**N/E**) ist optional **+/-D°M'S.SS" +/-D°M'S.SS"** *Beispiel:* **33°55'11.11" -22°44'66.66"**

4
Dezimalgrad, mit Vorzeichen (+/-); Plus-Zeichen für (**N/E**) ist optional **+/-D.DD +/-D.DD** *Beispiel:* **33.33 -22.22**

Altova Exif-Attribut: Geolocation

Der Altova XPath/XQuery-Prozessor generiert anhand der Exif-Standard-Metadaten-Tags das benutzerdefinierte Attribut **Geolocation**. **Geolocation** ist eine Verkettung von vier Exif-Tags: GPSLatitude, GPSLatitudeRef, GPSLongitude, GPSLongitudeRef mit hinzugefügten Werten (*siehe Tabelle unten*).

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 *als xs:string*, **((PolygonPoint** *als xs:string*)**+))** als **xs:boolean** XP3.1 XQ3.1

Ermittelt ob sich **Geolocation** (das erste Argument) innerhalb des durch die **PolygonPoint**-Argumente beschriebenen Polygonbereichs befindet. Wenn die `PolygonPoint`-Argumente keine geschlossene Form (wenn der erste und der letzte Punkt identisch sind) bilden, so wird der erste Punkt implizit zum letzten Punkt hinzugefügt, um die Form zu schließen. Alle Argumente (`Geolocation` und `PolygonPoint+`) werden durch Input-Strings für die geografische Position (*Formatliste siehe unten*) angegeben. Wenn sich das `Geolocation` Argument innerhalb des Polygons befindet, gibt die Funktion `true()`zurück; andernfalls gibt sie `false()` zurück. Die Breitenwerte liegen im Bereich von `+90` bis `-90` (`N` nach `S`). Die Längenwerte liegen im Bereich von `+180` bis `-180` (`O` nach `W`).

Anmerkung: Zur Bereitstellung des Input-String für die geografische Position können die Funktion `image-exif-data` 569 und das `@Geolocation` 569-Attribut der Exif-Metadaten verwendet werden.

Beispiele

- **altova:geolocation-within-polygon(**"33 -22", ("58 -32", "-78 -55", "48 24", "58 -32")**)** gibt `true()` zurück
- **altova:geolocation-within-polygon(**"33 -22", ("58 -32", "-78 -55", "48 24")**)** gibt `true()` zurück
- **altova:geolocation-within-polygon(**"33 -22", ("58 -32", "-78 -55", "48°51'29.6""N 24°17'40.2""")**)** gibt `true()` zurück

Input-String-Formate der Standortdaten:

Der Input-String für die geografische Position muss die Breite und Länge (in dieser Reihenfolge) getrennt durch ein Leerzeichen enthalten. Beide Werte können jedes der folgenden Formate haben. Auch Kombinationen sind zulässig, d.h. die Breite kann in einem anderen Format als die Länge angegeben werden. Die Breitenwerte liegen im Bereich `+90` bis `-90` (`N` nach `S`). Die Längenwerte liegen im Bereich von `+180` bis `-180` (`O` nach `W`).

Anmerkung: Wenn als Trennzeichen für den Input-String einfache oder doppelte Anführungszeichen verwendet werden, kann dies zu einer Fehlinterpretation der einfachen bzw. doppelten Anführungszeichen als Minuten- bzw. Sekundenwerte führen. In solchen Fällen müssen die zur Angabe der Minuten- und Sekundenwerte verwendeten Anführungszeichen durch Verdoppelung mit einem Escape-Zeichen versehen werden. In den Beispielen in diesem Abschnitt sind Anführungszeichen, die als Trennzeichen für den Input-String dienen, gelb markiert (`"`), während Maßeinheitsangaben blau (`""`) markiert sind.

- Grad, Minuten, Dezimalsekunden, mit nachgestellter Orientierung (`N/S, O/W`)
  D°M'S.SS"N/S D°M'S.SS"W/E
  Beispiel: **33°55'11.11"N 22°44'55.25"W**

 © 2019-2025 Altova GmbH

- Grad, Minuten, Dezimalsekunden mit Vorzeichen (`+/-`); das Plus-Zeichen für (**`N/O`**) ist optional
  `+/-D°M'S.SS"  +/-D°M'S.SS"`
  Beispiel: **`33°55'11.11"  -22°44'55.25"`**

- Grad, Dezimalminuten mit nachgestellter Orientierung (`N/S`, `O/W`)
  `D°M.MM'N/S  D°M.MM'W/E`
  Beispiel: **`33°55.55'N  22°44.44'W`**

- Grad, Dezimalminuten mit Vorzeichen (`+/-`); das Plus-Zeichen für (**`N/O`**) ist optional
  `+/-D°M.MM'  +/-D°M.MM'`
  Beispiel: **`+33°55.55'  -22°44.44'`**

- Dezimalgrade, mit nachgestellter Orientierung (`N/S`, `O/W`)
  `D.DDN/S  D.DDW/E`
  Beispiel: **`33.33N  22.22W`**

- Dezimalgrade mit Vorzeichen (`+/-`); das Plus-Zeichen für (**`N/S O/W`**) ist optional
  `+/-D.DD  +/-D.DD`
  Beispiel: **`33.33  -22.22`**

Beispiele für Formatkombinationen:
`33.33N  -22°44'55.25"`
`33.33  22°44'55.25"W`
`33.33  22.45`

Altova Exif-Attribut: Geolocation

Der Altova XPath/XQuery-Prozessor generiert anhand der Exif-Standard-Metadaten-Tags das benutzerdefinierte Attribut **`Geolocation`**. **`Geolocation`** ist eine Verkettung von vier Exif-Tags: `GPSLatitude`, `GPSLatitudeRef`, `GPSLongitude`, `GPSLongitudeRef` mit hinzugefügten Werten (*siehe Tabelle unten*).

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` *`als xs:string`*, **`RectCorner-1`** *`als xs:string`*, **`RectCorner-2`** *`als xs:string`*) als **`xs:boolean`** **XP3.1** **XQ3.1**
Ermittelt, ob sich **`Geolocation`** (das erste Argument) innerhalb des durch das zweite und dritte Argument, **`RectCorner-1`** und **`RectCorner-2`**, definierten Rechtecks befindet. **`RectCorner-1`** und **`RectCorner-2`** definieren gegenüberliegende Eckpunkte des Rechtecks. Alle Argumente (`Geolocation`, **`RectCorner-1`** und **`RectCorner-2`**) werden durch Input-Strings für die geografische Position (*Formatliste siehe unten*) angegeben. Wenn sich das `Geolocation`-Argument innerhalb des Rechtecks befindet, gibt die Funktion `true()` zurück; andernfalls gibt sie `false()` zurück. Die Breitenwerte liegen im Bereich von `+90` bis `-90` (`N`

nach S). Die Längenwerte liegen im Bereich von +180 bis -180 (O nach W).

Anmerkung: Zur Bereitstellung des Input-String für die geografische Position können die Funktion image-exif-data 569 und das @Geolocation 569-Attribut der Exif-Metadaten verwendet werden.

Beispiele

- **altova:geolocation-within-rectangle(**"33 -22", "58 -32", "-48 24") gibt true() zurück
- **altova:geolocation-within-rectangle(**"33 -22", "58 -32", "48 24") gibt false() zurück
- **altova:geolocation-within-rectangle(**"33 -22", "58 -32", "48°51'29.6""S 24° 17'40.2""") gibt true() zurück

Input-String-Formate der Standortdaten:

Der Input-String für die geografische Position muss die Breite und Länge (in dieser Reihenfolge) getrennt durch ein Leerzeichen enthalten. Beide Werte können jedes der folgenden Formate haben. Auch Kombinationen sind zulässig, d.h. die Breite kann in einem anderen Format als die Länge angegeben werden. Die Breitenwerte liegen im Bereich +90 bis -90 (N nach S). Die Längenwerte liegen im Bereich von +180 bis -180 (O nach W).

Anmerkung: Wenn als Trennzeichen für den Input-String einfache oder doppelte Anführungszeichen verwendet werden, kann dies zu einer Fehlinterpretation der einfachen bzw. doppelten Anführungszeichen als Minuten- bzw. Sekundenwerte führen. In solchen Fällen müssen die zur Angabe der Minuten- und Sekundenwerte verwendeten Anführungszeichen durch Verdoppelung mit einem Escape-Zeichen versehen werden. In den Beispielen in diesem Abschnitt sind Anführungszeichen, die als Trennzeichen für den Input-String dienen, gelb markiert (**"**), während Maßeinheitsangaben blau (**""**) markiert sind.

- Grad, Minuten, Dezimalsekunden, mit nachgestellter Orientierung (N/S, O/W)
  D°M'S.SS"N/S D°M'S.SS"W/E
  Beispiel: **33°55'11.11"N 22°44'55.25"W**

- Grad, Minuten, Dezimalsekunden mit Vorzeichen (+/-); das Plus-Zeichen für (**N/O**) ist optional
  +/-D°M'S.SS" +/-D°M'S.SS"
  Beispiel: **33°55'11.11" -22°44'55.25"**

- Grad, Dezimalminuten mit nachgestellter Orientierung (N/S, O/W)
  D°M.MM'N/S D°M.MM'W/E
  Beispiel: **33°55.55'N 22°44.44'W**

- Grad, Dezimalminuten mit Vorzeichen (+/-); das Plus-Zeichen für (**N/O**) ist optional
  +/-D°M.MM' +/-D°M.MM'
  Beispiel: **+33°55.55' -22°44.44'**

- Dezimalgrade, mit nachgestellter Orientierung (N/S, O/W)
  D.DDN/S D.DDW/E
  Beispiel: **33.33N 22.22W**

- Dezimalgrade mit Vorzeichen (+/-); das Plus-Zeichen für (**N/S O/W**) ist optional
  +/-D.DD +/-D.DD

© 2019-2025 Altova GmbH

Beispiel: **33.33 -22.22**

Beispiele für Formatkombinationen:
33.33N -22°44'55.25"
33.33 22°44'55.25"W
33.33 22.45

Altova Exif-Attribut: Geolocation

Der Altova XPath/XQuery-Prozessor generiert anhand der Exif-Standard-Metadaten-Tags das benutzerdefinierte Attribut **Geolocation**. **Geolocation** ist eine Verkettung von vier Exif-Tags: `GPSLatitude`, `GPSLatitudeRef`, `GPSLongitude`, `GPSLongitudeRef` mit hinzugefügten Werten (*siehe Tabelle unten*).

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

[Nach oben 557]

12.2.2.1.4 XPath/XQuery-Funktionen: Bildbezogene

Die folgenden XPath/XQuery-Erweiterungsfunktionen im Zusammenhang mit Bildern werden in der aktuellen Version von MapForce unterstützt und können in (i) in einem XSLT-Kontext in XPath-Ausdrücken oder (ii) in einem XQuery-Dokument in einem XQuery-Ausdruck verwendet werden.

Anmerkung zur Benennung von Funktionen und zur Anwendbarkeit der Sprache

Altova-Erweiterungsfunktionen können in XPath/XQuery-Ausdrücken verwendet werden. Dadurch stehen neben den Funktionen in der Standardbibliothek der XPath-, XQuery- und XSLT-Funktionen zusätzliche Funktionen zur Verfügung. Die Altova-Erweiterungsfunktionen befinden sich im **Altova-Erweiterungsfunktions-Namespace,** **`http://www.altova.com/xslt-extensions`** und sind in diesem Abschnitt mit dem Präfix **`altova:`**, das als an diesen Namespace gebunden angenommen wird, gekennzeichnet. Beachten Sie, dass manche Funktionen in zukünftigen Versionen Ihres Produkts eventuell nicht mehr unterstützt werden oder dass sich das Verhalten einzelner Funktionen ändern kann. Um zu sehen, welche Altova-Erweiterungsfunktionen unterstützt werden, lesen Sie bitte die Dokumentation zur jeweiligen Release.

XPath-Funktionen (in XPath-Ausdrücken in XSLT verwendet):	XP1 XP2 XP3.1
XSLT-Funktionen (in XPath-Ausdrücken in XSLT verwendet):	XSLT1 XSLT2 XSLT3
XQuery-Funktionen (in XQuery-Ausdrücken in XQuery verwendet):	XQ1 XQ3.1

suggested-image-file-extension [altova:]

altova:suggested-image-file-extension(Base64String *als string***)** als **string?** XP3.1 XQ3.1
Erhält die Base64-Kodierung einer Bilddatei als Argument und gibt die darin enthaltene Dateierweiterung des Bilds zurück. Der Rückgabewert ist ein Vorschlag, basierend auf den in der Kodierung enthaltenen Bilddateitypinformationen. Wenn diese Informationen nicht verfügbar sind, wird ein leerer String zurückgegeben. Diese Funktion ist nützlich, wenn Sie ein Base64-Bild als Datei speichern und die entsprechende Dateierweiterung dynamisch abrufen möchten.

Beispiele

- **altova:suggested-image-file-extension**(/MyImages/MobilePhone/Image20141130.01) gibt 'jpg' zurück
- **altova:suggested-image-file-extension**($XML1/Staff/Person/@photo) gibt '' zurück

In den Beispielen oben wird von den als Argument der Funktion bereitgestellten Nodes angenommen, dass sie ein Base64-kodiertes Bild enthalten. Im ersten Beispiel wird jpg als Dateityp bzw. Dateierweiterung abgerufen. Im zweiten Beispiel enthält die angegebene Base54-Kodierung keine brauchbaren Dateierweiterungsinformationen.

image-exif-data [altova:]

altova:image-exif-data(Base64BinaryString *als string***)** als **element?** XP3.1 XQ3.1
Erhält ein Base64-kodiertes JPEG-Bild als Argument und gibt ein Element namens **Exif** zurück, das die Exif-Metadaten des Bilds enthält. Die Exif-Metadaten werden als Attribut-Wert-Paare des Exif-Elements erstellt. Bei den Attributnamen handelt es sich um die Exif-Daten-Tags aus der Base64-Kodierung. Weiter unten sehen Sie eine Liste der Exif-Tags. Wenn die Exif-Daten einen anbieterspezifischen Tag enthalten, so wird auch dieser Tag und sein Wert als Attribut-Wert-Paar zurückgegeben. Zusätzlich zu den Standard-Exif-Metadatentags (*siehe Liste unten*) werden auch Altova-spezifische Attribut-Wert-Paare generiert. Diese Altova Exif-Attribute sind unten aufgelistet.

Beispiele

- Um ein einziges Attribut abzurufen, verwenden Sie die Funktion folgendermaßen:
  image-exif-data(//MyImages/Image20141130.01)/**@GPSLatitude**
  image-exif-data(//MyImages/Image20141130.01)/**@Geolocation**
- Um alle Attribute abzurufen, verwenden Sie die Funktion folgendermaßen:
  image-exif-data(//MyImages/Image20141130.01)/**@***
- Um die Namen aller Attribute abzurufen, verwenden Sie den folgenden Ausdruck:
  for $i in image-exif-data**(//MyImages/Image20141130.01)/**@* return name($i)
  Auf diese Art können Sie die Namen der von der Funktion zurückgegebenen Attribute eruieren.

Altova Exif-Attribut: Geolocation

Der Altova XPath/XQuery-Prozessor generiert anhand der Exif-Standard-Metadaten-Tags das benutzerdefinierte Attribut **Geolocation**. **Geolocation** ist eine Verkettung von vier Exif-Tags: GPSLatitude, GPSLatitudeRef, GPSLongitude, GPSLongitudeRef mit hinzugefügten Werten (*siehe Tabelle unten*).

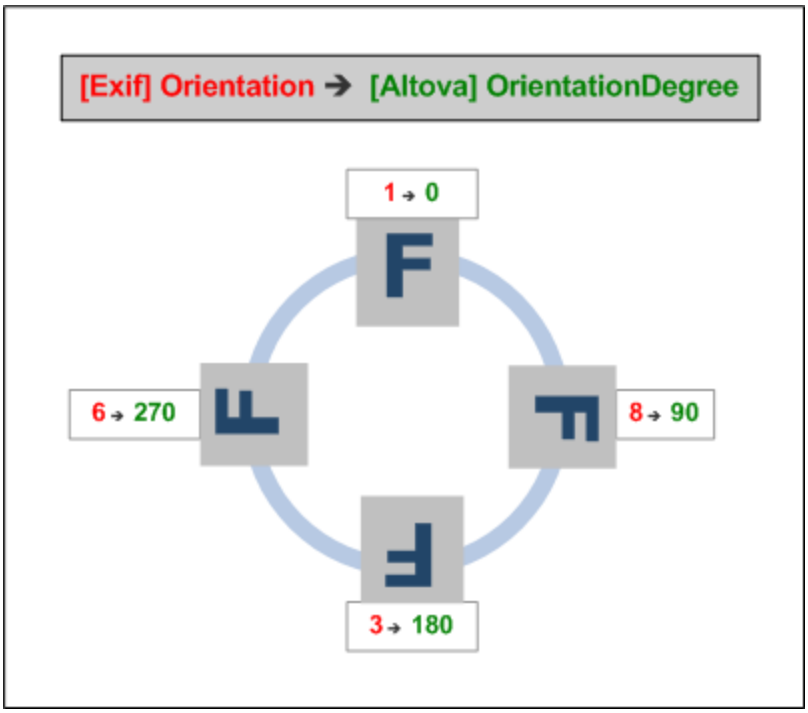
GPSLatitude	GPSLatitudeRef	GPSLongitude	GPSLongitudeRef	Geolocation

33 51 21.91	S	151 13 11.73	E	33°51'21.91"S 151° 13'11.73"E

Altova Exif-Attribut: OrientationDegree

Der Altova XPath/XQuery-Prozessor generiert anhand des Exif-Metadaten-Tags **`Orientation`** das benutzerdefinierte Attribut **`OrientationDegree`**.

`OrientationDegree` übersetzt den Standard-Exif-Tag `Orientation` von einem Ganzzahlwert (`1`, `8`, `3` oder `6`) in die entsprechenden Gradwerte dafür (`0`, `90`, `180`, `270`) (*siehe Abbildung unten*). Beachten Sie dass es keine Übersetzung der `Orientation`-Werte `2`, `4`, `5`, `7` gibt. (Diese Ausrichtungen werden durch Spiegelung des Bilds `1` an seiner senkrechten Mittelachse zur Erzeugung des Bilds mit dem Wert `2` und anschließende Drehung dieses Bilds um jeweils 90 Grad zur Erzeugung der Werte `7` bzw. `4` bzw. `5` erzielt).



Liste der Standard-Exif-Metatags

- `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

© 2019-2025 Altova GmbH

- 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

[Nach oben 569]

12.2.2.1.5 XPath/XQuery-Funktionen: Numerische

Die numerischen Erweiterungsfunktionen von Altova können in XPath- und XQuery-Ausdrücken verwendet werden und stellen zusätzliche Funktionen für die Verarbeitung von Daten zur Verfügung. Die Funktionen in diesem Abschnitt können mit dem **XPath 3.0- und XQuery 3.0**-Prozessor von Altova verwendet werden. Sie stehen im Zusammenhang mit XPath/XQuery zur Verfügung.

Anmerkung zur Benennung von Funktionen und zur Anwendbarkeit der Sprache

Altova-Erweiterungsfunktionen können in XPath/XQuery-Ausdrücken verwendet werden. Dadurch stehen neben den Funktionen in der Standardbibliothek der XPath-, XQuery- und XSLT-Funktionen zusätzliche Funktionen zur Verfügung. Die Altova-Erweiterungsfunktionen befinden sich im **Altova-Erweiterungsfunktions-Namespace, `http://www.altova.com/xslt-extensions`** und sind in diesem Abschnitt mit dem Präfix **`altova:`**, das als an diesen Namespace gebunden angenommen wird, gekennzeichnet. Beachten Sie, dass manche Funktionen in zukünftigen Versionen Ihres Produkts eventuell nicht mehr unterstützt werden oder dass sich das Verhalten einzelner Funktionen ändern kann. Um zu sehen, welche Altova-Erweiterungsfunktionen unterstützt werden, lesen Sie bitte die Dokumentation zur jeweiligen Release.

XPath-Funktionen (in XPath-Ausdrücken in XSLT verwendet):	XP1 XP2 XP3.1
XSLT-Funktionen (in XPath-Ausdrücken in XSLT verwendet):	XSLT1 XSLT2 XSLT3
XQuery-Funktionen (in XQuery-Ausdrücken in XQuery verwendet):	XQ1 XQ3.1

Funktionen zur automatischen Nummerierung

generate-auto-number [altova:]

altova:generate-auto-number(ID *als xs:string*, **StartsWith** *als xs:double*, **Increment** *als xs:double*, **ResetOnChange** *als xs:string*) als `xs:integer` XP1 XP2 XQ1 XP3.1 XQ3.1

Generiert jedes Mal, wenn die Funktion aufgerufen wird, eine Zahl. Die erste Zahl, die beim ersten Aufruf der Funktion generiert wird, wird durch das Argument `StartsWith` definiert. Bei jedem erneuten Aufruf der Funktion wird eine neue Zahl generiert. Diese Zahl wird durch den im Argument `Increment` definierten Wert anhand der zuvor generierten Zahl inkrementiert. Auf diese Art erstellt die Funktion `altova:generate-auto-number` einen Zähler, dessen Name durch das Argument `ID` definiert wird und der jedes Mal, wenn die Funktion aufgerufen wird, inkrementiert wird. Wenn sich der Wert des Arguments `ResetOnChange` seit dem vorherigen Funktionsaufruf geändert hat, so wird der Wert der zu generierenden Zahl auf den Wert `StartsWith` zurückgesetzt. Die Automatische Nummerierung kann auch mit der Funktion `altova:reset-auto-number` zurückgesetzt werden.

Beispiele

- `altova:generate-auto-number("ChapterNumber", 1, 1, "SomeString")` gibt bei jedem Aufruf

 © 2019-2025 Altova GmbH

der Funktion eine einzige Zahl beginnend mit `1` zurück, die bei jedem Aufruf der Funktion um `1` inkrementiert wird. Solange das vierte Argument in jedem anschließenden Aufruf `"SomeString"` bleibt, wird die Inkrementierung fortgesetzt. Wenn sich der Wert des vierten Arguments ändert, wird der Zähler (namens `ChapterNumber`) auf 1 zurückgesetzt. Der Wert von `ChapterNumber` kann auch folgendermaßen durch Aufruf der Funktion `altova:reset-auto-number` zurückgesetzt werden: `altova:reset-auto-number("ChapterNumber")`.

reset-auto-number [altova:]

`altova:reset-auto-number(ID` *`als xs:string`*`)` XP1 XP2 XQ1 XP3.1 XQ3.1

Diese Funktion setzt die Zahl des im `ID`-Argument angegebenen Zählers zur automatischen Nummerierung zurück. Die Zahl wird auf die Zahl zurückgesetzt, die durch das Argument `StartsWith` der Funktion `altova:generate-auto-number`, die den im `ID`-Argument genannten Zähler erstellt hat, definiert ist

Beispiele

- **`altova:reset-auto-number`**`("ChapterNumber")` setzt die Zahl des Zählers zur automatischen Nummerierung (`ChapterNumber`), der durch die Funktion `altova:generate-auto-number` erstellt wurde, zurück. Die Zahl wird auf den Wert des Arguments `StartsWith` der Funktion `altova:generate-auto-number`, die `ChapterNumber` erstellt hat, zurückgesetzt.

[Nach oben 574]

Numerische Funktionen

hex-string-to-integer [altova:]

`altova:hex-string-to-integer(HexString` *`als xs:string`*`)` als **`xs:integer`** XP3.1 XQ3.1

Verwendet ein String-Argument, das das Base-16-Äquivalent einer Ganzzahl im Dezimalsystem (Base-10) ist, und gibt die dezimale Ganzzahl zurück.

Beispiele

- **`altova:hex-string-to-integer`**`('1')` gibt `1` zurück
- **`altova:hex-string-to-integer`**`('9')` gibt `9` zurück
- **`altova:hex-string-to-integer`**`('A')` gibt `10` zurück
- **`altova:hex-string-to-integer`**`('B')` gibt `11` zurück
- **`altova:hex-string-to-integer`**`('F')` gibt `15` zurück
- **`altova:hex-string-to-integer`**`('G')` gibt einen Fehler zurück
- **`altova:hex-string-to-integer`**`('10')` gibt `16` zurück
- **`altova:hex-string-to-integer`**`('01')` gibt `1` zurück
- **`altova:hex-string-to-integer`**`('20')` gibt `32` zurück
- **`altova:hex-string-to-integer`**`('21')` gibt `33` zurück
- **`altova:hex-string-to-integer`**`('5A')` gibt `90` zurück
- **`altova:hex-string-to-integer`**`('USA')` gibt einen Fehler zurück

integer-to-hex-string [altova:]

`altova:integer-to-hex-string(Integer` *`as xs:integer`*`)` **`as xs:string`** XP3.1 XQ3.1

Verwendet ein Ganzzahlargument und gibt das Base-16-Äquivalent als String zurück.

Beispiele

- `altova:integer-to-hex-string(1)` gibt `1` zurück
- `altova:integer-to-hex-string(9)` gibt `'9'` zurück
- `altova:integer-to-hex-string(10)` gibt `'A'` zurück
- `altova:integer-to-hex-string(11)` gibt `'B'` zurück
- `altova:integer-to-hex-string(15)` gibt `'F'` zurück
- `altova:integer-to-hex-string(16)` gibt `'10'` zurück
- `altova:integer-to-hex-string(32)` gibt `'20'` zurück
- `altova:integer-to-hex-string(33)` gibt `'21'` zurück
- `altova:integer-to-hex-string(90)` gibt `'5A'` zurück

[Nach oben 574]

[nach oben 574]

12.2.2.1.6 XPath/XQuery-Funktionen: Schema

Die unten aufgelisteten Altova-Erweiterungsfunktionen geben Schemainformationen zurück. Weiter unten finden Sie Beschreibungen der Funktionen zusammen mit (i) Beispielen und (ii) einer Liste von Schemakomponenten und den dazugehörigen Eigenschaften. Diese Funktionen können mit dem **XPath 3.0-** und dem **XQuery 3.0-** Prozessor von Altova verwendet werden und stehen im Zusammenhang mit XPath/XQuery zur Verfügung.

Schemainformationen aus Schema-Dokumenten
Die Funktion **`altova:schema`** hat zwei Argumente: eines mit null Argumenten und das andere mit zwei Argumenten. Die Funktion mit null Argumenten gibt das gesamte Schema zurück. Sie können anschließend von diesem Ausgangspunkt aus durch das Schema navigieren und zu den gewünschten Schemakomponenten gehen. Die Funktion mit zwei Argumenten gibt eine bestimmte, durch Ihren QName identifizierte Komponentenart zurück. Der Rückgabewert ist in beiden Fällen eine Funktion. Um durch die zurückgegebene Komponente zu navigieren, müssen Sie eine Eigenschaft dieser spezifischen Komponente auswählen. Wenn es sich bei der Eigenschaft um ein nicht atomares Element handelt (d.h. wenn es sich um eine Komponente handelt), können Sie weiter navigieren, indem Sie eine Eigenschaft dieser Komponente auswählen. Wenn es sich bei der ausgewählten Eigenschaft um ein atomares Element handelt, wird der Wert des Elements zurückgegeben und Sie können nicht weiter navigieren.

Anmerkung: In XPath-Ausdrücken muss das Schema in die Verarbeitungsumgebung, z.B. in XSLT mit der Anweisung **`xslt:import-schema`** importiert werden. In XQuery-Ausdrücken muss das Schema explizit mit Hilfe eines Schemaimports importiert werden.

Schemainformationen aus XML-Nodes
Die Funktion **`altova:type`** übermittelt den Node eines XML-Dokuments und gibt die Typinformationen des Node aus dem PSVI zurück.

Anmerkung zur Benennung von Funktionen und zur Anwendbarkeit der Sprache

Altova-Erweiterungsfunktionen können in XPath/XQuery-Ausdrücken verwendet werden. Dadurch stehen neben den Funktionen in der Standardbibliothek der XPath-, XQuery- und XSLT-Funktionen zusätzliche Funktionen zur Verfügung. Die Altova-Erweiterungsfunktionen befinden sich im **Altova-**

 © 2019-2025 Altova GmbH

Erweiterungsfunktions-Namespace, `http://www.altova.com/xslt-extensions` und sind in diesem Abschnitt mit dem Präfix `altova:`, das als an diesen Namespace gebunden angenommen wird, gekennzeichnet. Beachten Sie, dass manche Funktionen in zukünftigen Versionen Ihres Produkts eventuell nicht mehr unterstützt werden oder dass sich das Verhalten einzelner Funktionen ändern kann. Um zu sehen, welche Altova-Erweiterungsfunktionen unterstützt werden, lesen Sie bitte die Dokumentation zur jeweiligen Release.

XPath-Funktionen (in XPath-Ausdrücken in XSLT verwendet):	XP1 XP2 XP3.1
XSLT-Funktionen (in XPath-Ausdrücken in XSLT verwendet):	XSLT1 XSLT2 XSLT3
XQuery-Funktionen (in XQuery-Ausdrücken in XQuery verwendet):	XQ1 XQ3.1

Schema (null Argumente)

`altova:schema() als (function(xs:string) als item()*)?` XP3.1 XQ3.1

Gibt die **schema**-Komponente als ganzes zurück. Durch Auswahl einer der Eigenschaften der **schema**-Komponente können Sie weiter in die **schema**-Komponente navigieren.

- Wenn es sich bei dieser Eigenschaft um eine Komponente handelt, können Sie durch Auswahl einer dieser Komponenteneigenschaften einen Schritt tiefer navigieren. Dieser Schritt kann wiederholt werden, um weiter in das Schema zu navigieren.
- Wenn es sich bei der Komponente um einen atomaren Wert handelt, wird der atomare Wert zurückgegeben und Sie können nicht tiefer navigieren.

Die Eigenschaften der **schema**-Komponente sind:

```
"type definitions"
"attribute declarations"
"element declarations"
"attribute group definitions"
"model group definitions"
"notation declarations"
"identity-constraint definitions"
```

Die Eigenschaften aller anderen Komponentenarten (neben **schema**) sind unten aufgelistet.

Anmerkung: In XQuery-Ausdrücken muss das Schema explizit importiert werden. In XPath-Ausdrücken muss das Schema in die Verarbeitungsumgebung, z.B. in XSLT mit der Anweisung **`xslt:import`** importiert worden sein.

Beispiele

- `import schema "" at "C:\Test\ExpReport.xsd"; for $typedef in altova:schema() ("type definitions")`
  `return $typedef ("name")` gibt die Namen aller simpleTypes oder complexTypes im Schema zurück.

- `import schema "" at "C:\Test\ExpReport.xsd";`
  `altova:schema() ("type definitions")[1]("name")` gibt den Namen des ersten aller simpleTypes oder complexTyps im Schema zurück.

Komponenten und ihre Eigenschaften

Assertion

Eigenschaftsname	Eigenschaftstyp	Eigenschaftswert
kind	String	"Assertion"
test	XPath-Eigenschaftsdatensatz	

Attribute Declaration

Eigenschaftsname	Eigenschaftstyp	Eigenschaftswert
kind	String	"Attribute Declaration"
name	String	Lokaler Name des Attributs
target namespace	String	Namespace URI des Attributs
type definition	SimpleType oder ComplexType	
scope	Eine Funktion mit Eigenschaften ("class":"Scope", "variety": "global" oder "local", "parent": der enthaltende ComplexTyp bzw. die enthaltende Attributgruppe)	
value constraint	Falls vorhanden, eine Funktion mit Eigenschaften ("class": "Value Constraint", "variety": "fixed" oder "default", "value": atomarer Wert, "lexical form": String. Beachten Sie, dass die Eigenschaft "value" für Namespace-sensitive Typen nicht zur Verfügung steht.	
inheritable	Boolean	

Attribute Group Declaration

Eigenschaftsname	Eigenschaftstyp	Eigenschaftswert
kind	String	"Attribute Group Definition"
name	String	Lokaler Name der Attributgruppe
target namespace	String	Namespace URI der Attributgruppe
attribute uses	Sequenz von (Attribute Use)	
attribute wildcard	Optionale Attribut-Wildcard	

Attribute Use

Eigenschaftsname	Eigenschaftstyp	Eigenschaftswert

© 2019-2025 Altova GmbH

kind	String	"Attribute Use"
required	Boolean	true, wenn das Attribut obligatorisch ist, false, wenn es optional ist.
value constraint	Siehe Attribute Declaration	
inheritable	Boolean	

Attribute Wildcard

Eigenschaftsname	Eigenschaftstyp	Eigenschaftswert
kind	string	"Wildcard"
namespace constraint	Funktion mit Eigenschaften ("class": "Namespace Constraint", "variety": "any"|"enumeration"|"not", "namespaces": Sequenz von xs:anyURI, "disallowed names": Liste mit QNames und/oder den Strings "defined" und "definedSiblings"	
process contents	String ("strict"|"lax"|"skip")	

Complex Type

Eigenschaftsname	Eigenschaftstyp	Eigenschaftswert
kind	String	"Complex Type"
name	String	Lokaler Name des Typs (leer, wenn anonym)
target namespace	String	Namespace URI des Typs (leer, wenn anonym)
base type definition	Complex Type Definition	
final	String-Sequenz ("restriction"|"extension")	
context	Leere Sequenz (nicht implementiert)	
derivation method	String ("restriction"|"extension")	
abstract	Boolean	
attribute uses	Attribute Use-Sequenz	
attribute wildcard	Optionale Attribute Wildcard	
content type	Funktion mit Eigenschaften: ("class":"Content Type", "variety":string ("element-only"|"empty"|"mixed"|"simple"), particle: optionales Partikel, "open content": Funktion mit Eigenschaften ("class":"Open Content", "mode": string	

	("interleave"|"suffix"), "wildcard": Wildcard), "simple type definition": Simple Type)	
prohibited substitutions	String-Sequenz ("restriction"|"extension")	
assertions	Assertion-Sequenz	

Element Declaration

Eigenschaftsname	Eigenschaftstyp	Eigenschaftswert
kind	String	"Complex Type"
name	String	Lokaler Name des Typs (leer, wenn anonym)
target namespace	String	Namespace URI des Typs (leer, wenn anonym)
type definition	Simple Type oder Complex Type	
type table	Funktion mit Eigenschaften ("class":"Type Table", "alternatives": Type Alternative-Sequenz, "default type definition": Simple Type oder Complex Type)	
scope	Funktion mit Eigenschaften ("class":"Scope", "variety": ("global"|"local"), "parent": optionaler Complex Type)	
value constraint	siehe Attribute Declaration	
nillable	Boolean	
identity-constraint definitions	Identity Constraint-Sequenz	
substitution group affiliations	Element Declaration-Sequenz	
substitution group exclusions	String-Sequenz ("restriction"|"extension")	
disallowed substitutions	String-Sequenz ("restriction"|"extension"|"substitution")	
abstract	Boolean	

Element Wildcard

Eigenschaftsname	Eigenschaftstyp	Eigenschaftswert
kind	String	"Wildcard"
namespace constraint	Funktion mit Eigenschaften ("class": "Namespace Constraint", "variety":	

© 2019-2025 Altova GmbH

	"any"|"enumeration"|"not", "namespaces": xs:anyURI-Sequenz, "disallowed names": Liste mit QNames und/oder den Strings "defined" und "definedSiblings"	
process contents	String ("strict"|"lax"|"skip")	

Facet

Eigenschaftsname	Eigenschaftstyp	Eigenschaftswert
kind	String	Der Name des Facet, z.B. "minLength" oder "enumeration"
value	abhängig vom Facet	Der Wert des Facet
fixed	Boolean	
typed-value	Nur für das Enumeration Facet, Array(xs:anyAtomicType*)	Ein Array, das Enumeration-Werte, von denen jeder im Allgemeinen eine Sequenz atomarer Werte sein kann, enthält. (Anmerkung: Die Eigenschaft "value" ist unabhängig vom tatsächlichen Typ für das Enumeration Facet eine String-Sequenz)

Identity Constraint

Eigenschaftsname	Eigenschaftstyp	Eigenschaftswert
kind	String	"Identity-Constraint Definition"
name	String	Lokaler Name des Constraint
target namespace	String	Namespace URI des Constraint
identity-constraint category	String ("key"|"unique"|"keyRef")	
selector	XPath-Eigenschaftsdatensatz	
fields	Sequenz von XPath-Eigenschaftsdatensätzen	
referenced key	(nur für keyRef): Identity Constraint	Der entsprechende Key Constraint

Model Group

Property name	Eigenschaftstyp	Eigenschaftswert
kind	String	"Model Group"
compositor	String ("sequence"|"choice"|"all")	
particles	Partikel-Sequenz	

Model Group Definition

Eigenschaftsname	Eigenschaftstyp	Eigenschaftswert
kind	String	"Model Group Definition"
name	String	Lokaler Name der Model Group
target namespace	String	Namespace URI der Model Group
model group	Model Group	

Notation

Eigenschaftsname	Eigenschaftstyp	Eigenschaftswert
kind	String	"Notation Declaration"
name	String	Lokaler Name der Notation
target namespace	String	Namespace URI der Notation
system identifier	anyURI	
public identifier	String	

Particle

Eigenschaftsname	Eigenschaftstyp	Eigenschaftswert
kind	String	"Particle"
min occurs	Integer	
max occurs	Integer oder String("unbounded")	
term	Element Declaration, Element Wildcard oder ModelGroup	

Simple Type

Eigenschaftsname	Eigenschaftstyp	Eigenschaftswert
kind	String	"Simple Type Definition"
name	String	Lokaler Name des Typs (leer, wenn anonym)
target namespace	String	Namespace URI des Typs (leer, wenn anonym)
final	String-Sequenz("restriction"|"extension"|"list"|"union")	
context	enthaltende Komponente	
base type definition	Simple Type	
facets	Facet-Sequenz	
fundamental facets	Leere Sequenz (nicht implementiert)	

© 2019-2025 Altova GmbH

variety	String ("atomic"|"list"|"union")	
primitive type definition	Simple Type	
item type definition	(nur für Listentypen) Simple Type	
member type definitions	(nur für Union-Typen) Simple Type-Sequenz	

Type Alternative

Eigenschaftsname	Eigenschaftstyp	Eigenschaftswert
kind	String	"Type Alternative"
test	XPath-Eigenschaftsdatensatz	
type definition	Simple Type oder Complex Type	

XPath Property Record

Eigenschaftsname	Eigenschaftstyp	Eigenschaftswert
namespace bindings	Sequenz von Funktionen mit Eigenschaften ("prefix": string, "namespace": anyURI)	
default namespace	anyURI	
base URI	anyURI	Die statische Basis-UI des XPath-Ausdrucks
expression	String	Der XPath-Ausdruck als String

Schema (zwei Argumente)

`altova:schema(ComponentKind` *`als xs:string`*`, Name` *`als xs:QName`*`) als (function(xs:string) als item()*)?` XP3.1 XQ3.1

Gibt die im ersten Argument angegebene Komponentenart zurück, welche einen Namen hat, der mit dem im zweiten Argument angegebenen Namen übereinstimmt. Durch Auswahl einer der Eigenschaften der Komponente können Sie weiter navigieren.

- Wenn es sich bei dieser Eigenschaft um eine Komponente handelt, können Sie durch Auswahl einer dieser Komponenteneigenschaften einen Schritt tiefer navigieren. Dieser Schritt kann wiederholt werden, um weiter in das Schema zu navigieren.
- Wenn es sich bei der Komponente um einen atomaren Wert handelt, wird der atomare Wert zurückgegeben und Sie können nicht tiefer navigieren.

Anmerkung: In XQuery-Ausdrücken muss das Schema explizit importiert werden. In XPath-Ausdrücken muss das Schema in die Verarbeitungsumgebung, z.B. in XSLT mit der Anweisung **`xslt:import`** importiert worden sein.

Beispiele

- `import schema "" at "C:\Test\ExpReport.xsd";`

```
altova:schema("element declaration", xs:QName("OrgChart"))("type definition")
("content type")("particles")[3]!.("term")("kind")
```
gibt die `kind`-Eigenschaft des Terms der dritten `particles`-Komponente zurück. Diese particles-Komponente ist ein Nachfahr der Element-Deklaration mit dem QName `QName 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")]
  ```
  gibt für jedes **`facet`** jeder **`emailType`**-Komponente ein Array mit der Art und dem Wert des Facet zurück.

Komponenten und ihre Eigenschaften

Assertion

Eigenschaftsname	Eigenschaftstyp	Eigenschaftswert
kind	String	"Assertion"
test	XPath-Eigenschaftsdatensatz	

Attribute Declaration

Eigenschaftsname	Eigenschaftstyp	Eigenschaftswert
kind	String	"Attribute Declaration"
name	String	Lokaler Name des Attributs
target namespace	String	Namespace URI des Attributs
type definition	SimpleType oder ComplexType	
scope	Eine Funktion mit Eigenschaften ("class":"Scope", "variety": "global" oder "local", "parent": der enthaltende ComplexTyp bzw. die enthaltende Attributgruppe)	
value constraint	Falls vorhanden, eine Funktion mit Eigenschaften ("class": "Value Constraint", "variety": "fixed" oder "default", "value": atomarer Wert, "lexical form": String. Beachten Sie, dass die Eigenschaft "value" für Namespace-sensitive Typen nicht zur Verfügung steht.	
inheritable	Boolean	

Attribute Group Declaration

Eigenschaftsname	Eigenschaftstyp	Eigenschaftswert

© 2019-2025 Altova GmbH
  ```

kind	String	"Attribute Group Definition"
name	String	Lokaler Name der Attributgruppe
target namespace	String	Namespace URI der Attributgruppe
attribute uses	Sequenz von (Attribute Use)	
attribute wildcard	Optionale Attribut-Wildcard	

Attribute Use

Eigenschaftsname	Eigenschaftstyp	Eigenschaftswert
kind	String	"Attribute Use"
required	Boolean	true, wenn das Attribut obligatorisch ist, false, wenn es optional ist.
value constraint	Siehe Attribute Declaration	
inheritable	Boolean	

Attribute Wildcard

Eigenschaftsname	Eigenschaftstyp	Eigenschaftswert
kind	string	"Wildcard"
namespace constraint	Funktion mit Eigenschaften ("class": "Namespace Constraint", "variety": "any"|"enumeration"|"not", "namespaces": Sequenz von xs:anyURI, "disallowed names": Liste mit QNames und/oder den Strings "defined" und "definedSiblings"	
process contents	String ("strict"|"lax"|"skip")	

Complex Type

Eigenschaftsname	Eigenschaftstyp	Eigenschaftswert
kind	String	"Complex Type"
name	String	Lokaler Name des Typs (leer, wenn anonym)
target namespace	String	Namespace URI des Typs (leer, wenn anonym)
base type definition	Complex Type Definition	
final	String-Sequenz ("restriction"|"extension")	
context	Leere Sequenz (nicht implementiert)	

derivation method	String ("restriction"|"extension")	
abstract	Boolean	
attribute uses	Attribute Use-Sequenz	
attribute wildcard	Optionale Attribute Wildcard	
content type	Funktion mit Eigenschaften: ("class":"Content Type", "variety":string ("element-only"|"empty"|"mixed"|"simple"), particle: optionales Partikel, "open content": Funktion mit Eigenschaften ("class":"Open Content", "mode": string ("interleave"|"suffix"), "wildcard": Wildcard), "simple type definition": Simple Type)	
prohibited substitutions	String-Sequenz ("restriction"|"extension")	
assertions	Assertion-Sequenz	

Element Declaration

Eigenschaftsname	Eigenschaftstyp	Eigenschaftswert
kind	String	"Complex Type"
name	String	Lokaler Name des Typs (leer, wenn anonym)
target namespace	String	Namespace URI des Typs (leer, wenn anonym)
type definition	Simple Type oder Complex Type	
type table	Funktion mit Eigenschaften ("class":"Type Table", "alternatives": Type Alternative-Sequenz, "default type definition": Simple Type oder Complex Type)	
scope	Funktion mit Eigenschaften ("class":"Scope", "variety": ("global"|"local"), "parent": optionaler Complex Type)	
value constraint	siehe Attribute Declaration	
nillable	Boolean	
identity-constraint definitions	Identity Constraint-Sequenz	
substitution group affiliations	Element Declaration-Sequenz	
substitution group	String-Sequenz ("restriction"|"extension")	

© 2019-2025 Altova GmbH

exclusions		
disallowed substitutions	String-Sequenz ("restriction"|"extension"|"substitution")	
abstract	Boolean	

Element Wildcard

Eigenschaftsname	Eigenschaftstyp	Eigenschaftswert
kind	String	"Wildcard"
namespace constraint	Funktion mit Eigenschaften ("class": "Namespace Constraint", "variety": "any"|"enumeration"|"not", "namespaces": xs:anyURI-Sequenz, "disallowed names": Liste mit QNames und/oder den Strings "defined" und "definedSiblings"	
process contents	String ("strict"|"lax"|"skip")	

Facet

Eigenschaftsname	Eigenschaftstyp	Eigenschaftswert
kind	String	Der Name des Facet, z.B. "minLength" oder "enumeration"
value	abhängig vom Facet	Der Wert des Facet
fixed	Boolean	
typed-value	Nur für das Enumeration Facet, Array(xs:anyAtomicType*)	Ein Array, das Enumeration-Werte, von denen jeder im Allgemeinen eine Sequenz atomarer Werte sein kann, enthält. (Anmerkung: Die Eigenschaft "value" ist unabhängig vom tatsächlichen Typ für das Enumeration Facet eine String-Sequenz)

Identity Constraint

Eigenschaftsname	Eigenschaftstyp	Eigenschaftswert
kind	String	"Identity-Constraint Definition"
name	String	Lokaler Name des Constraint
target namespace	String	Namespace URI des Constraint
identity-constraint category	String ("key"|"unique"|"keyRef")	
selector	XPath-Eigenschaftsdatensatz	

fields	Sequenz von XPath-Eigenschaftsdatensätzen	
referenced key	(nur für keyRef): Identity Constraint	Der entsprechende Key Constraint

Model Group

Property name	Eigenschaftstyp	Eigenschaftswert
kind	String	"Model Group"
compositor	String ("sequence"|"choice"|"all")	
particles	Partikel-Sequenz	

Model Group Definition

Eigenschaftsname	Eigenschaftstyp	Eigenschaftswert
kind	String	"Model Group Definition"
name	String	Lokaler Name der Model Group
target namespace	String	Namespace URI der Model Group
model group	Model Group	

Notation

Eigenschaftsname	Eigenschaftstyp	Eigenschaftswert
kind	String	"Notation Declaration"
name	String	Lokaler Name der Notation
target namespace	String	Namespace URI der Notation
system identifier	anyURI	
public identifier	String	

Particle

Eigenschaftsname	Eigenschaftstyp	Eigenschaftswert
kind	String	"Particle"
min occurs	Integer	
max occurs	Integer oder String("unbounded")	
term	Element Declaration, Element Wildcard oder ModelGroup	

Simple Type

Eigenschaftsname	Eigenschaftstyp	Eigenschaftswert
kind	String	"Simple Type Definition"

© 2019-2025 Altova GmbH

name	String	Lokaler Name des Typs (leer, wenn anonym)
target namespace	String	Namespace URI des Typs (leer, wenn anonym)
final	String-Sequenz("restriction"|"extension"|"list"|"union")	
context	enthaltende Komponente	
base type definition	Simple Type	
facets	Facet-Sequenz	
fundamental facets	Leere Sequenz (nicht implementiert)	
variety	String ("atomic"|"list"|"union")	
primitive type definition	Simple Type	
item type definition	(nur für Listentypen) Simple Type	
member type definitions	(nur für Union-Typen) Simple Type-Sequenz	

Type Alternative

Eigenschaftsname	Eigenschaftstyp	Eigenschaftswert
kind	String	"Type Alternative"
test	XPath-Eigenschaftsdatensatz	
type definition	Simple Type oder Complex Type	

XPath Property Record

Eigenschaftsname	Eigenschaftstyp	Eigenschaftswert
namespace bindings	Sequenz von Funktionen mit Eigenschaften ("prefix": string, "namespace": anyURI)	
default namespace	anyURI	
base URI	anyURI	Die statische Basis-UI des XPath-Ausdrucks
expression	String	Der XPath-Ausdruck als String

Type

altova:type(Node *als item?***) als (function(xs:string) als item()*)?** XP3.1 XQ3.1

Die Funktion **altova:type** übermittelt einen Element- oder Attribut-Node eines XML-Dokuments und gibt die Typinformationen des Node aus dem PSVI zurück.

Anmerkung: Das XML-Dokument muss eine Schema-Deklaration haben, damit das Schema referenziert werden kann.

Beispiele

- ```
  for $element in //Email
  let $type := altova:type($element)
  return $type
  ```
  gibt eine Funktion zurück, die die Typinformationen des Node `Email` enthält.

- ```
  for $element in //Email
  let $type := altova:type($element)
  return $type ("kind")
  ```
  ermittelt anhand der Typ-Komponente des Node (Simple Type oder Complex Type) den Wert der Eigenschaft **`kind`** der Komponente.

Der Parameter **`"_props"`** gibt die Eigenschaften der ausgewählten Komponente zurück, z.B:

- ```
  for $element in //Email
  let $type := altova:type($element)
  return ($type ("kind"), $type ("_props"))
  ```
  nimmt die Typkomponente des Node `Email` (Simple Type oder Complex Type) und gibt (i) den Wert der Eigenschaft **`kind`** der Komponente zurück und anschließend (ii) die Eigenschaften dieser Komponente.

Komponenten und ihre Eigenschaften

Assertion

Eigenschaftsname	Eigenschaftstyp	Eigenschaftswert
kind	String	"Assertion"
test	XPath-Eigenschaftsdatensatz	

Attribute Declaration

Eigenschaftsname	Eigenschaftstyp	Eigenschaftswert
kind	String	"Attribute Declaration"
name	String	Lokaler Name des Attributs
target namespace	String	Namespace URI des Attributs
type definition	SimpleType oder ComplexType	
scope	Eine Funktion mit Eigenschaften ("class":"Scope", "variety": "global" oder "local", "parent": der enthaltende ComplexTyp bzw. die enthaltende Attributgruppe)	
value constraint	Falls vorhanden, eine Funktion mit Eigenschaften ("class": "Value	

© 2019-2025 Altova GmbH
  ```

	Constraint", "variety": "fixed" oder "default", "value": atomarer Wert, "lexical form": String. Beachten Sie, dass die Eigenschaft "value" für Namespace-sensitive Typen nicht zur Verfügung steht.	
inheritable	Boolean	

Attribute Group Declaration

Eigenschaftsname	Eigenschaftstyp	Eigenschaftswert
kind	String	"Attribute Group Definition"
name	String	Lokaler Name der Attributgruppe
target namespace	String	Namespace URI der Attributgruppe
attribute uses	Sequenz von (Attribute Use)	
attribute wildcard	Optionale Attribut-Wildcard	

Attribute Use

Eigenschaftsname	Eigenschaftstyp	Eigenschaftswert
kind	String	"Attribute Use"
required	Boolean	true, wenn das Attribut obligatorisch ist, false, wenn es optional ist.
value constraint	Siehe Attribute Declaration	
inheritable	Boolean	

Attribute Wildcard

Eigenschaftsname	Eigenschaftstyp	Eigenschaftswert
kind	string	"Wildcard"
namespace constraint	Funktion mit Eigenschaften ("class": "Namespace Constraint", "variety": "any"|"enumeration"|"not", "namespaces": Sequenz von xs:anyURI, "disallowed names": Liste mit QNames und/oder den Strings "defined" und "definedSiblings"	
process contents	String ("strict"|"lax"|"skip")	

Complex Type

Eigenschaftsname	Eigenschaftstyp	Eigenschaftswert

kind	String	"Complex Type"
name	String	Lokaler Name des Typs (leer, wenn anonym)
target namespace	String	Namespace URI des Typs (leer, wenn anonym)
base type definition	Complex Type Definition	
final	String-Sequenz ("restriction"|"extension")	
context	Leere Sequenz (nicht implementiert)	
derivation method	String ("restriction"|"extension")	
abstract	Boolean	
attribute uses	Attribute Use-Sequenz	
attribute wildcard	Optionale Attribute Wildcard	
content type	Funktion mit Eigenschaften: ("class":"Content Type", "variety":string ("element-only"|"empty"|"mixed"|"simple"), particle: optionales Partikel, "open content": Funktion mit Eigenschaften ("class":"Open Content", "mode": string ("interleave"|"suffix"), "wildcard": Wildcard), "simple type definition": Simple Type)	
prohibited substitutions	String-Sequenz ("restriction"|"extension")	
assertions	Assertion-Sequenz	

Element Declaration

Eigenschaftsname	Eigenschaftstyp	Eigenschaftswert
kind	String	"Complex Type"
name	String	Lokaler Name des Typs (leer, wenn anonym)
target namespace	String	Namespace URI des Typs (leer, wenn anonym)
type definition	Simple Type oder Complex Type	
type table	Funktion mit Eigenschaften ("class":"Type Table", "alternatives": Type Alternative-Sequenz, "default type definition": Simple Type oder Complex Type)	
scope	Funktion mit Eigenschaften ("class":"Scope", "variety":	

 © 2019-2025 Altova GmbH

	("global"|"local"), "parent": optionaler Complex Type)	
value constraint	siehe Attribute Declaration	
nillable	Boolean	
identity-constraint definitions	Identity Constraint-Sequenz	
substitution group affiliations	Element Declaration-Sequenz	
substitution group exclusions	String-Sequenz ("restriction"|"extension")	
disallowed substitutions	String-Sequenz ("restriction"|"extension"|"substitution")	
abstract	Boolean	

Element Wildcard

Eigenschaftsname	Eigenschaftstyp	Eigenschaftswert
kind	String	"Wildcard"
namespace constraint	Funktion mit Eigenschaften ("class": "Namespace Constraint", "variety": "any"|"enumeration"|"not", "namespaces": xs:anyURI-Sequenz, "disallowed names": Liste mit QNames und/oder den Strings "defined" und "definedSiblings"	
process contents	String ("strict"|"lax"|"skip")	

Facet

Eigenschaftsname	Eigenschaftstyp	Eigenschaftswert
kind	String	Der Name des Facet, z.B. "minLength" oder "enumeration"
value	abhängig vom Facet	Der Wert des Facet
fixed	Boolean	
typed-value	Nur für das Enumeration Facet, Array(xs:anyAtomicType*)	Ein Array, das Enumeration-Werte, von denen jeder im Allgemeinen eine Sequenz atomarer Werte sein kann, enthält. (Anmerkung: Die Eigenschaft "value" ist unabhängig vom tatsächlichen Typ für das Enumeration Facet eine String-Sequenz)

Identity Constraint

Eigenschaftsname	Eigenschaftstyp	Eigenschaftswert
kind	String	"Identity-Constraint Definition"
name	String	Lokaler Name des Constraint
target namespace	String	Namespace URI des Constraint
identity-constraint category	String ("key"|"unique"|"keyRef")	
selector	XPath-Eigenschaftsdatensatz	
fields	Sequenz von XPath-Eigenschaftsdatensätzen	
referenced key	(nur für keyRef): Identity Constraint	Der entsprechende Key Constraint

Model Group

Property name	Eigenschaftstyp	Eigenschaftswert
kind	String	"Model Group"
compositor	String ("sequence"|"choice"|"all")	
particles	Partikel-Sequenz	

Model Group Definition

Eigenschaftsname	Eigenschaftstyp	Eigenschaftswert
kind	String	"Model Group Definition"
name	String	Lokaler Name der Model Group
target namespace	String	Namespace URI der Model Group
model group	Model Group	

Notation

Eigenschaftsname	Eigenschaftstyp	Eigenschaftswert
kind	String	"Notation Declaration"
name	String	Lokaler Name der Notation
target namespace	String	Namespace URI der Notation
system identifier	anyURI	
public identifier	String	

Particle

Eigenschaftsname	Eigenschaftstyp	Eigenschaftswert
kind	String	"Particle"
min occurs	Integer	

© 2019-2025 Altova GmbH

max occurs	Integer oder String("unbounded")	
term	Element Declaration, Element Wildcard oder ModelGroup	

Simple Type

Eigenschaftsname	Eigenschaftstyp	Eigenschaftswert
kind	String	"Simple Type Definition"
name	String	Lokaler Name des Typs (leer, wenn anonym)
target namespace	String	Namespace URI des Typs (leer, wenn anonym)
final	String-Sequenz("restriction"|"extension"|"list"|"union")	
context	enthaltende Komponente	
base type definition	Simple Type	
facets	Facet-Sequenz	
fundamental facets	Leere Sequenz (nicht implementiert)	
variety	String ("atomic"|"list"|"union")	
primitive type definition	Simple Type	
item type definition	(nur für Listentypen) Simple Type	
member type definitions	(nur für Union-Typen) Simple Type-Sequenz	

Type Alternative

Eigenschaftsname	Eigenschaftstyp	Eigenschaftswert
kind	String	"Type Alternative"
test	XPath-Eigenschaftsdatensatz	
type definition	Simple Type oder Complex Type	

XPath Property Record

Eigenschaftsname	Eigenschaftstyp	Eigenschaftswert
namespace bindings	Sequenz von Funktionen mit Eigenschaften ("prefix": string, "namespace": anyURI)	
default namespace	anyURI	
base URI	anyURI	Die statische Basis-UI des XPath-Ausdrucks

expression	String	Der XPath-Ausdruck als String

12.2.2.1.7 XPath/XQuery-Funktionen: Sequenz

Die Sequenz-Erweiterungsfunktionen von Altova können in XPath- und XQuery-Ausdrücken verwendet werden und stellen zusätzliche Funktionen für die Verarbeitung von Daten zur Verfügung. Die Funktionen in diesem Abschnitt können mit dem **XPath 3.0-** und **XQuery 3.0**-Prozessor von Altova verwendet werden. Sie stehen im Zusammenhang mit XPath/XQuery zur Verfügung.

Anmerkung zur Benennung von Funktionen und zur Anwendbarkeit der Sprache

Altova-Erweiterungsfunktionen können in XPath/XQuery-Ausdrücken verwendet werden. Dadurch stehen neben den Funktionen in der Standardbibliothek der XPath-, XQuery- und XSLT-Funktionen zusätzliche Funktionen zur Verfügung. Die Altova-Erweiterungsfunktionen befinden sich im **Altova-Erweiterungsfunktions-Namespace,** `http://www.altova.com/xslt-extensions` und sind in diesem Abschnitt mit dem Präfix `altova:`, das als an diesen Namespace gebunden angenommen wird, gekennzeichnet. Beachten Sie, dass manche Funktionen in zukünftigen Versionen Ihres Produkts eventuell nicht mehr unterstützt werden oder dass sich das Verhalten einzelner Funktionen ändern kann. Um zu sehen, welche Altova-Erweiterungsfunktionen unterstützt werden, lesen Sie bitte die Dokumentation zur jeweiligen Release.

XPath-Funktionen (in XPath-Ausdrücken in XSLT verwendet):	XP1 XP2 XP3.1
XSLT-Funktionen (in XPath-Ausdrücken in XSLT verwendet):	XSLT1 XSLT2 XSLT3
XQuery-Funktionen (in XQuery-Ausdrücken in XQuery verwendet):	XQ1 XQ3.1

attributes [altova:]

altova:attributes(AttributeName *als xs:string***)** als **attribute()*** XP3.1 XQ3.1
Gibt alle Attribute zurück, die einen lokalen Namen haben, der mit dem im Input-Argument `AttributeName` angegebenen Namen identisch ist. Die Groß- und Kleinschreibung wird bei der Suche, die entlang der `attribute::` Achse durchgeführt wird, beachtet. Das bedeutet, dass der Kontext-Node der Parent-Element-Node sein muss.

Beispiele

- `altova:attributes("MyAttribute")` gibt `MyAttribute()*` zurück

altova:attributes(AttributeName *als xs:string***, SearchOptions** *als xs:string***)** als **attribute()*** XP3.1 XQ3.1
Gibt alle Attribute zurück, die einen lokalen Namen haben, der mit dem im Input-Argument `AttributeName` angegebenen Namen identisch ist. Die Groß- und Kleinschreibung wird bei der Suche, die entlang der `attribute::` Achse durchgeführt wird, beachtet. Der Kontext-Node muss der Parent-Element-Node sein. Das zweite Argument ist ein String, der Options-Flags enthält. Zur Verfügung stehen die folgenden Flags:
r = wechselt zu einer Suche mittels Regular Expression; bei `AttributeName` muss es sich in diesem Fall

 © 2019-2025 Altova GmbH

um einen Regular Expression-Suchstring handeln;
f = Wenn diese Option definiert ist, liefert `AttributeName` eine vollständige Übereinstimmung; andernfalls muss `AttributeName` nur teilweise mit einem Attributnamen übereinstimmen, damit dieses Attribut zurückgegeben wird. Wenn **f** z.B. nicht definiert ist, gibt `MyAtt MyAttribute` zurück;
i = wechselt zu einer Suche ohne Berücksichtigung der Groß- und Kleinschreibung;
p = inkludiert das Namespace-Präfix in die Suche; `AttributeName` sollte in diesem Fall das Namespace-Präfix enthalten, z.B.: `altova:MyAttribute`.
Die Flags können in jeder Reihenfolge angegeben werden. Ungültige Flags erzeugen eine Fehlermeldung. Sie können ein oder mehrere Flags weglassen. Es ist auch der leere String zulässig. Das Resultat ist dasselbe wie bei Verwendung der Funktion mit nur einem Argument (*siehe vorherige Signatur*). Unzulässig ist jedoch die Verwendung einer leeren Sequenz als zweites Argument.

Beispiele

- **altova:attributes**("MyAttribute", "rfip") gibt MyAttribute()* zurück
- **altova:attributes**("MyAttribute", "pri") gibt MyAttribute()* zurück
- **altova:attributes**("MyAtt", "rip") gibt MyAttribute()* zurück
- **altova:attributes**("MyAttributes", "rfip") gibt keine Übereinstimmung zurück
- **altova:attributes**("MyAttribute", "") gibt MyAttribute()* zurück
- **altova:attributes**("MyAttribute", "Rip") gibt einen Fehler zurück, dass das Flag unbekannt ist.
- **altova:attributes**("MyAttribute",) gibt den Fehler zurück, dass das zweite Argument fehlt.

elements [altova:]

altova:elements(ElementName *als xs:string***)** als **element()*** **XP3.1** **XQ3.1**
Gibt alle Elemente zurück, die einen lokalen Namen haben, der mit dem im Input-Argument `ElementName` angegebenen Namen identisch ist. Die Groß- und Kleinschreibung wird bei der Suche, die entlang der `child::` Achse durchgeführt wird, beachtet. Der Kontext-Node muss der Parent-Node des gesuchten Elements sein.

Beispiele

- **altova:elements**("MyElement") gibt MyElement()* zurück

altova:elements(ElementName *als xs:string*, **SearchOptions** *als xs:string***)** als **element()*** **XP3.1** **XQ3.1**
Gibt alle Elemente zurück, die einen lokalen Namen haben, der mit dem im Input-Argument `ElementName` angegebenen Namen identisch ist. Die Groß- und Kleinschreibung wird bei der Suche, die entlang der `child::` Achse durchgeführt wird, beachtet. Der Kontext-Node muss der Parent-Node des gesuchten Elements sein. Das zweite Argument ist ein String, der Options-Flags enthält. Zur Verfügung stehen die folgenden Flags:
r = wechselt zu einer Suche mittels Regular Expression; bei `ElementName` muss es sich in diesem Fall um einen Regular Expression-Suchstring handeln;
f = Wenn diese Option definiert ist, liefert `ElementName` eine vollständige Übereinstimmung; andernfalls muss `ElementName` nur teilweise mit einem Elementnamen übereinstimmen, damit dieses Element zurückgegeben wird. Wenn **f** z.B. nicht definiert ist, gibt `MyElem MyElement` zurück;
i = wechselt zu einer Suche ohne Berücksichtigung der Groß- und Kleinschreibung;
p = inkludiert das Namespace-Präfix in die Suche; `ElementName` sollte in diesem Fall das Namespace-Präfix enthalten, z.B.: `altova:MyElement`.
Die Flags können in jeder Reihenfolge angegeben werden. Ungültige Flags erzeugen eine Fehlermeldung. Sie können ein oder mehrere Flags weglassen. Es ist auch der leere String zulässig. Das Resultat ist dasselbe wie bei Verwendung der Funktion mit nur einem Argument (siehe vorherige Signatur). Unzulässig

ist jedoch die Verwendung einer leeren Sequenz.

Beispiele

- **altova:elements**("MyElement", "rip") gibt MyElement()* zurück
- **altova:elements**("MyElement", "pri") gibt MyElement()* zurück
- **altova:elements**("MyElement", "") gibt MyElement()* zurück
- **altova:elements**("MyElem", "rip") gibt MyElement()* zurück
- **altova:elements**("MyElements", "rfip") gibt keine Übereinstimmung zurück
- **altova:elements**("MyElement", "Rip") gibt einen Fehler zurück, dass das Flag unbekannt ist.
- **altova:elements**("MyElement",) gibt den Fehler zurück, dass das zweite Argument fehlt.

find-first [altova:]

altova:find-first((*item()), (CheckFunction(item() *als xs:boolean*))** als **item()?** XP3.1 XQ3.1

Diese Funktion verwendet zwei Argumente. Das erste Argument ist eine Sequenz von einem oder mehreren Elementen eines beliebigen Datentyps. Das zweite Argument, Condition, ist eine Referenz zu einer XPath-Funktion, die ein Argument erhält. (hat einen Stellenwert 1) und einen Booleschen Wert zurückgibt. Jedes Element von **Sequence** wird der Reihe nach der in Condition referenzierten Funktion bereitgestellt. (*Beachten Sie:* Die Funktion hat ein einziges Argument.) Das erste **Sequence** Element, bei dem das Resultat von **Condition true()** ist, wird als das Ergebnis von altova:find-first zurückgegeben. Anschließend wird die Iteration gestoppt.

Beispiele

- **altova:find-first**(5 to 10, function($a) {$a mod 2 = 0}) gibt xs:integer 6 zurück

  Das Argument **Condition** referenziert die XPath 3.0 Inline-Funktion, **function()**, welche eine Inline-Funktion **$a** deklariert und diese anschließend definiert. Die einelnen Elemente im Argument **Sequence** von **altova:find-first** werden der Reihe nach an **$a** als sein Input-Wert übergeben. Der Input-Wert wird an der Bedingung in der Funktionsdefinition ($a mod 2 = 0) überprüft. Der erste Input-Wert, der diese Bedingung erfüllt, wird als das Ergebnis von **altova:find-first** (in diese Fall **6**) zurückgegeben.

- **altova:find-first**((1 to 10), (function($a) {$a+3=7})) gibt xs:integer 4 zurück

Weitere Beispiele

Wenn die Datei **C:\Temp\Customers.xml** vorhanden ist:

- **altova:find-first**(("C:\Temp\Customers.xml", "http://www.altova.com/index.html"), (doc-available#1)) gibt C:\Temp\Customers.xml zurück

Wenn die Datei **C:\Temp\Customers.xml** nicht vorhanden ist und **http://www.altova.com/index.html** vorhanden ist:

- **altova:find-first**(("C:\Temp\Customers.xml", "http://www.altova.com/index.html"), (doc-available#1)) gibt http://www.altova.com/index.html zurück

Wenn weder die Datei **C:\Temp\Customers.xml** noch **http://www.altova.com/index.html** vorhanden ist:

- **altova:find-first**(("C:\Temp\Customers.xml", "http://www.altova.com/index.html"),

© 2019-2025 Altova GmbH

(`doc-available#1`)) gibt kein Ergebnis zurück

Anmerkungen zu den obigen Beispielen

- Die XPath 3.0-Funktion, `doc-available`, erhält ein einziges Argument, das als URI verwendet wird. Sie gibt nur dann `true` zurück, wenn unter der angegebenen URI ein Dokument-Node gefunden wird. Das Dokument unter der angegebenen URI muss daher ein XML-Dokument sein.
- Die Funktion `doc-available` kann für **`Condition,`** das zweite Argument von `altova:find-first` verwendet werden, da sie nur ein Argument erhält (Stelligkeit=1), da sie ein Element `item()` als Input erhält (ein String, der als URI verwendet wird) und einen Booleschen Wert zurückgibt.
- Beachten Sie, dass `doc-available` nur referenziert und nicht direkt aufgerufen wird. Das angehängte Suffix `#1` gibt eine Funktion mit einer Stelligkeit 1 an. Als Ganzes bedeutet `doc-available#1`: *Verwende die Funktion doc-availabe(), welche die Stelligkeit=1 hat und übergib die einzelnen Elemente in der ersten Sequenz der Reihe nach als einziges Argument an die Funktion.* Als Ergebnis wird jeder der beiden Strings an **`doc-available()`** übergeben. Die Funktion verwendet den String als URI und überprüft, ob unter der URI ein Dokument-Node vorhanden ist. Wenn dies der Fall ist, wird **`doc-available()`** zu `true()` ausgewertet und der String wird als Ergebnis der Funktion **`altova:find-first`** zurückgegeben. *Beachten Sie zur Funktion doc-available(), dass relative Pfade relativ zu aktuellen Basis-URI aufgelöst werden. Die Basis-URI ist standardmäßig die URI des XML-Dokuments, von dem aus die Funktion geladen wird.*

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`***`)) as item()*`** XP3.1 XQ3.1

Diese Funktion verwendet drei Argumente:

- Die ersten beiden Argumente, **`Seq-01`** und **`Seq-02`**, sind Sequenzen von einem oder mehreren Elementen eines beliebigen Datentyps.
- Das dritte Argument, **`Condition`**, ist eine Referenz auf eine XPath-Funktion, die zwei Argumente erhält (d.h. eine Stelligkeit **`2`** hat) und einen Booleschen Wert zurückgibt.

Die Elemente von **`Seq-01`** und **`Seq-02`** werden als die Argumente der Funktion **`Condition`** in geordneten Paaren übergeben (je ein Element aus jeder Sequenz bildet ein Paar). Die Paare sind folgendermaßen geordnet.

```
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)
```

Das erste geordnete Paar, bei dem die Funktion **`Condition`** zu **`true()`** ausgewertet wird, wird als Ergebnis von **`altova:find-first-combination`** zurückgegeben. Beachten Sie: (i) Wenn die Funktion **`Condition`** durch die bereitgestellten Argumentpaare iteriert und nicht ein einziges Mal zu **`true()`** ausgewertet wird, so gibt **`altova:find-first-combination`** *Keine Ergebnisse* zurück; (ii) Das Ergebnis von **`altova:find-first-combination`** ist immer ein Elementpaar (eines beliebigen Datentyps) oder gar kein Element.

Beispiele

- **altova:find-first-combination**(11 to 20, 21 to 30, function($a, $b) {$a+$b = 32}) gibt die Sequenz xs:integers (11, 21) zurück
- **altova:find-first-combination**(11 to 20, 21 to 30, function($a, $b) {$a+$b = 33}) gibt die Sequenz xs:integers (11, 22) zurück
- **altova:find-first-combination**(11 to 20, 21 to 30, function($a, $b) {$a+$b = 34}) gibt die Sequenz xs:integers (11, 23) zurück

find-first-pair [altova:]

altova:find-first-pair((Seq-01 *als item()**), **(Seq-02** *als item()**), **(Condition(Seq-01-Item, Seq-02-Item** *als xs:boolean*)) als **item()*** XP3.1 XQ3.1
Diese Funktion erhält drei Argumente:

- Die ersten beiden Argumente, **Seq-01** und **Seq-02**, sind Sequenzen von einem oder mehreren Elementen eines beliebigen Datentyps.
- Das dritte Argument, **Condition**, ist eine Referenz auf eine XPath-Funktion, die zwei Argumente erhält (d.h. eine Stelligkeit **2** hat) und einen Booleschen Wert zurückgibt.

Die Elemente von **Seq-01** und **Seq-02** werden als die Argumente der Funktion **Condition** in geordneten Paaren übergeben. Die Paare sind folgendermaßen geordnet.

```
If   Seq-01 = X1, X2, X3 ... Xn
And  Seq-02 = Y1, Y2, Y3 ... Yn
Then (X1 Y1), (X2 Y2), (X3 Y3) ... (Xn Yn)
```

Das erste geordnete Paar, bei dem die Funktion **Condition** zu **true()** ausgewertet wird, wird als Ergebnis von **altova:find-first-pair** zurückgegeben. Beachten Sie: (i) Wenn die Funktion **Condition** durch die bereitgestellten Argumentpaare iteriert und nicht ein einziges Mal zu **true()** ausgewertet wird, so gibt **altova:find-first-pair** *Keine Ergebnisse* zurück; (ii) Das Ergebnis von **altova:find-first-pair** ist immer ein Elementpaar (eines beliebigen Datentyps) oder gar kein Element.

Beispiele

- **altova:find-first-pair**(11 to 20, 21 to 30, function($a, $b) {$a+$b = 32}) gibt die Sequenz xs:integers (11, 21) zurück
- **altova:find-first-pair**(11 to 20, 21 to 30, function($a, $b) {$a+$b = 33}) gibt *Keine Ergebnisse* zurück

Beachten Sie anhand der zwei Beispiel oben, dass die Paare folgendermaßen geordnet sind: (11, 21) (12, 22) (13, 23)...(20, 30). Aus diesem Grund gibt das zweite Beispiel *Keine Ergebnisse* zurück (da keine geordnetes Paar die Summe 33 ergibt).

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*)) **as xs:integer** XP3.1 XQ3.1
Diese Funktion erhält drei Argumente:

- Die ersten beiden Argumente, **Seq-01** und **Seq-02**, sind Sequenzen von einem oder mehreren Elementen eines beliebigen Datentyps.

© 2019-2025 Altova GmbH

- Das dritte Argument, **Condition**, ist eine Referenz auf eine XPath-Funktion, die zwei Argumente erhält (d.h. eine Stelligkeit **2** hat) und einen Booleschen Wert zurückgibt.

Die Elemente von **Seq-01** und **Seq-02** werden als die Argumente der Funktion **Condition** in geordneten Paaren übergeben. Die Paare sind folgendermaßen geordnet.

```
If   Seq-01 = X1, X2, X3 ... Xn
And  Seq-02 = Y1, Y2, Y3 ... Yn
Then (X1 Y1), (X2 Y2), (X3 Y3) ... (Xn Yn)
```

Als Ergebnis von **altova:find-first-pair-pos** wird die Indexposition des ersten geordneten Paars, bei dem die Funktion **Condition** zu **true()** ausgewertet wird, zurückgegeben. Beachten Sie: Wenn die Funktion **Condition** durch die bereitgestellten Argumentpaare iteriert und kein einziges Mal zu **true()** ausgewertet wird, so gibt **altova:find-first-pair-pos** *Keine Ergebnisse* zurück.

Beispiele

- **altova:find-first-pair-pos**(`11 to 20, 21 to 30, function($a, $b) {$a+$b = 32}`) gibt `1` zurück
- **altova:find-first-pair-pos**(`11 to 20, 21 to 30, function($a, $b) {$a+$b = 33}`) gibt *Keine Ergebnisse* zurück

Beachten Sie anhand der zwei Beispiel oben, dass die Paare folgendermaßen geordnet sind: `(11, 21) (12, 22) (13, 23)...(20, 30)`. Im ersten Beispiel gibt die Funktion **Condition** bei Auswertung des ersten Paars **true()** zurück, daher wird dessen Indexposition in der Sequenz, `1`, zurückgegeben. Das zweite Beispiel gibt *Keine Ergebnisse* zurück (da keine geordnetes Paar die Summe `33` ergibt).

find-first-pos [altova:]

altova:find-first-pos((*item()), (CheckFunction(item() *als xs:boolean*))** als **xs:integer?** XP3.1 XQ3.1

Diese Funktion verwendet zwei Argumente. Das erste Argument ist eine Sequenz von einem oder mehreren Elementen eines beliebigen Datentyps. Das zweite Argument, `Condition`, ist eine Referenz zu einer XPath-Funktion, die ein Argument erhält. (hat einen Stellenwert `1`) und einen Booleschen Wert zurückgibt. Jedes Element von **Sequence** wird der Reihe nach der in `Condition` referenzierten Funktion bereitgestellt. (*Beachten Sie:* Die Funktion hat ein einziges Argument.) Das erste **Sequence** Element, bei dem das Resultat von **Condition true()** ist, wird als das Ergebnis von **altova:find-first-pos** zurückgegeben. Anschließend wird die Iteration gestoppt.

Beispiele

- **altova:find-first-pos**(`5 to 10, function($a) {$a mod 2 = 0}`) gibt `xs:integer 2` zurück

Das Argument **Condition** referenziert die XPath 3.0 Inline-Funktion, **function()**, welche eine Inline-Funktion **$a** deklariert und diese anschließend definiert. Die einelnen Elemente im Argument **Sequence** von **altova:find-first-pos** werden der Reihe nach an **$a** als sein Input-Wert übergeben. Der Input-Wert wird an der Bedingung in der Funktionsdefinition (`$a mod 2 = 0`) überprüft. Die Indexposition in der Sequenz des ersten Input-Werts, die diese Bedingung erfüllt, wird als das Ergebnis von **altova:find-first-pos** zurückgegeben (in diesem Fall `2`, da **6,** der erste Wert in der Sequenz, der die Bedingung erfüllt, sich in der Sequenz an der Indexposition `2` befindet).

Weitere Beispiele
Wenn die Datei **C:\Temp\Customers.xml** vorhanden ist:

- **altova:find-first-pos**(("C:\Temp\Customers.xml", "http://www.altova.com/index.html"), (doc-available#1)) gibt 1 zurück

Wenn die Datei **C:\Temp\Customers.xml** nicht vorhanden ist und **http://www.altova.com/index.html** vorhanden ist:

- **altova:find-first-pos**(("C:\Temp\Customers.xml", "http://www.altova.com/index.html"), (doc-available#1)) gibt 2 zurück

Wenn weder die Datei **C:\Temp\Customers.xml** noch **http://www.altova.com/index.html** vorhanden ist:

- **altova:find-first-pos**(("C:\Temp\Customers.xml", "http://www.altova.com/index.html"), (doc-available#1)) gibt kein Ergebnis zurück

Anmerkungen zu den obigen Beispielen

- Die XPath 3.0-Funktion, `doc-available`, erhält ein einziges Argument, das als URI verwendet wird. Sie gibt nur dann `true` zurück, wenn unter der angegebenen URI ein Dokument-Node gefunden wird. (Das Dokument unter der angegebenen URI muss daher ein XML-Dokument sein.)
- Die Funktion `doc-available` kann für **Condition,** das zweite Argument von `altova:find-first-pos` verwendet werden, da sie nur ein Argument erhält (Stelligkeit=1), da sie ein Element `item()` als Input erhält (ein String, der als URI verwendet wird) und einen Booleschen Wert zurückgibt.
- Beachten Sie, dass `doc-available` nur referenziert und nicht direkt aufgerufen wird. Das angehängte Suffix `#1` gibt eine Funktion mit einer Stelligkeit 1 an. Als Ganzes bedeutet `doc-available#1`: *Verwende die Funktion doc-availabe(), welche die Stelligkeit=1 hat und übergib die einzelnen Elemente in der ersten Sequenz der Reihe nach als einziges Argument an die Funktion.* Als Ergebnis wird jeder der beiden Strings an **doc-available()** übergeben. Die Funktion verwendet den String als URI und überprüft, ob unter der URI ein Dokument-Node vorhanden ist. Wenn dies der Fall ist, wird **doc-available()** zu `true()` ausgewertet und der String wird als Ergebnis der Funktion **altova:find-first** zurückgegeben. *Beachten Sie zur Funktion doc-available(), dass relative Pfade relativ zu aktuellen Basis-URI aufgelöst werden. Die Basis-URI ist standardmäßig die URI des XML-Dokuments, von dem aus die Funktion geladen wird.*

for-each-attribute-pair [altova:]

altova:for-each-attribute-pair(Seq1 *als element()?*, **Seq2** *als element()?*, **Function** *als function()*) als **item()*** XP3.1 XQ3.1

Die beiden ersten Argumente identifizieren zwei Elemente, anhand deren Attribute Attributpaare gebildet werden, wobei das eine Attribut eines Paars aus dem ersten Element und das andere aus dem zweiten Element stammt. Die Attributpaare werden auf Basis ihres übereinstimmenden Namens ausgewählt und alphabetisch (nach ihren Namen) zu einer Gruppe geordnet. Falls es zu einem Attribut im anderen Element keine Entsprechung gibt, ist das Paar "nicht verbunden", d.h. es besteht nur aus einem Mitglied. Das Funktionselement (das dritte Argument `Function`) wird auf die einzelnen Paare (verbundene und nicht

 © 2019-2025 Altova GmbH

verbundene) in der Sequenz der Paare separat angewendet, wodurch als Ausgabe eine Sequenz von Einträgen erzeugt wird.

Beispiele

- **altova:for-each-attribute-pair**(/Example/Test-A, /Example/Test-B, function($a, $b) {$a+b}) gibt zurück ...

  (2, 4, 6) wenn
  <Test-A att1="1" att2="2" att3="3" />
  <Test-B att1="1" att2="2" att3="3" />

  (2, 4, 6) wenn
  <Test-A att2="2" att1="1" att3="3" />
  <Test-B att3="3" att2="2" att1="1" />

  (2, 6) wenn
  <Test-A att4="4" att1="1" att3="3" />
  <Test-B att3="3" att2="2" att1="1" />

  Anmerkung: Das Ergebnis **(2, 6)** *wird mit Hilfe der folgenden Aktion ermittelt:* **(1+1, ()+2, 3+3, 4+())**. *Wenn einer der Operanden eine leere Sequenz ist, wie dies bei Eintrag 2 und 4 der Fall ist, so ist das Ergebnis der Addition eine leere Sequenz.*

- **altova:for-each-attribute-pair**(/Example/Test-A, /Example/Test-B, concat#2) gibt zurück ...

  (11, 22, 33) wenn
  <Test-A att1="1" att2="2" att3="3" />
  <Test-B att1="1" att2="2" att3="3" />

  (11, 2, 33, 4) wenn
  <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 *als item()**, **SecondSequence** *als item()**, **Function($i,$j){$i || $j})** als **item()*** XP3.1 XQ3.1

Die Elemente der zwei Sequenzen in den ersten beiden Argumenten werden miteinander kombiniert, so dass jedes Element in der ersten Sequenz der Reihe nach einmal mit jedem Element in der zweiten Sequenz kombiniert wird. Die als drittes Argument angegebene Funktion wird auf die einzelnen Kombinationen in der erzeugten Sequenz angewendet, wodurch als Ausgabe eine Sequenz von Elementen erzeugt wird (*siehe Beispiel*).

Beispiele

- **altova:for-each-combination**(('a', 'b', 'c'), ('1', '2', '3'), function($i, $j) {$i || $j}) gibt ('a1', 'a2', 'a3', 'b1', 'b2', 'b3', 'c1', 'c2', 'c3') zurück

for-each-matching-attribute-pair [altova:]

altova:for-each-matching-attribute-pair(Seq1 *als element()?*, **Seq2** *als element()?*,

Function *als function()*) als **item()*** XP3.1 XQ3.1

Die beiden ersten Argumente identifizieren zwei Elemente, anhand deren Attribute Attributpaare gebildet werden, wobei das eine Attribut eines Paars aus dem ersten Element und das andere aus dem zweiten Element stammt. Die Attributpaare werden auf Basis ihres übereinstimmenden Namens ausgewählt und alphabetisch (nach ihren Namen) zu einer Gruppe geordnet. Falls es zu einem Attribut im anderen Element keine Entsprechung gibt, wird kein Paar gebildet. Das Funktionselement (das dritte Argument `Function`) wird auf die einzelnen Paare in der Sequenz der Paare separat angewendet, wodurch als Ausgabe eine Sequenz von Einträgen erzeugt wird.

Beispiele

- **altova:for-each-matching-attribute-pair**(`/Example/Test-A, /Example/Test-B, function($a, $b){$a+b}`) gibt zurück ...

  (2, 4, 6) wenn
  ```
  <Test-A att1="1" att2="2" att3="3" />
  <Test-B att1="1" att2="2" att3="3" />
  ```

  (2, 4, 6) wenn
  ```
  <Test-A att2="2" att1="1" att3="3" />
  <Test-B att3="3" att2="2" att1="1" />
  ```

  (2, 6) wenn
  ```
  <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`) gibt zurück ....

  (11, 22, 33) wenn
  ```
  <Test-A att1="1" att2="2" att3="3" />
  <Test-B att1="1" att2="2" att3="3" />
  ```

  (11, 33) wenn
  ```
  <Test-A att4="4" att1="1" att3="3" />
  <Test-B att3="3" att2="2" att1="1" />
  ```

substitute-empty [altova:]

altova:substitute-empty(FirstSequence *als item()**, **SecondSequence** *als item()*) als **item()*** XP3.1 XQ3.1

Wenn `FirstSequence` leer ist, wird `SecondSequence` zurückgegeben. Wenn `FirstSequence` nicht leer ist, wird `FirstSequence` zurückgegeben.

Beispiele

- **altova:substitute-empty**(`(1,2,3), (4,5,6)`) gibt `(1,2,3)` zurück
- **altova:substitute-empty**(`(), (4,5,6)`) gibt `(4,5,6)`zurück

© 2019-2025 Altova GmbH

12.2.2.1.8 XPath/XQuery-Funktionen: String

Die folgenden XPath/XQuey-Erweiterungsfunktionen für Strings werden in der aktuellen Version Ihres Altova-Produkts unterstützt und bieten Zusatzfunktionalitäten für die Verarbeitung von Daten. Die Funktionen in diesem Abschnitt können mit dem **XPath 3.0-** und **XQuery 3.0**-Prozessor von Altova verwendet werden. Sie stehen im Zusammenhang mit XPath/XQuery zur Verfügung.

Anmerkung zur Benennung von Funktionen und zur Anwendbarkeit der Sprache

Altova-Erweiterungsfunktionen können in XPath/XQuery-Ausdrücken verwendet werden. Dadurch stehen neben den Funktionen in der Standardbibliothek der XPath-, XQuery- und XSLT-Funktionen zusätzliche Funktionen zur Verfügung. Die Altova-Erweiterungsfunktionen befinden sich im **Altova-Erweiterungsfunktions-Namespace, `http://www.altova.com/xslt-extensions`** und sind in diesem Abschnitt mit dem Präfix **`altova:`**, das als an diesen Namespace gebunden angenommen wird, gekennzeichnet. Beachten Sie, dass manche Funktionen in zukünftigen Versionen Ihres Produkts eventuell nicht mehr unterstützt werden oder dass sich das Verhalten einzelner Funktionen ändern kann. Um zu sehen, welche Altova-Erweiterungsfunktionen unterstützt werden, lesen Sie bitte die Dokumentation zur jeweiligen Release.

XPath-Funktionen (in XPath-Ausdrücken in XSLT verwendet):	XP1 XP2 XP3.1
XSLT-Funktionen (in XPath-Ausdrücken in XSLT verwendet):	XSLT1 XSLT2 XSLT3
XQuery-Funktionen (in XQuery-Ausdrücken in XQuery verwendet):	XQ1 XQ3.1

camel-case [altova:]

`altova:camel-case(InputString` *`als xs:string`*`)` als **`xs:string`** XP3.1 XQ3.1
Gibt den Input-String **`InputString`** in CamelCase zurück. Der String wird mit Hilfe der Regular Expression **`'\s'`** (welches ein Kürzel für das Leerzeichen ist) analysiert. Das erste Zeichen nach einem Leerzeichen oder einer Sequenz aufeinander folgender Leerzeichen, das kein Leerzeichen ist, wird mit einem Großbuchstaben geschrieben. Das erste Zeichen im Ausgabestring wird mit einem Großbuchstaben geschrieben.

Beispiele

- **`altova:camel-case`**`("max")` gibt `Max` zurück
- **`altova:camel-case`**`("max max")` gibt `Max Max` zurück
- **`altova:camel-case`**`("file01.xml")` gibt `File01.xml` zurück
- **`altova:camel-case`**`("file01.xml file02.xml")` gibt `File01.xml File02.xml` zurück
- **`altova:camel-case`**`("file01.xml  file02.xml")` gibt `File01.xml  File02.xml` zurück
- **`altova:camel-case`**`("file01.xml -file02.xml")` gibt `File01.xml -file02.xml` zurück

`altova:camel-case(InputString` *`als xs:string`*`,` **`SplitChars`** *`als xs:string`*`,` **`IsRegex`** *`als xs:boolean`*`)` als **`xs:string`** XP3.1 XQ3.1
Konvertiert den Input-String **`InputString`** in CamelCase, indem anhand von **`SplitChars`** festgelegt wird, welche(s) Zeichen die nächste Konvertierung in Großbuchstaben auslöst. **`SplitChars`** wird als Regular Expression verwendet, wenn **`IsRegex = true()`** oder als einfache Zeichen, wenn **`IsRegex = false()`**. Das erste Zeichen im Ausgabestring wird mit einem Großbuchstaben geschrieben.

Beispiele

- **`altova:camel-case`**`("setname getname", "set|get", true())` gibt `setName getName` zurück

- **altova:camel-case**("altova\documents\testcases", "\", false()) gibt Altova\Documents\Testcases zurück

char [altova:]

altova:char(Position *as xs:integer***)** als **xs:string** XP3.1 XQ3.1
Gibt einen String zurück, der das Zeichen an der durch das Argument Position definierten Position enthält. Dieses Zeichen wird durch Konvertierung des Werts des Kontextelements in xs:string ermittelt. Der Ergebnisstring ist leer, wenn an dem durch das Position Argument gelieferten Index kein Zeichen vorhanden ist.

Beispiele

Wenn das Kontextelement **1234ABCD** lautet:

- **altova:char**(2) gibt 2 zurück
- **altova:char**(5) gibt A zurück
- **altova:char**(9) gibt den leeren String zurück.
- **altova:char**(-2) gibt den leeren String zurück.

altova:char(InputString *als xs:string*, **Position** *als xs:integer***)** als **xs:string** XP3.1 XQ3.1
Gibt einen String zurück, der das Zeichen enthält, das sich in dem als InputString Argument gelieferten String an der durch das Argument Position definierten Position befindet. Der Ergebnisstring ist leer, wenn an dem durch das Position Argument gelieferten Index kein Zeichen vorhanden ist.

Beispiele

- **altova:char**("2014-01-15", 5) gibt - zurück
- **altova:char**("USA", 1) gibt U zurück
- **altova:char**("USA", 1) gibt den leeren String zurück.
- **altova:char**("USA", -2) gibt den leeren String zurück.

create-hash-from-string[altova:]

altova:create-hash-from-string(InputString *als xs:string***)** als **xs:string** XP2 XQ1 XP3.1 XQ3.1
altova:create-hash-from-string(InputString *als xs:string*, **HashAlgo** *als xs:string***)** als **xs:string** XP2 XQ1 XP3.1 XQ3.1
Generiert anhand von InputString mit Hilfe des durch das Argument HashAlgo definierten Hash-Algorithmus einen Hash-String. Es können die folgenden Hash-Algorithmen definiert werden (in Groß- oder Kleinbuchstaben): **MD5**, **SHA-1**, **SHA-224**, **SHA-256**, **SHA-384**, **SHA-512**. Wenn das zweite Argument nicht definiert ist (*siehe erste Signatur oben*), wird der Hash-Algorithmus **SHA-256** verwendet.

Beispiele

- **altova:create-hash-from-string**('abc') gibt einen Hash-String zurück, der mit Hilfe des Hash-Algorithmus **SHA-256** generiert wurde.
- **altova:create-hash-from-string**('abc', 'md5') gibt einen Hash-String zurück, der mit Hilfe des Hash-Algorithmus **MD5** generiert wurde.
- **altova:create-hash-from-string**('abc', 'MD5') gibt einen Hash-String zurück, der mit Hilfe des Hash-Algorithmus **MD5** generiert wurde.

© 2019-2025 Altova GmbH

first-chars [altova:]

altova:first-chars(X-Number *as xs:integer*) als **xs:string** XP3.1 XQ3.1
Gibt einen String zurück, der die ersten x Zeichen (bezeichnet durch `X-Number`) des String enthält, der durch Konvertierung des Werts des Kontextelements in `xs:string` erzeugt wird.

Beispiele

Wenn das Kontextelement **1234ABCD** lautet:

- **altova:first-chars**(2) gibt 12 zurück
- **altova:first-chars**(5) gibt 1234A zurück
- **altova:first-chars**(9) gibt 1234ABCD zurück

altova:first-chars(InputString *als xs:string*, **X-Number** *als xs:integer*) als **xs:string** XP3.1 XQ3.1
Gibt einen String zurück, der die ersten x Zeichen (bezeichnet durch `X-Number`) des String enthält, das als das Argument `InputString` angegeben ist.

Beispiele

- **altova:first-chars**("2014-01-15", 5) gibt 2014- zurück
- **altova:first-chars**("USA", 1) gibt U zurück

format-string [altova:]

altova:format-string(InputString *als xs:string*, **FormatSequence** *als item()**) als **xs:string** XP3.1 XQ3.1
Der Input String (erstes Argument) enthält Positionsparameter (%1, %2, usw.). Jeder Parameter wird durch das String-Element ersetzt, das sich in der (als zweites Argument bereitgestellten) Formatsequenz an der entsprechenden Position befindet. Daher ersetzt des erste Element in der Formatsequenz den Positionsparameter %1, das zweite den Positionsparameter %2, usw. Die Funktion gibt diesen formatierten String zurück, der die Ersetzungen enthält. Wenn für einen Positionsparameter kein String existiert, wird der Positionsparameter selbst zurückgegeben. Dies kommt vor, wenn der Index eines Positionsparameters größer als die Anzahl der Elemente in der Formatsequenz ist.

Beispiele

- **altova:format-string**('Hello %1, %2, %3', ('Jane','John','Joe')) gibt "Hello Jane, John, Joe" zurück.
- **altova:format-string**('Hello %1, %2, %3', ('Jane','John','Joe', 'Tom')) gibt "Hello Jane, John, Joe" zurück.
- **altova:format-string**('Hello %1, %2, %4', ('Jane','John','Joe', 'Tom')) gibt "Hello Jane, John, Tom" zurück.
- **altova:format-string**('Hello %1, %2, %4', ('Jane','John','Joe')) gibt "Hello Jane, John, %4" zurück.

last-chars [altova:]

altova:last-chars(X-Number *als xs:integer*) als **xs:string** XP3.1 XQ3.1
Gibt einen String zurück, der die letzten x Zeichen (bezeichnet durch `X-Number`) des String enthält, der durch Konvertierung des Werts des Kontextelements in `xs:string` erzeugt wird.

Beispiele

Wenn das Kontextelement **1234ABCD** lautet:

- **altova:last-chars**(2) gibt CD zurück
- **altova:last-chars**(5) gibt 4ABCD zurück
- **altova:last-chars**(9) gibt 1234ABCD zurück

altova:last-chars(InputString *als xs:string*, **X-Number** *als xs:integer*) als **xs:string** XP3.1 XQ3.1

Gibt einen String zurück, der die letzten x Zeichen (bezeichnet durch X-Number) des String enthält, das als das Argument InputString angegeben ist.

Beispiele

- **altova:last-chars**("2014-01-15", 5) gibt 01-15- zurück
- **altova:last-chars**("USA", 10) gibt USA zurück

pad-string-left [altova:]

altova:pad-string-left(StringToPad *als xs:string*, **Repeats** *als xs:integer*, **PadCharacter** *als xs:string*) als **xs:string** XP3.1 XQ3.1

Das Argument PadCharacter ist ein einzelnes Zeichen. Es wird links vom String als Auffüllzeichen eingefügt, um die Anzahl der Zeichen in StringToPad zu erhöhen, damit diese Anzahl dem Ganzzahlwert des Arguments StringLength entspricht. Das Argument StringLength kann jeden beliebigen (positiven oder negativen) Ganzzahlwert haben, Auffüllzeichen werden aber nur verwendet, wenn der Wert von StringLength größer als die Anzahl der Zeichen in StringToPad ist. Wenn StringToPad mehr Zeichen als der Wert von StringLength hat, bleibt StringToPad unverändert.

Beispiele

- **altova:pad-string-left**('AP', 1, 'Z') gibt 'AP' zurück
- **altova:pad-string-left**('AP', 2, 'Z') gibt 'AP' zurück
- **altova:pad-string-left**('AP', 3, 'Z') gibt 'ZAP' zurück
- **altova:pad-string-left**('AP', 4, 'Z') gibt 'ZZAP' zurück
- **altova:pad-string-left**('AP', -3, 'Z') gibt 'AP' zurück
- **altova:pad-string-left**('AP', 3, 'YZ')gibt einen Fehler zurück, dass das Auffüllzeichen zu lang ist

pad-string-right [altova:]

altova:pad-string-right(StringToPad *als xs:string*, **Repeats** *als xs:integer*, **PadCharacter** *als xs:string*) als **xs:string** XP3.1 XQ3.1

Das Argument PadCharacter ist ein einzelnes Zeichen. Es wird rechts vom String als Auffüllzeichen eingefügt, um die Anzahl der Zeichen in StringToPad zu erhöhen, damit diese Anzahl dem Ganzzahlwert des Arguments StringLength entspricht. Das Argument StringLength kann jeden beliebigen (positiven oder negativen) Ganzzahlwert haben, Auffüllzeichen werden aber nur verwendet, wenn der Wert von StringLength größer als die Anzahl der Zeichen in StringToPad ist. Wenn StringToPad mehr Zeichen als der Wert von StringLength hat, bleibt StringToPad unverändert.

Beispiele

- **altova:pad-string-right**('AP', 1, 'Z') gibt 'AP' zurück
- **altova:pad-string-right**('AP', 2, 'Z') gibt 'AP' zurück

© 2019-2025 Altova GmbH

- `altova:pad-string-right('AP', 3, 'Z')` gibt `'ZAP'` zurück
- `altova:pad-string-right('AP', 4, 'Z')` gibt `'ZZAP'` zurück
- `altova:pad-string-right('AP', -3, 'Z')` gibt `'AP'` zurück
- `altova:pad-string-right('AP', 3, 'YZ')`gibt einen Fehler zurück, dass das Auffüllzeichen zu lang ist

repeat-string [altova:]

altova:repeat-string(InputString *als xs:string*, **Repeats** *als xs:integer*) als **xs:string** XP2 XQ1 XP3.1 XQ3.1

Generiert einen String, der sich zusammensetzt aus dem ersten `InputString`-Argument, das die Anzahl der `Repeats` wiederholt wird.

Beispiele

- `altova:repeat-string("Altova #", 3)` gibt `"Altova #Altova #Altova #"` zurück

substring-after-last [altova:]

altova:substring-after-last(MainString *als xs:string*, **CheckString** *als xs:string*) als **xs:string** XP3.1 XQ3.1

Falls in `MainString` `CheckString` gefunden wird, so wird der Substring zurückgegeben, der in `MainString` nach `CheckString` steht. Falls `CheckString` in `MainString` nicht gefunden wird, so wird der leere String zurückgegeben. Wenn `CheckString` ein leerer String ist, so wird der gesamte `MainString` zurückgegeben. Falls `CheckString` mehrmals in `MainString`, vorkommt, so wird der Substring nach der letzten Instanz von `CheckString` zurückgegeben.

Beispiele

- `altova:substring-after-last('ABCDEFGH', 'B')` gibt `'CDEFGH'` zurück
- `altova:substring-after-last('ABCDEFGH', 'BC')` gibt `'DEFGH'`zurück
- `altova:substring-after-last('ABCDEFGH', 'BD')` gibt `''` zurück
- `altova:substring-after-last('ABCDEFGH', 'Z')` gibt `''` zurück
- `altova:substring-after-last('ABCDEFGH', '')` gibt `'ABCDEFGH'` zurück
- `altova:substring-after-last('ABCD-ABCD', 'B')` gibt `'CD'` zurück
- `altova:substring-after-last('ABCD-ABCD-ABCD', 'BCD')` gibt `''` zurück

substring-before-last [altova:]

altova:substring-before-last(MainString *as xs:string*, **CheckString** *as xs:string*) **as xs:string** XP3.1 XQ3.1

Falls in `MainString` `CheckString` gefunden wird, so wird der Substring zurückgegeben, der in `MainString` vor `CheckString` steht. Falls `CheckString` in `MainString` nicht gefunden wird, so wird der leere String zurückgegeben. Wenn `CheckString` ein leerer String ist, so wird der gesamte `MainString` zurückgegeben. Falls `CheckString` mehrmals in `MainString`, vorkommt, so wird der Substring vor der letzten Instanz von `CheckString` zurückgegeben.

Beispiele

- `altova:substring-before-last('ABCDEFGH', 'B')` gibt `'A'` zurück
- `altova:substring-before-last('ABCDEFGH', 'BC')` gibt `'A'` zurück

- **altova:substring-before-last**('ABCDEFGH', 'BD') gibt '' zurück
- **altova:substring-before-last**('ABCDEFGH', 'Z') gibt '' zurück
- **altova:substring-before-last**('ABCDEFGH', '') gibt '' zurück
- **altova:substring-before-last**('ABCD-ABCD', 'B') gibt 'ABCD-A' zurück
- **altova:substring-before-last**('ABCD-ABCD-ABCD', 'ABCD') gibt 'ABCD-ABCD-' zurück

substring-pos [altova:]

altova:substring-pos(StringToCheck *als xs:string*, **StringToFind** *als xs:string*) als **xs:integer** XP3.1 XQ3.1

Gibt die Zeichenposition der ersten Instanz von StringToFind im String StringToCheck zurück. Die Zeichenposition wird in Form einer Ganzzahl angegeben. Das erste Zeichen von StringToCheck hat die Position 1. Wenn StringToFind in StringToCheck nicht vorkommt, wird die Ganzzahl 0 zurückgegeben. Um den String auf eine zweite oder eine weiter hinten folgende Instanz von StringToCheck zu überprüfen, verwenden Sie die nächste Signatur dieser Funktion.

Beispiele

- **altova:substring-pos**('Altova', 'to') gibt 3 zurück
- **altova:substring-pos**('Altova', 'tov') gibt 3 zurück
- **altova:substring-pos**('Altova', 'tv') gibt 0 zurück
- **altova:substring-pos**('AltovaAltova', 'to') gibt 3 zurück

altova:substring-pos(StringToCheck *als xs:string*, **StringToFind** *als xs:string*, **Integer** *als xs:integer*) als **xs:integer** XP3.1 XQ3.1

Gibt die Zeichenposition von StringToFind im String StringToCheck zurück. Die Suche nach StringToFind beginnt an der durch das Argument Integer angegebenen Zeichenposition; der Zeichen-Substring vor dieser Position wird nicht durchsucht. Die zurückgegebene Ganzzahl gibt jedoch die Position des gefundenen String innerhalb des *gesamten* String StringToCheck an. Diese Signatur dient dazu, die zweite oder eine weiter hinten folgende Position eines String zu finden, der mehrmals in StringToCheck vorkommt. Wenn StringToFind in StringToCheck nicht vorkommt, wird die Ganzzahl 0 zurückgegeben.

Beispiele

- **altova:substring-pos**('Altova', 'to', 1) gibt 3 zurück
- **altova:substring-pos**('Altova', 'to', 3) gibt 3 zurück
- **altova:substring-pos**('Altova', 'to', 4) gibt 0 zurück
- **altova:substring-pos**('Altova-Altova', 'to', 0) gibt 3 zurück
- **altova:substring-pos**('Altova-Altova', 'to', 4) gibt 10 zurück

trim-string [altova:]

altova:trim-string(InputString *als xs:string*) als **xs:string** XP3.1 XQ3.1

Diese Funktion verwendet ein xs:string Argument, entfernt alle voran- und nachgestellten Leerzeichen und gibt einen "getrimmten" xs:string zurück.

Beispiele

- **altova:trim-string**(" Hello World ") gibt "Hello World" zurück
- **altova:trim-string**("Hello World ") gibt "Hello World" zurück
- **altova:trim-string**(" Hello World") gibt "Hello World" zurück

© 2019-2025 Altova GmbH

- **altova:trim-string**("Hello World") gibt "Hello World" zurück
- **altova:trim-string**("Hello World") gibt "Hello World" zurück

trim-string-left [altova:]

altova:trim-string-left(InputString *als xs:string***)** als **xs:string** XP3.1 XQ3.1
Diese Funktion verwendet ein xs:string Argument, entfernt alle vorangestellten Leerzeichen und gibt einen "links getrimmten" xs:string zurück.

Beispiele

- **altova:trim-string-left**(" Hello World ") gibt "Hello World " zurück
- **altova:trim-string-left**("Hello World ") gibt "Hello World " zurück
- **altova:trim-string-left**(" Hello World") gibt "Hello World" zurück
- **altova:trim-string-left**("Hello World") gibt "Hello World" zurück
- **altova:trim-string-left**("Hello World") gibt "Hello World" zurück

trim-string-right [altova:]

altova:trim-string-right(InputString *als xs:string***)** als **xs:string** XP3.1 XQ3.1
Diese Funktion verwendet ein xs:string Argument, entfernt alle nachgestellten Leerzeichen und gibt einen "rechts getrimmten" xs:string zurück.

Beispiele

- **altova:trim-string-right**(" Hello World ") gibt " Hello World" zurück
- **altova:trim-string-right**("Hello World ") gibt "Hello World" zurück
- **altova:trim-string-right**(" Hello World") gibt " Hello World" zurück
- **altova:trim-string-right**("Hello World") gibt "Hello World" zurück
- **altova:trim-string-right**("Hello World") gibt "Hello World" zurück

12.2.2.1.9 XPath/XQuery-Funktionen: Diverse Funktionen

Die folgenden XPath/XQuery-Funktionen für allgemeine Zwecke werden in der aktuellen Version von MapForce unterstützt und können in (i) in einem XSLT-Kontext in XPath-Ausdrücken oder (ii) in einem XQuery-Dokument in XQuery-Ausdrücken verwendet werden.

Anmerkung zur Benennung von Funktionen und zur Anwendbarkeit der Sprache

Altova-Erweiterungsfunktionen können in XPath/XQuery-Ausdrücken verwendet werden. Dadurch stehen neben den Funktionen in der Standardbibliothek der XPath-, XQuery- und XSLT-Funktionen zusätzliche Funktionen zur Verfügung. Die Altova-Erweiterungsfunktionen befinden sich im **Altova-Erweiterungsfunktions-Namespace, http://www.altova.com/xslt-extensions** und sind in diesem Abschnitt mit dem Präfix **altova:**, das als an diesen Namespace gebunden angenommen wird, gekennzeichnet. Beachten Sie, dass manche Funktionen in zukünftigen Versionen Ihres Produkts eventuell nicht mehr unterstützt werden oder dass sich das Verhalten einzelner Funktionen ändern kann. Um zu sehen, welche Altova-Erweiterungsfunktionen unterstützt werden, lesen Sie bitte die Dokumentation zur jeweiligen Release.

XPath-Funktionen (in XPath-Ausdrücken in XSLT verwendet):	XP1 XP2 XP3.1
XSLT-Funktionen (in XPath-Ausdrücken in XSLT verwendet):	XSLT1 XSLT2 XSLT3
XQuery-Funktionen (in XQuery-Ausdrücken in XQuery verwendet):	XQ1 XQ3.1

decode-string [altova:]

altova:decode-string(Input *als xs:base64Binary*) **als xs:string** XP3.1 XQ3.1
altova:decode-string(Input *als xs:base64Binary*, **Encoding** *als xs:string*) **als xs:string** XP3.1 XQ3.1
Dekodiert den angegebenen base64Binary-Input anhand der definierten Kodierung zu einem String. Wenn keine Kodierung definiert ist, wird die UTF-8-Kodierung verwendet. Die folgenden Kodierungen werden unterstützt: `US-ASCII, ISO-8859-1, UTF-16, UTF-16LE, UTF-16BE, ISO-10646-UCS2, UTF-32, UTF-32LE, UTF-32BE, ISO-10646-UCS4`

Beispiele

- **altova:decode-string**(`$XML1/MailData/Meta/b64B`) gibt den base64Binary-Input als UTF-8-kodierten String zurück.
- **altova:decode-string**(`$XML1/MailData/Meta/b64B, "UTF-8"`) gibt den base64Binary-Input als UTF-8-kodierten String zurück.
- **altova:decode-string**(`$XML1/MailData/Meta/b64B, "ISO-8859-1"`) gibt den base64Binary-Input als ISO-8859-1-kodierten String zurück.

encode-string [altova:]

altova:encode-string(InputString *als xs:string*) **als xs:base64Binaryinteger** XP3.1 XQ3.1
altova:encode-string(InputString *als xs:string*, **Encoding** *als xs:string*) **als xs:base64Binaryinteger** XP3.1 XQ3.1
Kodiert den angegebenen String gemäß der definierten Kodierung, falls eine angegeben wird. Wenn keine Kodierung definiert ist, wird die UTF-8-Kodierung verwendet. Der kodierte String wird in base64Binary-Zeichen konvertiert und es wird der konvertierte base64Binary-Wert zurückgegeben. Anfangs wird die UTF-8-Kodierung unterstützt. Die Unterstützung wird auf die folgenden Kodierungen ausgeweitet werden: `US-ASCII, ISO-8859-1, UTF-16, UTF-16LE, UTF-16BE, ISO-10646-UCS2, UTF-32, UTF-32LE, UTF-32BE, ISO-10646-UCS4`

Beispiele

- **altova:encode-string**(`"Altova"`) gibt das base64Binary-Äquivalent des UTF-8-kodierten String `"Altova"` zurück.
- **altova:encode-string**(`"Altova", "UTF-8"`) gibt das base64Binary-Äquivalent des UTF-8-kodierten String `"Altova"` zurück.

get-temp-folder [altova:]

altova:get-temp-folder() als `xs:string` XP2 XQ1 XP3.1 XQ3.1
Diese Funktion hat kein Argument. Sie gibt den Pfad zum temporären Ordner des aktuellen Benutzers zurück.

© 2019-2025 Altova GmbH

Beispiele

- **altova:get-temp-folder**() würde auf einem Windows-Rechner z.B. den folgenden Pfad als `xs:string` zurückgeben: `C:\Users\<UserName>\AppData\Local\Temp\`.

generate-guid [altova:]

altova:generate-guid() als **xs:string** XP2 XQ1 XP3.1 XQ3.1
Generiert einen eindeutigen String GUID-String.

Beispiele

- **altova:generate-guid**() gibt (z.B.) `85F971DA-17F3-4E4E-994E-99137873ACCD` zurück

high-res-timer [altova:]

altova:high-res-timer() als **xs:double** XP3.1 XQ3.1
Gibt einen hochauflösenden System-Timer-Wert in Sekunden zurück. Wenn in einem System ein hochauflösender Timer zur Verfügung steht, können bei Bedarf (z.B. bei Animationen und zur Ermittlung des exakten Codeausführungszeitpunkts) hochauflösende Zeitmessungen vorgenommen werden. Diese Funktion stellt die Auflösung des Hochauflösungs-Timers des Systems zur Verfügung.

Beispiele

- **altova:high-res-timer**() gibt eine Wert wie `'1.16766146154566E6'` zurück.

parse-html [altova:]

altova:parse-html(HTMLText *als xs:string***)** als **node()** XP3.1 XQ3.1
Das Argument `HTMLText` ist ein String, der den Text eines HTML-Dokuments enthält. Die Funktion erstellt anhand des Strings eine HTML-Struktur. Der bereitgestellte String kann das HTML-Element enthalten, muss dies aber nicht tun. In beiden Fällen ist das Root-Element der Struktur ein Element namens **HTML**. Sie sollten sicher stellen, dass der HTML-Code im bereitgestellten String gültiger HTML-Code ist.

Beispiel

- **altova:parse-html**("<html><head/><body><h1>Header</h1></body></html>") erstellt anhand des bereitgestellten Strings eine HTML-Struktur.

sleep[altova:]

altova:sleep(Millisecs *als xs:integer***)** als **empty-sequence()** XP2 XQ1 XP3.1 XQ3.1
Unterbricht die Ausführung der aktuellen Operation für die Anzahl der durch das Argument `Millisecs` angegebenen Millisekunden.

Beispiel

- **altova:sleep**(1000) unterbricht die Ausführung der aktuellen Operation für 1000 Millisekunden.

[Nach oben 611]

12.2.2.2 Diverse Erweiterungsfunktionen

Es gibt in Programmiersprachen wie Java und C# eine Reihe von fertigen Funktionen, die nicht als XQuery / XPath 2.0- oder XSLT-Funktionen zur Verfügung stehen. Ein gutes Beispiel dafür sind die mathematischen in Java verfügbaren Funktionen wie z.B. `sin()` und `cos()`. Stünden diese Funktionen für die Erstellung von XSLT Stylesheets und XQuery-Abfragen zur Verfügung, würde sich der Einsatzbereich von Stylesheets und Abfragen erweitern und die Erstellung von Stylesheets wäre viel einfacher. Der in einer Reihe von Altova-Produkten verwendete XSLT- und XQuery-Prozessor von Altova unterstützt die Verwendung von Erweiterungsfunktionen in Java[614] und .NET[624] sowie MSXSL Skripts für XSLT[630]. MSXSL-Skripts für XSLT[630] und die Altova-Erweiterungsfunktionen[534]. In diesem Abschnitt wird beschrieben, wie Sie Erweiterungsfunktionen und MSXSL-Skripts in Ihren XSLT Stylesheets und XQuery-Dokumenten verwenden können. Diese Beschreibungen finden Sie in den folgenden Abschnitten:

- Java-Erweiterungsfunktionen[614]
- .NET-Erweiterungsfunktionen[624]
- MSXSL-Skripts für XSLT[630]

Hauptsächlich werden dabei die folgenden beiden Punkte behandelt: (i) Wie Funktionen in den entsprechenden Bibliotheken aufgerufen werden; und (ii) welche Regeln beim Konvertieren von Argumenten in einem Funktionsaufruf in das erforderliche Format der Funktion befolgt werden und welche Regeln bei der Rückwärtskonvertierung (Funktionsresultat in XSLT/XQuery Datenobjekt) befolgt werden.

Voraussetzungen

Damit die Erweiterungsfunktionen unterstützt werden, muss auf dem Rechner, auf dem die XSLT-Transformation oder die XQuery-Ausführung stattfindet, eine Java Runtime-Umgebung (zum Aufrufen der Java-Funktionen) und ein .NET Framework 2.0 (Mindestvoraussetzung für Zugriff auf .NET-Funktionen) installiert sein oder es muss Zugriff auf eine solche bestehen.

12.2.2.2.1 Java-Erweiterungsfunktionen

Eine Java-Erweiterungsfunktion kann in einem XPath- oder XQuery-Ausdruck verwendet werden, um einen Java-Konstruktor oder eine Java-Methode (statisch oder Instanz) aufzurufen.

Ein Feld in einer Java-Klasse wird als Methode ohne Argument betrachtet. Bei einem Feld kann es sich um ein statisches Feld oder eine Instanz handeln. Wie man Felder aufruft, wird in den entsprechenden Unterabschnitten zu statischen Feldern und Instanzen beschrieben.

Dieser Abschnitt enthält die folgenden Unterabschnitte:

- Java: Konstruktoren[620]
- Java: Statische Methoden und statische Felder[621]
- Java: Instanzmethoden und Instanzfelder[621]
- Datentypen: XPath/XQuery in Java[622]
- Datentypen: Java in XPath/XQuery[623]

Beachten Sie die folgenden Punkte

- Wenn Sie ein Altova Desktop-Produkt verwenden, versucht die Altova-Applikation, den Pfad zur Java

 © 2019-2025 Altova GmbH

Virtual Machine automatisch zu ermitteln. Dazu wird zuerst (i) die Windows Registry und dann (ii) die **JAVA_HOME**-Umgebungsvariable gelesen. Sie können im Dialogfeld "Optionen" der Applikation auch einen benutzerdefinierten Pfad hinzufügen. Dieser Eintrag hat Vorrang vor allen anderen automatisch ermittelten Java VM-Pfaden.

- Wenn Sie ein Altova Server-Produkt auf einem Windows-Rechner ausführen, wird der Pfad zur Java Virtual Machine zuerst aus der Windows Registry ausgelesen. Falls dies nicht gelingt, wird die **JAVA_HOME**-Umgebungsvariable verwendet.
- Wenn Sie ein Altova Server-Produkt auf einem Linux- oder macOS-Rechner ausführen, sollten Sie sicherstellen, dass die **JAVA_HOME**-Umgebungsvariable ordnungsgemäß definiert ist und dass sich die Java Virtual Machine-Bibliothek (unter Windows die Datei **jvm.dll**) entweder im Verzeichnis **\bin\server** oder **\bin\client** befindet.

Form der Erweiterungsfunktion

Die Erweiterungsfunktion im XPath/XQuery-Ausdruck muss die folgenden Form haben `präfix:fname()`.

- Der Teil `präfix:` kennzeichnet die Erweiterungsfunktion als Java-Funktion, indem er die Erweiterungsfunktion mit einer in-scope Namespace-Deklaration verknüpft, deren URI mit `java:` beginnen muss (*Beispiele siehe unten*). Die Namespace-Deklaration sollte eine Java-Klasse bezeichnen, z.B:
  `xmlns:myns="java:java.lang.Math"`. Sie könnte aber auch einfach lauten:
  `xmlns:myns="java"` (ohne Doppelpunkt), wobei die Identifizierung der Java-Klasse dem `fname()` Teil der Erweiterungsfunktion überlassen bleibt.
- Der Teil `fname()` identifiziert die aufgerufene Java-Methode und liefert die Argumente für die Methode (*Beispiele siehe unten*). Wenn die durch das `prefix:` Teil identifizierte Namespace URI jedoch keine Java-Klasse bezeichnet (*siehe vorheriger Punkt),* dann sollte die Java-Klasse im `fname()` Teil vor der Klasse identifiziert werden und von der Klasse durch einen Punkt getrennt sein (*siehe zweites XSLT-Beispiel unten*).

Anmerkung: Die aufgerufene Klasse muss sich unter dem Klassenpfad des Rechners befinden.

XSLT-Beispiel

Hier sehen Sie zwei Beispiele dafür, wie eine statische Methode aufgerufen werden kann. Im ersten Beispiel ist der Klassenname (`java.lang.Math`) in der Namespace URI enthalten und darf daher nicht im `fname()` Teil enthalten sein. Im zweiten Beispiel liefert der `prefix:` Teil das Präfix `java:`, während der `fname()` Teil die Klasse sowie die Methode identifiziert.

```
<xsl:value-of xmlns:jMath="java:java.lang.Math"
              select="jMath:cos(3.14)" />

<xsl:value-of xmlns:jmath="java"
              select="jmath:java.lang.Math.cos(3.14)" />
```

Die in der Erweiterungsfunktion (im Beispiel oben `cos()`) angegebene Methode muss mit dem Namen einer öffentlichen statischen Methode in der angegebenen Java-Klasse (im Beispiel oben `java.lang.Math`) übereinstimmen.

XQuery-Beispiel

Hier sehen Sie ein XQuery-Beispiel, das dem XSLT-Beispiel oben ähnlich ist:

```
<cosine xmlns:jMath="java:java.lang.Math">
   {jMath:cos(3.14)}
</cosine>
```

Benutzerdefinierte Java-Klassen

Wenn Sie Ihre eigenen Java-Klassen erstellt haben, werden die Methoden in diesen Klassen unterschiedlich aufgerufen, je nachdem: (i) ob die Klassen über eine JAR-Datei oder eine Klassendatei aufgerufen werden, und (ii) ob sich diese Dateien (JAR oder Klasse) im aktuellen Verzeichnis befinden (im selben Verzeichnis wie das XSLT- oder XQuery-Dokument) oder nicht. Wie Sie diese Dateien finden, wird in den Abschnitten Benutzerdefinierte Klassendateien 616 und Benutzerdefinierte Jar-Dateien 619 beschrieben. Pfade zu Klassendateien, die sich nicht im aktuellen Verzeichnis befinden, und Pfade zu allen JAR-Dateien müssen jedoch angegeben werden.

12.2.2.2.1.1 Benutzerdefinierte Klassendateien

Wenn der Zugriff über eine Klassendatei erfolgt, gibt es vier Möglichkeiten:

- Die Klassendatei befindet sich in einem Paket. Die XSLT-oder XQuery-Datei befindet sich im selben Ordner wie das Java-Paket. (*Siehe Beispiel unten* 617.)
- Die Klassendatei befindet sich nicht in einem Paket. Die XSLT-oder XQuery-Datei befindet sich im selben Ordner wie die Klassendatei. (*Siehe Beispiel unten* 617.)
- Die Klassendatei befindet sich in einem Paket. Die XSLT-oder XQuery-Datei befindet sich in irgendeinem beliebig gewählten Ordner. (*Siehe Beispiel unten* 618.)
- Die Klassendatei befindet sich nicht in einem Paket. Die XSLT-oder XQuery-Datei befindet sich in irgendeinem beliebig gewählten Ordner. (*Siehe Beispiel unten* 618.)

Gesetzt der Fall, die Klassendatei befindet sich nicht in einem Paket, sondern im selben Ordner wie das XSLT- oder XQuery-Dokument, so muss der Dateipfad nicht angegeben werden, da alle Klassen im Ordner gefunden werden. Die Syntax zum Identifizieren einer Klasse lautet:

```
java:classname
```

wobei

`java:` angibt, dass eine benutzerdefinierte Java-Funktion aufgerufen wird; (Java-Klassen im aktuellen Verzeichnis werden standardmäßig geladen)
`classname` der Name der Klasse der erforderlichen Methode ist

die Klasse in einer Namespace URI identifiziert wird und der Namespace einem Methodenaufruf als Präfix vorangestellt wird.

 © 2019-2025 Altova GmbH

Klassendatei in einem Paket, XSLT/XQuery-Datei befindet sich im selben Ordner wie das Java-Paket

Im Beispiel unten wird die Methode `getVehicleType()`der Klasse `Car` des Pakets `com.altova.extfunc` aufgerufen. Das Paket `com.altova.extfunc` befindet sich im Ordner `JavaProject`. Die XSLT-Datei befindet sich ebenfalls im Ordner `JavaProject`.

```
<xsl:stylesheet version="2.0"
       xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
       xmlns:xs="http://www.w3.org/2001/XMLSchema"
       xmlns:fn="http://www.w3.org/2005/xpath-functions"
       xmlns:car="java:com.altova.extfunc.Car" >
<xsl:output exclude-result-prefixes="fn car xsl fo xs"/>

<xsl:template match="/">
   <a>
   <xsl:value-of select="car:getVehicleType()"/>
   </a>
</xsl:template>

</xsl:stylesheet>
```

Die Klassendatei wird referenziert, die XSLT/XQuery-Datei befindet sich im selben Ordner wie die Klassendatei

Im Beispiel unten wird die Methode `getVehicleType()` der Klasse `Car` des Pakets `com.altova.extfunc` aufgerufen. Angenommen, (i) die Klassendatei `Car` class befindet sich im folgenden Ordner: `JavaProject/com/altova/extfunc` und (ii) dieser Ordner ist der aktuelle Ordner im Beispiel unten. Die XSLT-Datei befindet sich ebenfalls im Ordner `JavaProject/com/altova/extfunc`.

```
<xsl:stylesheet version="2.0"
       xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
       xmlns:xs="http://www.w3.org/2001/XMLSchema"
       xmlns:fn="http://www.w3.org/2005/xpath-functions"
       xmlns:car="java:Car" >
<xsl:output exclude-result-prefixes="fn car xsl fo xs"/>

<xsl:template match="/">
   <a>
   <xsl:value-of select="car:getVehicleType()"/>
   </a>
</xsl:template>

</xsl:stylesheet>
```

Die Klassendatei befindet sich in einem Paket, die XSLT/XQuery-Datei befindet sich in einem beliebigen Ordner

Im Beispiel unten wird die Methode `getCarColor()`der Klasse `Car` des Pakets `com.altova.extfunc` aufgerufen. Das Paket `com.altova.extfunc` befindet sich im Ordner `JavaProject`. Die XSLT-Datei befindet sich in einem beliebigen Ordner. In diesem Fall muss der Pfad des Pakets mit der URI als Abfragestring definiert werden. Die Syntax lautet:

```
java:classname[?path=uri-of-classfile]
```

wobei

`java:` angibt, dass eine benutzerdefinierte Java-Funktion aufgerufen wird
`uri-of-classfile` die URI der Klassendatei ist
`classname` der Name der Klasse der benötigten Methode ist

die Klasse in einer Namespace URI identifiziert wird und der Namespace einem Methodenaufruf als Präfix vorangestellt wird. Im Beispiel unten sehen Sie, wie eine Klassendatei aufgerufen wird, die sich in einem anderen als dem aktuellen Verzeichnis befindet.

```
<xsl:stylesheet version="2.0"
       xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
       xmlns:xs="http://www.w3.org/2001/XMLSchema"
       xmlns:fn="http://www.w3.org/2005/xpath-functions"
       xmlns:car="java:com.altova.extfunc.Car?path=file:///C:/JavaProject/" >

<xsl:output exclude-result-prefixes="fn car xsl xs"/>

<xsl:template match="/">
   <xsl:variable name="myCar" select="car:new('red')" />
   <a><xsl:value-of select="car:getCarColor($myCar)"/></a>
</xsl:template>

</xsl:stylesheet>
```

Die Klassendatei wird referenziert, die XSLT/XQuery-Datei befindet sich in einem beliebigen Ordner

Im Beispiel unten wird die Methode `getCarColor()`der Klasse `Car` aufgerufen. Angenommen, die Klassendatei `Car` befindet sich im Ordner `C:/JavaProject/com/altova/extfunc`. Die XSLT-Datei befindet sich in einem beliebigen Ordner. Der Pfad der Klassendatei muss dann in der Namespace-URI als Abfragestring definiert werden. Die Syntax lautet:

```
java:classname[?path=<uri-of-classfile>]
```

wobei

`java:` angibt, dass eine benutzerdefinierte Java-Funktion aufgerufen wird
`uri-of-classfile` die URI der Klassendatei ist
`classname` der Name der Klasse der benötigten Methode ist

 © 2019-2025 Altova GmbH

die Klasse in einer Namespace URI identifiziert wird und der Namespace einem Methodenaufruf als Präfix vorangestellt wird. Im Beispiel unten sehen Sie, wie eine Klassendatei aufgerufen wird, die sich in einem anderen als dem aktuellen Verzeichnis befindet.

```
<xsl:stylesheet version="2.0"
       xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
       xmlns:xs="http://www.w3.org/2001/XMLSchema"
       xmlns:fn="http://www.w3.org/2005/xpath-functions"
       xmlns:car="java:Car?path=file:///C:/JavaProject/com/altova/extfunc/" >

<xsl:output exclude-result-prefixes="fn car xsl xs"/>

<xsl:template match="/">
   <xsl:variable name="myCar" select="car:new('red')" />
   <a><xsl:value-of select="car:getCarColor($myCar)"/></a>
</xsl:template>

</xsl:stylesheet>
```

Anmerkung: Wenn ein Pfad über eine Erweiterungsfunktion angegeben wird, wird er zum ClassLoader hinzugefügt.

12.2.2.2.1.2 Benutzerdefinierte Jar-Dateien

JAR-Dateien

Wenn der Zugriff über eine JAR-Datei erfolgt, muss die URI der JAR-Datei mit Hilfe der folgenden Syntax definiert werden:

```
xmlns:classNS="java:classname?path=jar:uri-of-jarfile!/"
```

Die Methode wird anschließend durch Verwendung des Präfix der Namespace URI aufgerufen, der die Klasse bezeichnet: `classNS:method()`

wobei im obigen Beispiel:

`java:` angibt, dass eine Java-Funktion aufgerufen wird
`classname` der Name der Klasse der benutzerdefinierten Klasse ist
`?` das Trennzeichen zwischen dem Klassennamen und dem Pfad ist
`path=jar:` angibt, dass es sich um einen Pfad zu einer JAR-Datei handelt
`uri-of-jarfile` die URI der jar-Datei angibt
`!/` das Trennzeichen am Ende des Pfades ist
`classNS:method()` der Aufruf der Methode ist

Alternativ dazu kann der Klassenname mit dem Methodenaufruf angegeben werden. Hier sehen Sie zwei Beispiele für die Syntax:

```
xmlns:ns1="java:docx.layout.pages?path=jar:file:///c:/projects/docs/docx.jar!/"
ns1:main()

xmlns:ns2="java?path=jar:file:///c:/projects/docs/docx.jar!/"
ns2:docx.layout.pages.main()
```

Hier sehen Sie ein komplettes XSLT-Beispiel, in dem eine JAR-Datei verwendet wird, um eine Java-Erweiterungsfunktion aufzurufen.:

```
<xsl:stylesheet version="2.0"
       xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
       xmlns:xs="http://www.w3.org/2001/XMLSchema"
       xmlns:fn="http://www.w3.org/2005/xpath-functions"
       xmlns:car="java?path=jar:file:///C:/test/Car1.jar!/" >
<xsl:output exclude-result-prefixes="fn car xsl xs"/>

<xsl:template match="/">
       <xsl:variable name="myCar" select="car:Car1.new('red')" />
       <a><xsl:value-of select="car:Car1.getCarColor($myCar)"/></a>
</xsl:template>

<xsl:template match="car"/>

</xsl:stylesheet>
```

Anmerkung: Wenn ein Pfad über eine Erweiterungsfunktion angegeben wird, wird er zum ClassLoader hinzugefügt.

12.2.2.2.1.3 Java: Konstruktoren

Eine Erweiterungsfunktion kann dazu verwendet werden, um einen Java-Konstruktor aufzurufen. Alle Konstruktoren werden mit der Pseudofunktion `new()` aufgerufen.

Wenn das Ergebnis eines Java-Konstruktors implizit in XPath/XQuery-Datentypen konvertiert werden kann 623, dann gibt die Java-Erweiterungsfunktion eine Sequenz zurück, bei der es sich um einem XPath/XQuery-Datentyp handelt. Wenn das Ergebnis eines Konstruktoraufrufs nicht in einen passenden XPath/XQuery-Datentyp konvertiert werden kann, dann erstellt der Konstruktor ein wrapped Java-Objekt mit einem Typ, der den Namen der Klasse hat, die dieses Java-Objekt zurückgibt. Wenn z.B. ein Konstruktor für die Klasse `java.util.Date` aufgerufen wird (`java.util.Date.new()`), so wird ein Objekt vom Typ `java.util.Date` zurückgegeben. Das lexikalische Format des zurückgegebenen Objekts stimmt unter Umständen nicht mit dem lexikalischen Format des XPath-Datentyps überein und der Wert müsste daher in das lexikalische Format des erforderlichen XPath-Datentyps und anschließend in den erforderlichen XPath-Datentyp konvertiert werden.

Ein von einem Konstruktor erstelltes Java-Objekt kann für zwei Zwecke verwendet werden:

- Es kann einer Variable zugewiesen werden:
  ```
  <xsl:variable name="currentdate" select="date:new()"
  xmlns:date="java:java.util.Date" />
  ```
- Es kann an eine Erweiterungsfunktion übergeben werden (*siehe* *Instanzmethode und Instanzfelder* 621):
  ```
  <xsl:value-of select="date:toString(date:new())" xmlns:date="java:java.util.Date" />
  ```

 © 2019-2025 Altova GmbH

12.2.2.2.1.4 Java: Statische Methoden und statische Felder

Eine statische Methode wird direkt über ihren Java-Namen und durch Angabe der Argumente für die Methode aufgerufen. Statische Felder (Methoden, die keine Argumente haben), wie z.B. die Konstantenwertfelder `E` und `PI` werden ohne Angabe eines Arguments aufgerufen.

XSLT-Beispiele

Hier sehen Sie einige Beispiele dafür, wie statische Methoden und Felder aufgerufen werden können:

```
<xsl:value-of xmlns:jMath="java:java.lang.Math"
              select="jMath:cos(3.14)" />

<xsl:value-of xmlns:jMath="java:java.lang.Math"
              select="jMath:cos( jMath:PI() )" />

<xsl:value-of xmlns:jMath="java:java.lang.Math"
              select="jMath:E() * jMath:cos(3.14)" />
```

Beachten Sie, dass die Erweiterungsfunktionen die Form `prefix:fname()` haben. Das Präfix ist in allen drei Fällen `jMath:`. Es ist mit der Namespace URI `java:java.lang.Math` verknüpft. (Die Namespace URI muss mit `java:`. beginnen. In den obigen Beispielen wurde es um den Klassennamen erweitert (`java.lang.Math`).) Der Teil `fname()` der Erweiterungsfunktionen muss mit dem Namen der öffentlichen Klasse (z.B. `java.lang.Math`) gefolgt vom Namen einer öffentlichen statischen Methode mit ihrem/ihren Argument(en) (wie z.B.`(3.14)`) oder einem öffentlichen statischen Feld (z.B. `PI()`) übereinstimmen.

In den obigen Beispielen wurde der Klassenname in die Namespace URI inkludiert. Wäre sie nicht in der Namespace URI enthalten, müsste sie in den `fname()` Teil der Erweiterungsfunktion inkludiert werden. Z.B:

```
<xsl:value-of xmlns:java="java:"
              select="java:java.lang.Math.cos(3.14)" />
```

XQuery-Beispiel

Ein ähnliches Beispiel in XQuery wäre:

```
<cosine xmlns:jMath="java:java.lang.Math">
   {jMath:cos(3.14)}
</cosine>
```

12.2.2.2.1.5 Java: Instanzmethoden und Instanzfelder

Bei einer Instanzmethode wird als erstes Argument eines Methodenaufrufs ein Java-Objekt an die Methode übergeben. Ein solches Java-Objekt würde normalerweise mit Hilfe einer Erweiterungsfunktion (z.B. eines Konstruktoraufrufs) oder eines Stylesheet-Parameters/einer Stylesheet-Variablen erstellt. Ein XSLT-Beispiel dafür wäre:

```
<xsl:stylesheet version="1.0" exclude-result-prefixes="date"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:date="java:java.util.Date"
  xmlns:jlang="java:java.lang">
  <xsl:param name="CurrentDate" select="date:new()"/>
  <xsl:template match="/">
      <enrollment institution-id="Altova School"
                date="{date:toString($CurrentDate)}"
                type="{jlang:Object.toString(jlang:Object.getClass( date:new() ))}">
      </enrollment>
  </xsl:template>
</xsl:stylesheet>
```

Im Beispiel oben wird der Wert des Node `enrollment/@type` folgendermaßen erstellt:

1. Es wird ein Objekt mit einem Konstruktor für die Klasse `java.util.Date` (mit dem Konstruktor `date:new()`) erstellt.
2. Dieses Java-Objekt wird als das Argument der Methode `jlang.Object.getClass` übergeben.
3. Das mit der Methode `getClass` abgerufene Objekt wird als das Argument an die Methode `jlang.Object.toString` übergeben.

Das Ergebnis (der Wert von `@type`) ist ein String, der den Wert `java.util.Date` hat.

Ein Instanzfeld unterscheidet sich theoretisch insofern von einer Instanzmethode, als es sich nicht um ein Java-Objekt per se handelt, das als Argument an das Instanzfeld übergeben wird. Stattdessen wird ein Parameter oder eine Variable als Argument übergeben. Der Parameter/die Variable kann allerdings selbst den Wert enthalten, der von einem Java-Objekt zurückgegeben wird. So erhält z.B. der Parameter `CurrentDate` den Wert, der von einem Konstruktor für die Klasse `java.util.Date` zurückgegeben wird. Dieser Wert wird anschließend als Argument an die Instanzmethode `date:toString` übergeben, um den Wert von `/enrollment/@date` bereitzustellen.

12.2.2.2.1.6 Datentypen: XPath/XQuery in Java

Wenn von einem XPath/XQuery-Ausdruck aus eine Java-Funktion aufgerufen wird, spielt der Datentyp der Argumente der Funktion eine wichtige Rolle, welche von mehreren Java-Klassen desselben Namens aufgerufen wird.

In Java gelten die folgenden Regeln:

- Wenn es mehr als eine Java-Methode mit demselben Namen gibt, jede aber eine andere Anzahl von Argumenten als die andere(n) hat, so wird die Java-Methode ausgewählt, die der Anzahl der Argumente im Funktionsaufruf am ehesten entspricht.
- Die XPath/XQuery-Datentypen "string", "number" und "boolean" (*siehe Liste unten*) werden implizit in einen entsprechenden Java-Datentyp konvertiert. Wenn der bereitgestellte XPath/XQuery-Datentyp in mehr als einen Java-Typ konvertiert werden kann (z.B: `xs:integer`), so wird jener Java-Typ ausgewählt, der für die ausgewählte Methode deklariert wurde. Wenn die aufgerufene Java-Methode z.B. `fx(decimal)` und der bereitgestellte XPath/XQuery-Datentyp `xs:integer` ist, so wird `xs:integer` in den Java-Datentyp `decimal` konvertiert.

In der Tabelle unten sehen Sie eine Liste der impliziten Konvertierungen der XPath/XQuery-Datentypen "string", "number" und "boolean"in Java-Datentypen.

 © 2019-2025 Altova GmbH

xs:string	java.lang.String
xs:boolean	boolean (primitive), java.lang.Boolean
xs:integer	int, long, short, byte, float, double und die Wrapper-Klassen davon wie z.B. java.lang.Integer
xs:float	float (primitive), java.lang.Float, double (primitive)
xs:double	double (primitive), java.lang.Double
xs:decimal	float (primitive), java.lang.Float, double(primitive), java.lang.Double

Die oben aufgelisteten Subtypen von XML-Schema-Datentypen (die in XPath und XQuery verwendet werden) werden ebenfalls in den/die Java-Typ(en), der/die dem übergeordneten Subtyp entsprechen, konvertiert.

In einigen Fällen ist es nicht möglich, auf Basis der verfügbaren Informationen die richtige Java-Methode auszuwählen. Nehmen Sie als Beispiel den folgenden Fall.

- Das bereitgestellte Argument ist ein xs:untypedAtomic Wert 10 und ist für die Methode mymethod(float) bestimmt.
- Es gibt jedoch eine weitere Methode in der Klasse, die ein Argument eines anderen Datentyps erhält: mymethod(double).
- Da die Methodennamen dieselben sind und der bereitgestellte Typ (xs:untypedAtomic) sowohl in float als auch double korrekt konvertiert werden könnte, kann es geschehen, dass xs:untypedAtomic in double anstelle von float konvertiert wird.
- Infolgedessen handelt es sich dann bei der ausgewählten Methode nicht um die benötigte Methode, sodass nicht das erwartete Ergebnis erzielt wird. Als Umgehungslösung können Sie eine benutzerdefinierte Methode mit einem anderen Namen erstellen und diese Methode verwenden.

Typen, die in der Liste oben nicht enthalten sind (z.B. xs:date), werden nicht konvertiert und generieren einen Fehler. Beachten Sie jedoch, dass es in einigen Fällen unter Umständen möglich ist, den benötigten Java-Typ mittels eines Java-Konstruktors zu erstellen.

12.2.2.2.1.7 Datentypen: Java in XPath/XQuery

Wenn eine Java-Methode einen Wert zurückgibt und der Datentyp des Werts "string", "numeric" oder "boolean" ist, wird anschließend in den entsprechenden XPath/XQuery-Typ konvertiert. So werden z.B. die Java-Datentypen javaa.lang.Boolean und boolean in xsd:boolean konvertiert.

Von Funktionen zurückgegebene eindimensionale Arrays werden zu einer Sequenz erweitert. Mehrdimensionale Arrays werden nicht konvertiert und sollten daher in einen Wrapper gesetzt werden.

Wenn ein wrapped Java-Objekt oder ein Datentyp zurückgegeben wird, bei dem es sich nicht um den Typ "string", "numeric" oder "boolean" handelt, können Sie sicherstellen, dass die Konvertierung in den benötigten XPath/XQuery-Typ erfolgt, indem Sie zuerst eine Java-Methode (e.g toString) verwenden, um das Java-Objekt in einen String zu konvertieren. In XPath/XQuery kann der String geändert werden, damit er der lexikalischen

Darstellung des benötigten Typs entspricht, und anschließend z.B. mit Hilfe des Ausdrucks `cast as` in den benötigten Typ konvertiert werden.

12.2.2.2.2 .NET-Erweiterungsfunktionen

Wenn Sie auf einem Windows-Rechner mit der .NET-Plattform arbeiten, können Sie Erweiterungsfunktionen verwenden, die in jeder beliebigen der .NET-Sprachen geschrieben wurden (z.B. C#). Eine .NET Erweiterungsfunktion kann in einem XPath- oder XQuery-Ausdruck verwendet werden, um einen Konstruktor, eine Eigenschaft oder Methode (statische oder Instanz) in einer .NET-Klasse aufzurufen.

Eine Eigenschaft einer .NET-Klasse wird mit der Syntax `get_PropertyName()` aufgerufen.

Dieser Abschnitt ist in die folgenden Unterabschnitte gegliedert:

- .NET: Konstruktoren 626
- .NET: Statische Methoden und statische Felder 627
- .NET: Instanzmethoden und Instanzfelder 628
- Datentypen: XPath/XQuery in .NET 629
- Datentypen: .NET in XPath/XQuery 630

Form der Erweiterungsfunktion

Die Erweiterungsfunktion im XPath/XQuery-Ausdruck muss die folgende Form haben `präfix:fname()`.

- Der Teil `präfix:` ist mit einer URI verknüpft, die die benötigte .NET-Klasse definiert.
- Der Teil `fname()` identifiziert den Konstruktor, die Eigenschaft oder die Methode (statisch oder Instanz) innerhalb der .NET-Klasse und liefert alle gegebenenfalls benötigten Argumente.

- Die URI muss mit `clitype:` beginnen (welches die Funktion als .NET-Erweiterungsfunktion kennzeichnet).
- Die Form `prefix:fname()` der Erweiterungsfunktion kann mit Systemklassen und mit Klassen in einer geladenen Assembly verwendet werden. Wenn eine Klasse allerdings geladen werden muss, müssen zusätzliche Parameter mit den benötigten Informationen bereitgestellt werden.

Parameter

Zum Laden einer Assembly werden die folgenden Parameter verwendet:

`asm`	Der Name der zu ladenden Assembly
`ver`	Die Versionsnummer: eine Maximalzahl von vier Ganzzahlen, die durch Punkte getrennt sind
`sn`	Das Key Token des Strong Name der Assembly (16 Hex-Stellen).
`from`	Eine URI gibt den Pfad der zu ladenden Assembly (DLL) an. Wenn die URI relativ ist, ist sie relativ zum XSLT- oder XQuery-Dokument. Wenn dieser Parameter vorhanden ist, werden alle anderen Parameter ignoriert.

 © 2019-2025 Altova GmbH

`partialname`	Der partielle Name der Assembly. Er wird für `Assembly.LoadWith.PartialName()` bereitgestellt, welches versuchen wird, die Assembly zu laden. Wenn `partialname` vorhanden ist, werden alle anderen Parameter ignoriert.
`loc`	Die Locale, z.B. `en-US`. Die Standardeinstellung ist `neutral`

Wenn die Assembly aus einer DLL geladen werden soll, verwenden Sie den `from` Parameter und lassen Sie den `sn` Parameter weg. Wenn die Assembly aus dem Global Assembly Cache (GAC) geladen werden soll, verwenden Sie den `sn` Parameter und lassen Sie den `from` Parameter weg.

Vor dem ersten Parameter muss ein Fragezeichen eingefügt werden. Parameter müssen durch ein Semikolon getrennt werden. Der Wert des Parameternamens wird durch ein Ist-Gleich-Zeichen angegeben (*siehe Beispiele unten*).

Beispiele für Namespace-Deklarationen

Ein Beispiel für eine Namespace Deklaration in XSLT, die die Systemklasse `System.Environment`: identifiziert.

```
xmlns:myns="clitype:System.Environment"
```

Ein Beispiel für eine Namespace Deklaration in XSLT, die die zu ladende Klasse als `Trade.Forward.Scrip`: identifiziert.

```
xmlns:myns="clitype:Trade.Forward.Scrip?asm=forward;version=10.6.2.1"
```

Ein Beispiel für eine Namespace-Deklaration in XQuery, die die Systemklasse `MyManagedDLL.testClass` identifiziert. Es werden zwei Klassen unterschieden:

1. Wenn die Assembly aus dem GAC geladen wird:
   `declare namespace cs="clitype:MyManagedDLL.testClass?asm=MyManagedDLL; ver=1.2.3.4;loc=neutral;sn=b9f091b72dccfba8";`

2. Wenn die Assembly aus der DLL geladen wird (vollständige und partielle Referenzen unten):
   `declare namespace cs="clitype:MyManagedDLL.testClass?from=file:///C:/Altova Projects/extFunctions/MyManagedDLL.dll;`

```
declare namespace cs="clitype:MyManagedDLL.testClass?from=MyManagedDLL.dll;
```

XSLT-Beispiel

Hier sehen Sie ein vollständiges XSLT-Beispiel, in dem Funktionen in der Systemklasse `System.Math`: aufgerufen werden:

```
<xsl:stylesheet version="2.0"
   xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
   xmlns:xs="http://www.w3.org/2001/XMLSchema"
   xmlns:fn="http://www.w3.org/2005/xpath-functions">
   <xsl:output method="xml" omit-xml-declaration="yes" />
   <xsl:template match="/">
      <math xmlns:math="clitype:System.Math">
```

```
        <sqrt><xsl:value-of select="math:Sqrt(9)"/></sqrt>
        <pi><xsl:value-of select="math:PI()"/></pi>
        <e><xsl:value-of select="math:E()"/></e>
        <pow><xsl:value-of select="math:Pow(math:PI(), math:E())"/></pow>
      </math>
   </xsl:template>
</xsl:stylesheet>
```

Die Namespace-Deklaration für das Element `math` verknüpft das Präfix `math:` mit der URI `clitype:System.Math`. Der Beginn der URI `clitype:` gibt an, dass danach entweder eine Systemklasse oder eine geladene Klasse definiert wird. Das Präfix `math:` im XPath-Ausdruck verknüpft die Erweiterungsfunktionen mit der URI (und durch Erweiterung der Klasse) `System.Math`. Die Erweiterungsfunktionen identifizieren Methoden in der Klasse `System.Math` und stellen Argumente bereit, wo dies erforderlich ist.

XQuery-Beispiel

Hier sehen Sie ein XQuery-Beispielfragment ähnlich dem XSLT-Beispiel oben:

```
<math xmlns:math="clitype:System.Math">
   {math:Sqrt(9)}
</math>
```

Wie beim XSLT-Beispiel weiter oben identifiziert die Namespace-Deklaration die .NET-Klasse, in diesem Fall eine Systemklasse. Der XQuery-Ausdruck identifiziert die aufzurufenden Methode und liefert das Argument.

12.2.2.2.2.1 .NET: Konstruktoren

Eine Erweiterungsfunktion kann verwendet werden, um einen .NET-Konstruktor aufzurufen. Alle Konstruktoren werden mit der Pseudofunktion `new()` aufgerufen. Wenn es mehrere Konstruktoren für eine Klasse gibt, wird der Konstruktor ausgewählt, der der Anzahl der bereitgestellten Argumente am ehesten entspricht. Wenn kein passender Konstruktor gefunden wird, der den bereitgestellten Argumenten entspricht, wird die Fehlermeldung `'No constructor found'` zurückgegeben.

Konstruktoren, die XPath/XQuery-Datentypen zurückgeben

Wenn das Ergebnis eines .NET-Konstruktors implizit in XPath/XQuery-Datentypen konvertiert werden kann 623, gibt die .NET-Erweiterungsfunktion eine Sequenz zurück, bei der es sich um einen XPath/XQuery-Datentyp handelt.

Konstruktoren, die .NET-Objekte zurückgeben

Wenn das Ergebnis eines .NET-Konstruktoraufrufs nicht in einen passenden XPath/XQuery-Datentyp konvertiert werden kann, erstellt der Konstruktor ein wrapped .NET-Objekt mit einem Typ, der der Name der Klasse ist, die dieses Objekt zurückgibt. Wenn z.B. ein Konstruktor für die Klasse `System.DateTime` aufgerufen wird (mit `System.DateTime.new()`), so wird ein Objekt mit dem Typ `System.DateTime` zurückgegeben.

Das lexikalische Format des zurückgegebenen Objekts stimmt unter Umständen nicht mit dem lexikalischen Format eines erforderlichen XPath-Datentyps überein. In solchen Fällen müsste der zurückgegebene Wert: (i)

 © 2019-2025 Altova GmbH

in das lexikalische Format des benötigten XPath-Datentyps konvertiert werden; und (ii) in den erforderlichen XPath-Datentyp konvertiert werden.

Ein von einem Konstruktor erstelltes .NET-Objekt kann für drei Zwecke verwendet werden:

- Es kann innerhalb einer Variable verwendet werden:
  ```
  <xsl:variable name="currentdate" select="date:new(2008, 4, 29)"
  xmlns:date="clitype:System.DateTime" />
  ```
- Es kann an eine Erweiterungsfunktion übergeben werden (*siehe* *Instanzmethode und Instanzfelder* 621):
  ```
  <xsl:value-of select="date:ToString(date:new(2008, 4, 29))"
  xmlns:date="clitype:System.DateTime" />
  ```
- Es kann in einen String, eine Zahl oder einen Booleschen Ausdruck konvertiert werden:
- ```
  <xsl:value-of select="xs:integer(date:get_Month(date:new(2008, 4, 29)))"
  xmlns:date="clitype:System.DateTime" />
  ```

12.2.2.2.2.2 .NET: Statische Methoden und statische Felder

Eine statische Methode wird direkt über ihren Namen und durch Angabe der Argumente für die Methode aufgerufen. Der im Aufruf verwendete Name muss exakt mit einer öffentlichen statischen Methode in der angegebenen Klasse übereinstimmen. Wenn der Methodenname und die Anzahl der in der Funktion angegebenen Argumente mit mehr als einer Methode in einer Klasse übereinstimmen, werden die Typen der bereitgestellten Argumente nach der besten Übereinstimmung überprüft. Wenn keine eindeutig passende Methode gefunden werden kann, wird ein Fehler ausgegeben.

Anmerkung: Ein Feld in einer .NET-Klasse wird als Methode ohne Argument betrachtet. Eine Eigenschaft wird mit der Syntax `get_PropertyName()` aufgerufen.

Beispiele

Ein XSLT-Beispiel, in dem Sie einen Methodenaufruf mit einem Argument (`System.Math.Sin(arg)`) sehen:

```
<xsl:value-of select="math:Sin(30)" xmlns:math="clitype:System.Math"/>
```

Ein XSLT-Beispiel, in dem Sie einen Aufruf eines Felds (wird als Methode ohne Argument betrachtet) sehen (`System.Double.MaxValue()`):

```
<xsl:value-of select="double:MaxValue()" xmlns:double="clitype:System.Double"/>
```

Ein XSLT-Beispiel, in dem Sie einen Aufruf einer Eigenschaft (Syntax ist `get_PropertyName()`) (`System.String()`) sehen:

```
<xsl:value-of select="string:get_Length('my string')"
xmlns:string="clitype:System.String"/>
```
```

Ein XQuery-Beispiel, in dem Sie einen Aufruf einer Methode mit einem Argument (`System.Math.Sin(arg)`) sehen:

```
<sin xmlns:math="clitype:System.Math">
   { math:Sin(30) }
</sin>
```

12.2.2.2.2.3 .NET: Instanzmethoden und Instanzfelder

Bei einer Instanzmethode wird als erstes Argument des Methodenaufrufs ein .NET-Objekt an die Methode übergeben. Dieses .NET-Objekt wird normalerweise mit Hilfe einer Erweiterungsfunktion (z.B. durch einen Konstruktoraufruf) oder einen Stylesheet-Parameter/eine Stylesheet-Variable erstellt. Ein XSLT-Beispiel dieser Art wäre:

```
<xsl:stylesheet version="2.0"
   xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
   xmlns:xs="http://www.w3.org/2001/XMLSchema"
   xmlns:fn="http://www.w3.org/2005/xpath-functions">
   <xsl:output method="xml" omit-xml-declaration="yes"/>
   <xsl:template match="/">
      <xsl:variable name="releasedate"
         select="date:new(2008, 4, 29)"
         xmlns:date="clitype:System.DateTime"/>
      <doc>
         <date>
            <xsl:value-of select="date:ToString(date:new(2008, 4, 29))"
               xmlns:date="clitype:System.DateTime"/>
         </date>
         <date>
            <xsl:value-of select="date:ToString($releasedate)"
               xmlns:date="clitype:System.DateTime"/>
         </date>
      </doc>
   </xsl:template>
</xsl:stylesheet>
```

Im Beispiel oben wird ein `System.DateTime` Konstruktor (`new(2008, 4, 29)`) verwendet, um ein .NET-Objekt vom Typ `System.DateTime` zu erstellen. Diese Objekt wird zweimal erstellt, einmal als Wert der Variablen `releasedate`, ein zweites Mal als das erste und einzige Argument der Methode `System.DateTime.ToString()`. Die Instanzmethode `System.DateTime.ToString()` wird zwei Mal aufgerufen, beide Male mit dem `System.DateTime` Konstruktor (`new(2008, 4, 29)`) als erstem und einzigem Argument. In einer dieser Instanzen wird die Variable `releasedate` verwendet, um das .NET-Objekt abzurufen.

Instanzmethoden und Instanzfelder

Der Unterschied zwischen einer Instanzmethode und einem Instanzfeld ist ein theoretischer. In einer Instanzmethode wird ein .NET-Objekt direkt als Argument übergeben; in einem Instanzfeld wird stattdessen ein Parameter oder eine Variable übergeben - auch wenn der Parameter bzw. die Variable selbst ein .NET-Objekt enthalten kann. So enthält z.B. die Variable `releasedate` im Beispiel oben ein .NET-Objekt und es ist diese

 © 2019-2025 Altova GmbH

Variable, die als das Argument von `ToString()` an den zweiten `date` Elementkonstruktor übergeben wird. Die `ToString()` Instanz im ersten `date` Element ist daher eine Instanzmethode, während die zweite als Instanzfeld betrachtet wird. Das in beiden Instanzen erzeugte Ergebnis ist jedoch dasselbe.

12.2.2.2.2.4 Datentypen: XPath/XQuery in .NET

Wenn in einem XPath/XQuery-Ausdruck eine .NET-Erweiterungsfunktion verwendet wird, spielen die Datentypen der Argumente der Funktion eine wichtige Rolle bei der Entscheidung, welche der vielen .NET-Methoden mit demselben Namen aufgerufen werden soll.

In .NET gelten die folgenden Regeln:

- Wenn es mehr als eine Methode mit demselben Namen in einer Klasse gibt, so stehen nur die Methoden zur Auswahl, die dieselbe Anzahl von Argumenten wie der Funktionsaufruf haben.
- Die XPath/XQuery-Datentypen "string", "number" und "boolean" (*siehe Liste unten*) werden implizit in einen entsprechenden .NET-Datentyp konvertiert. Wenn der bereitgestellte XPath/XQuery-Datentyp in mehr als einen .NET-Typ konvertiert werden kann (z.B: `xs:integer`), so wird jener .NET-Typ ausgewählt, der für die ausgewählte Methode deklariert wurde. Wenn die aufgerufene .NET-Methode z.B. `fx(double)` und der bereitgestellte XPath/XQuery-Datentyp `xs:integer` ist, so wird `xs:integer` in den .NET-Datentyp `double`

In der Tabelle unten sehen Sie eine Liste der impliziten Konvertierungen der XPath/XQuery-Datentypen "string", "number" und "boolean"in .NET-Datentypen.

`xs:string`	`StringValue, string`
`xs:boolean`	`BooleanValue, bool`
`xs:integer`	`IntegerValue, decimal, long, integer, short, byte, double, float`
`xs:float`	`FloatValue, float, double`
`xs:double`	`DoubleValue, double`
`xs:decimal`	`DecimalValue, decimal, double, float`

Die oben aufgelisteten Subtypen von XML-Schema-Datentypen (die in XPath und XQuery verwendet werden) werden ebenfalls in den/die .NET-Typ(en), der/die dem übergeordneten Subtyp entsprechen, konvertiert.

In einigen Fällen ist es nicht möglich, auf Basis der verfügbaren Informationen die richtige .NET-Methode auszuwählen. Nehmen Sie als Beispiel den folgenden Fall.

- Das bereitgestellte Argument ist ein `xs:untypedAtomic` Wert 10 und ist für die Methode `mymethod(float)` bestimmt.
- Es gibt jedoch eine weitere Methode in der Klasse, die ein Argument eines anderen Datentyps erhält: `mymethod(double)`.
- Da die Methodennamen dieselben sind und der bereitgestellte Typ (`xs:untypedAtomic`) sowohl in `float` als auch `double` korrekt konvertiert werden könnte, kann es geschehen, dass `xs:untypedAtomic` in `double` anstelle von `float` konvertiert wird.
- Infolgedessen handelt es sich dann bei der ausgewählten Methode nicht um die benötigte Methode, sodass nicht das erwartete Ergebnis erzielt wird. Als Umgehungslösung können Sie eine benutzerdefinierte Methode mit einem anderen Namen erstellen und diese Methode verwenden.

Typen, die in der Liste oben nicht enthalten sind (z.B. `xs:date`), werden nicht konvertiert und generieren einen Fehler.

12.2.2.2.2.5 Datentypen: .NET in XPath/XQuery

Wenn eine .NET-Methode einen Wert zurückgibt und der Datentyp des Werts "string", "numeric" oder "boolean" ist, wird er anschließend in den entsprechenden XPath/XQuery-Typ konvertiert. So wird z.B. der .NET-Datentyp `decimal` in `xsd:decimal` konvertiert.

Wenn ein .NET-Objekt oder ein Datentyp zurückgegeben wird, bei dem es sich nicht um den Typ "string", "numeric" oder "boolean" handelt, können Sie sicherstellen, dass die Konvertierung in den benötigten XPath/XQuery-Typ erfolgt, indem Sie zuerst eine .NET-Methode (z.B. `System.DateTime.ToString()`) verwenden, um das .NET-Objekt in einen String zu konvertieren. In XPath/XQuery kann der String geändert werden, damit er der lexikalischen Darstellung des benötigten Typs entspricht, und anschließend z.B. mit Hilfe des Ausdrucks `cast as` in den benötigten Typ konvertiert werden.

12.2.2.2.3 MSXSL-Skripts für XSLT

Das Element `<msxsl:script>` enthält benutzerdefinierte Funktionen und Variablen, die von XPath-Ausdrücken im XSLT-Stylesheet aufgerufen werden können. Das Element `<msxsl:script>` ist ein Element der obersten Ebene, d.h. es muss ein Child-Element von `<xsl:stylesheet>` oder `<xsl:transform>` sein.

Das Element `<msxsl:script>` muss sich im Namespace `urn:schemas-microsoft-com:xslt` (*siehe Beispiel unten*) befinden.

Scripting-Sprache und Namespace

Die im Block verwendete Scripting-Sprache wird im Attribut `language` des Elements `<msxsl:script>` definiert und der für Funktionsaufrufe von XPath-Ausdrücken aus zu verwendende Namespace wird durch das Attribut `implements-prefix` (*siehe unten*) identifiziert.

```
<msxsl:script language="scripting-language implements-prefix="user-namespace-prefix">

   function-1 or variable-1
   ...
   function-n or variable-n

</msxsl:script>
```

Das Element `<msxsl:script>` interagiert mit der Windows Scripting Runtime. Daher können nur Sprachen, die auf Ihrem Rechner installiert sind, im Element `<msxsl:script>` verwendet werden. **Um MXSL Scripts verwenden zu können muss die Plattform .NET Framework 2.0 oder höher installiert sein.** Folglich können die .NET Scripting Sprachen innerhalb des Elements `<msxsl:script>` verwendet werden.

 © 2019-2025 Altova GmbH

Das Attribut `language` akzeptiert dieselben Werte wie das Attribut `language` des HTML `<script>` Elements. Wenn das Attribut `language` nicht definiert ist, wird als Standardsprache Microsoft JScript verwendet.

Das Attribut `implements-prefix` erhält einen Wert, der ein Präfix eines deklarierten in-scope Namespace ist. Bei diesem Namespace handelt es sich normalerweise um einen Benutzer-Namespace, der für eine Funktionsbibliothek reserviert ist. Alle Funktionen und Variablen, die im Element `<msxsl:script>` definiert sind, werden sich im Namespace befinden, der durch das im Attribut `implements-prefix` definierte Präfixe identifiziert wird. Wenn eine Funktion von einem XPath-Ausdruck aus aufgerufen wird, muss sich der vollständig qualifizierte Funktionsname im selben Namespace wie die Funktionsdefinition befinden.

Beispiel

Hier sehen Sie ein Beispiel für ein vollständiges XSLT Stylesheet, in dem eine Funktion verwendet wird, die in einem `<msxsl:script>` Element definiert ist.

```
<?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>
```

Datentypen

Die Werte von Parametern, die an und aus dem Script-Block heraus übergeben werden, sind auf XPath-Datentypen beschränkt. Diese Einschränkung gilt nicht für Daten, die zwischen Funktionen und Variablen innerhalb des Script-Blocks übergeben werden.

Assemblies

Eine Assembly kann über das Element **`msxsl:assembly`** in das Script importiert werden. Die Assembly wird über einen Namen oder eine URL identifiziert. Die Assembly wird beim Kompilieren des Stylesheet importiert. Hier sehen Sie ein einfaches Beispiel, wie das Element `msxsl:assembly` zu verwenden ist.

```
<msxsl:script>
   <msxsl:assembly name="myAssembly.assemblyName" />
   <msxsl:assembly href="pathToAssembly" />

   ...

</msxsl:script>
```

Der Assembly-Name kann ein vollständiger Name sein, wie z.B.:

```
"system.Math, Version=3.1.4500.1 Culture=neutral PublicKeyToken=a46b3f648229c514"
```

oder ein Kurzname wie z.B. `"myAssembly.Draw"`.

Namespaces

Namespaces können mit dem Element **`msxsl:using`** deklariert werden. Auf diese Art können Assembly-Klassen ohne ihre Namespaces in das Script geschrieben werden, wodurch Sie sich das mühsame Eintippen ersparen. Hier sehen Sie, wie das Element `msxsl:using` verwendet wird, um Namespaces zu deklarieren.

```
<msxsl:script>
   <msxsl:using namespace="myAssemblyNS.NamespaceName" />

   ...

</msxsl:script>
```

Der Wert des `namespace` Attributs ist der Name des Namespace.

© 2019-2025 Altova GmbH

12.3 Technische Daten

Dieses Kapitel enthält Informationen zu einigen technischen Aspekten Ihrer Software. Es ist in die folgenden Abschnitte gegliedert:

- OS- und Arbeitsspeicheranforderungen 633
- Altova-Prozessoren 633
- Unicode-Unterstützung 634
- Internet-Verwendung 634

12.3.1 OS- und Arbeitsspeicheranforderungen

Betriebssystem

Die Altova-Software-Applikationen stehen für die folgenden Plattformen zur Verfügung:

- Windows 10, Windows 11
- Windows Server 2016 oder höher

Arbeitsspeicher

Da die Software in C++ geschrieben wurde, wird dafür nicht so viel Platz wie in einer Java Runtime Umgebung benötigt und normalerweise wird dafür weniger Arbeitsspeicher als bei einer vergleichbaren Java-basierten Applikation benötigt. Es ist notwendig, dass die einzelnen Dokumente in den Hauptarbeitsspeicher geladen werden, damit jedes Dokument zur Gänze geparst und analysiert werden kann und die Anzeige- und Bearbeitungsgeschwindigkeit während der normalen Arbeit verbessert wird. Daher steigt mit der Größe des Dokuments auch der Arbeitsspeicherbedarf.

Auch die unbegrenzte Rückgängig-Funktion kann einiges an Arbeitsspeicher in Anspruch nehmen. Wenn Sie große Abschnitte in großen Dokumenten immer wieder ausschneiden und einfügen, kann dies enorm viel Speicherplatz verbrauchen.

12.3.2 Altova-Prozessoren

XML Validator

Wenn Sie ein XML-Dokument öffnen, verwendet die Applikation den integrierten XML Validator, um das Dokument auf Wohlgeformtheit zu prüfen, es anhand eines Schemas zu validieren (falls eines angegeben wurde) und Baumstrukturen und Infosets zu erstellen. Der Altova XML Validator dient auch dazu, beim Editieren von Dokumenten intelligente Eingabehilfen zur Verfügung zu stellen und etwaige Validierungsfehler dynamisch anzuzeigen.

Im integrierten XML Validator ist die Final Recommendation der W3C XML Schema Spezifikationen 1.0 und 1.1 implementiert. Neue Entwicklungen, die von der XML Schema-Arbeitsgruppe empfohlen werden, werden ständig in den XML Validator integriert, sodass Ihnen mit Altova-Produkten immer eine Entwicklungsumgebung auf dem neuesten Stand der Technik zur Verfügung steht.

XSLT- und XQuery-Prozessor

Die Altova Produkte arbeiten mit dem Altova XSLT 1.0, 2.0 und 3.0-Prozessor und dem Altova XQuery 1.0 und 3.1-Prozessor. Falls einer dieser Prozessoren im Produkt enthalten ist, finden Sie die Dokumentation zurm implementierungsspezifischen Verhalten der einzelnen Prozessoren in den Anhängen zu dieser Dokumentation.

Anmerkung: Altova MapForce verwendet den XSLT 1.0-, 2.0- und XQuery 1.0-Prozessor zur Codegenerierung.

12.3.3 Unicode-Unterstützung

Die XML-Produkte von Altova bieten vollständige Unicode-Unterstützung. Um ein XML-Dokument benötigen Sie eine Schriftart, die die von diesem Dokument verwendeten Unicode-Zeichen unterstützt.

Beachten Sie bitte, dass die meisten Schriftarten nur eine bestimmte Untergruppe des gesamten Unicode-Bereichs enthalten und normalerweise für das entsprechende Schriftsystem ausgelegt sind. Wenn Zeichen falsch dargestellt werden, könnte der Grund darin liegen, dass die gewählte Schriftart die erforderlichen Glyphen nicht enthält. Es ist daher nützlich, eine Schriftart zu verwenden, die den gesamten Unicode-Bereich abdeckt - v.a. wenn Sie XML-Dokumente in unterschiedlichen Sprachen oder Schriftsystemen editieren. Ein typische auf Windows PCs installierte Unicode-Schriftart ist Arial Unicode MS.

Im Ordner `/Examples` Ihres Applikationsordners finden Sie eine XHTML-Datei mit dem Namen `UnicodeUTF-8.html`, die den folgenden Satz in verschiedenen Sprachen und Schriftsystemen enthält:

- *When the world wants to talk, it speaks Unicode*
- *Wenn die Welt miteinander spricht, spricht sie Unicode*
- 世界的に話すなら、Unicode です。)

Wenn Sie diese XHTML-Datei öffnen, erhalten Sie einen kurzen Eindruck davon, was mit Unicode möglich ist und welche Schriftsysteme von den auf Ihrem PC verfügbaren Schriftarten unterstützt werden.

12.3.4 Internet-Verwendung

Altova-Applikationen können für Sie auch eine Verbindung mit dem Internet herstellen. Dies geschieht in den folgenden Fällen:

- Wenn Sie im Registrierungsdialogfeld (**Hilfe | Software-Aktivierung**) auf "Kostenlosen Evaluierungs-Key anfordern" klicken, werden die drei Felder im Registrierungsdialogfeld über eine normale HTTP-Verbindung (Port 80) an unseren Webserver übertragen. Sie erhalten dann per E-Mail (normales SMTP) den kostenlosen Evaluierungs-Keycode zugesandt.
- In einige Altova-Produkten können Sie eine Datei über das Internet öffnen (**Datei | Öffnen | Zu URL wechseln**). In diesem Fall wird das Dokument mittels einer der folgenden Protokollmethoden und Verbindungen aufgerufen: HTTP (normalerweise Port 80), FTP (nomalerweise Port 20/21), HTTPS (normalerweise Port 443). Sie können auch einen HTTP-Server auf Port 8080 verwendden. (Definieren Sie den Port im Dialogfeld "URL", indem Sie ihn durch ein Komma getrennt nach dem Servernamen angeben.*)*

 © 2019-2025 Altova GmbH

- Wenn Sie ein XML-Dokument öffnen, das sich auf ein XML-Schema oder eine DTD bezieht und das Dokument durch eine URL definiert wird, wird das referenzierte Schema-Dokument ebenfalls über eine HTTP-Verbindung (Port 80) oder ein anderes in der URL definiertes Protokoll (siehe Punkt 2 oben) aufgerufen. Ebenso wird ein Schema-Dokument aufgerufen, wenn eine XML-Datei validiert wird. Beachten Sie, dass die Validierung automatisch beim Öffnen des Dokuments erfolgen kann, wenn Sie dies in der Applikation so konfiguriert haben (Register "Datei" des Dialogfelds "Optionen" **Extras | Optionen**).
- Webservice-Verbindungen werden in Altova Applikationen, die WSDL und SOAP verwenden, mittels WSDL-Dokumenten definiert.
- Wenn Sie den Befehl "**Als Mail senden**..." (**Datei | Als Mail senden**) in XMLSpy verwenden, wird die aktuelle Auswahl bzw. Datei über ein MAPI-kompatibles Mail-Programm, das auf dem PC des Benutzers installiert ist, versendet
- Im Rahmen der Altova-Software-Lizenzvereinbarung beschriebenen Software-Aktivierung und beim Live-Update.

12.4 Lizenzinformationen

Dieser Anhang enthält die folgenden Informationen:

- Informationen über den Vertrieb dieses Software-Produkts
- Informationen zur Software-Aktivierung und Lizenzüberwachung
- die Lizenzvereinbarung zu diesem Software-Produkt

Lesen Sie die Informationen bitte sorgfältig - sie sind rechtlich bindend, da Sie sich bei der Installation dieses Software-Produkts damit einverstanden erklärt haben.

Den Inhalt aller Altova-Lizenzenvereinbarungen finden Sie auf der Altova Website unter Rechtliches.

12.4.1 Electronic Software Distribution

Dieses Produkt ist über EDS (Electronic Software Distribution), also auf elektronischem Weg erhältlich, eine Methode, die die folgenden einzigartigen Vorteile bietet:

- Sie können die Software kostenlos 30 Tage lang testen, bevor Sie sich zu einem Kauf entscheiden. (*Anmerkung: Die Lizenz für MobileTogether Designer ist kostenlos*.)
- Wenn Sie sich entschieden haben, die Software zu kaufen, können Sie Ihre Bestellung online auf der Altova Website tätigen. Sie erhalten dann innerhalb weniger Minuten ein vollständig lizenziertes Produkt.
- Sie erhalten immer die neueste Version unserer Software
- Die Software enthält ein umfassendes Hilfesystem, das Sie von der Benutzeroberfläche der Appikation aus aufrufen können. Die neueste Version des Benutzerhandbuchs steht auf unserer Website www.altova.com (i) im HTML-Format zum Aufrufen online und (ii) im PDF-Format zum Download und Ausdrucken zur Verfügung.

30-Tage-Evaluierungszeitraum

Nachdem Sie dieses Software-Produkt heruntergeladen haben, können Sie es 30 Tage lang kostenlos testen. Während dieses Zeitraums werden Sie nach etwa 20 Tagen in regelmäßigen Abständen daran erinnert, dass die Software noch nicht lizenziert wurde. Diese Erinnerungsmeldung wird allerdings nur einmal, nämlich bei jedem Start des Programms, angezeigt. Wenn Sie das Programm nach Ablauf des 30-tägigen Evaluierungszeitraums weiterhin verwenden möchten, müssen Sie eine Produktlizenz erwerben, die Sie in Form einer Lizenzdatei mit einem Keycode erhalten. Laden Sie die Lizenzdatei über das Dialogfeld "Software-Aktivierung" Ihres Produkts hoch, um das Produkt freizuschalten.

Sie können Ihre Produktlizenz über https://shop.altova.com/ erwerben.

© 2019-2025 Altova GmbH

Weitergabe der Software an andere Mitarbeiter in Ihrem Unternehmen zu Testzwecken

Wenn Sie die Evaluierungsversion der Software auch anderen Personen in Ihrem Unternehmen über das Netzwerk zur Verfügung stellen möchten oder wenn Sie sie auf einem PC installieren möchten, der nicht mit dem Internet verbunden ist, dürfen Sie nur das Installationsprogramm weitergeben, vorausgesetzt es wurde nicht modifiziert. Jeder, der das von Ihnen zur Verfügung gestellte Installationsprogramm aufruft, muss einen eigenen Evaluierungs-Keycode für 30 Tage anfordern. Nach Ablauf des Testzeitraums, muss eine Lizenz erworben werden, damit das Produkt weiter verwendet werden kann.

12.4.2 Software-Aktivierung und Lizenzüberwachung

Im Rahmen der Aktivierung der Software durch Altova, verwendet die Software unter Umständen Ihr internes Netzwerk und Ihre Internetverbindung, um die Lizenzdaten während der Installation, Registrierung, der Verwendung oder der Aktualisierung an einen von Altova betriebenen Lizenzserver zu übertragen und die Authentizität der Lizenzdaten zu überprüfen, damit Altova-Software nicht ohne Lizenz oder auf unzulässige Art und Weise verwendet werden kann und um den Kundenservice gleichzeitig zu verbessern. Bei der Aktivierung werden zwischen Ihrem Computer und dem Altova-Lizenzserver für die Lizenzierung erforderliche Daten wie Informationen über Betriebssystem, IP-Adresse, Datum/Uhrzeit, Software-Version und Computername sowie andere Informationen ausgetauscht.

Ihr Altova-Produkt verfügt über ein integriertes Lizenzüberwachungsmodul, das ebenfalls dazu beiträgt, unbeabsichtigte Verletzungen der Lizenzvereinbarung zu vermeiden. Ihr Produkt kann entweder mit einer Einzelplatzlizenz oder einer Mehrfachlizenz erworben werden. Je nach Lizenz stellt das Lizenzüberwachungsmodul sicher, dass nicht mehr als die lizenzierte Anzahl an Benutzern die Applikation gleichzeitig verwendet.

Bei dieser Lizenzüberwachungsmethode wird Ihr LAN-Netzwerk verwendet, um die Kommunikation zwischen Instanzen der Applikation, die auf verschiedenen Computern laufen, zu überwachen.

Einzelplatzlizenz

Beim Start der Applikation wird im Rahmen der Lizenzüberprüfung ein kurzes Broadcast-Datagramm abgesendet, um andere Instanzen des Produkts, die auf anderen Computern im selben Netzwerk laufen, zu finden. Wenn keine Antwort einlangt, wird ein Port geöffnet, der Informationen von anderen Instanzen der Applikation empfangen kann.

Mehrplatzlizenz

Wenn Sie im selben LAN mehrere Instanzen der Applikation verwenden, kommunizieren diese beim Start kurz miteinander, um Keycode-Informationen auszutauschen, damit Sie sicher sein können, dass nicht mehr als die lizenzierte Anzahl an Lizenzen gleichzeitig in Verwendung ist. Dieselbe Lizenzüberwachungstechnologie wird auch bei Unix und vielen anderen Datenbankentwicklungstools verwendet. Sie gestattet Benutzern den Erwerb von Parallellizenzen für mehrere Benutzer zu vernünftigen Preisen.

Wir sind außerdem bestrebt, nur wenige, kleine Netzwerkpakete zu versenden, um Ihr Netzwerk nicht zu überlasten. Die von Ihrem Altova Produkt verwendeten TCP/IP Ports (2799) sind offiziell bei IANA registriert, (*nähere Informationen siehe IANA Service Name Registry*) und unser Lizenzüberwachungsmodul basiert auf einer bewährten und erprobten Technologie.

Wenn Sie eine Firewall verwenden, werden Sie unter Umständen feststellen, dass die Computer, auf denen Altova-Produkte laufen, über Port 2799 miteinander kommunizieren. Sie können diesen Netzwerkverkehr

zwischen verschiedenen Gruppen in Ihrem Unternehmen natürlich blockieren, solange Sie mit anderen Mitteln sicherstellen können, dass Ihre Lizenzvereinbarung eingehalten wird.

Anmerkung zu Zertifikaten

Ihre Altova Applikation kontaktiert den Altova Lizenzierungsserver über HTTPS (`link.altova.com`). Für diese Kommunikation verwendet Altova ein registriertes SSL-Zertifikat. Wenn dieses Zertifikat ersetzt wird (z.B. von Ihrer IT-Abteilung oder einer externen Agentur), werden Sie von Ihrer Altova Applikation gewarnt, dass die Verbindung nicht sicher ist. Sie könnten Ihre Altova Applikation mit dem Ersetzungszertifikat starten. Dies würde jedoch auf Ihr eigenes Risiko geschehen. Wenn Sie eine Warnung sehen, dass die *Verbindung nicht sicher* ist, überprüfen Sie den Ursprung des Zertifikats und wenden Sie sich an Ihr IT-Team (die in der Lage sein sollten, zu entscheiden, ob das Abfangen und die Ersetzung des Altova-Zertifkats fortgesetzt werden soll).

Wenn Ihr Unternehmen sein eigenes Zertifikat verwenden muss (z.B. um die Kommunikation zu und von Client-Rechnern zu überwachen), empfehlen wir Ihnen, Altova LicenseServer, die kostenlose Lizenzverwaltungssoftware von Altova in Ihrem Netzwerk zu installieren. Client-Rechner verwenden mit dieser Konfiguration weiterhin die Zertifikate Ihres Unternehmens, während der Altova LicenseServer für die Kommunikation mit Altova das Altova-Zertifikat das Altova-Zertifikat verwenden kann.

12.4.3 Altova Endbenutzer-Lizenzvereinbarung

- Die Altova-Endbenutzer-Lizenzvereinbarung kann unter https://www.altova.com/de/legal/eula eingesehen werden.
- Die Altova-Datenschutzbestimmungen finden Sie unter https://www.altova.com/de/privacy.

12.4.4 Verpacken von Lizenzdateien in den MapForce Installer

Wenn Sie eine stillen Installation von MapForce durchführen möchten, können Sie gegebenenfalls die MSI-Datenbank ändern, so dass Sie Ihre Lizenzdatei(en) enthält. Dadurch installiert der Installer nicht nur das Produkt, sondern lizenziert es auch. Eine genauere Anleitung dazu finden Sie, wenn Sie diese ZIP-Datei von der Altova-Website herunterladen und das darin enthaltene PDF-Dokument öffnen.

© 2019-2025 Altova GmbH

Index

.

.NET Erweiterungsfunktionen,
Datentypkonvertierungen, .NET in XPath/XQuery, 630
Datentypkonvertierungen, XPath/XQuery in .NET, 629
für XSLT und XQuery, 624
Instanzmethoden, Instanzfelder, 628
Konstruktoren, 626
statische Methoden, statische Felder, 627

A

A bis Z,
Sortierkomponente, 188
Abgeleitete Typen,
mappen von/auf, 136
xsi:type, 136
abs,
als MapForce-Funktion (in xpath2 | numeric functions), 370
ActiveX,
Integration auf Applikationsebene, 455
Integration auf Dokumentebene, 457
Voraussetzungen für die Integration, 452
ActiveX Controls,
zur Visual Studio Toolbox hinzufügen, 454
add,
als MapForce-Funktion (in core | math functions), 284
Allgemeine Verfahren, 63
Ausgabe validieren, 63
Ausgabedateieinstellungen, 74
Code generieren, 65
Codegenerierung, 74
Mapping validieren, 63
Mapping-Einstellungen, 74
Pfade im generierten Code, 74
Suchen in der Textansicht, 71
Textansicht, 67
valid, 63
Validierung, 63
Altova XML Parser,
Info, 633
Altova-Erweiterungen,
Diagrammfunktionen (siehe Diagrammfunktionen), 534
Anpassen,
Befehle, 505
Befehle löschen, 505
Kontextmenüs, 505
Kürzel, 506
Menüleisten zurücksetzen, 505
Menüs, 505
Menüschatten, 505
Standardmenü vs. MapForce Design, 505
Tastatur, 506
Ansicht,
Annotationen anzeigen, 502
Ausgewählte Komponentenkonnektoren anzeigen, 502
Bibliothek in Funktionstitelleiste anzeigen, 502
Bibliotheken, 502
Bibliotheken verwalten, 502
Datentypen anzeigen, 502
Debug-Fenster, 502
Meldungen, 502
Menübefehl, 502
Projekt-Fenster, 502
Quell- und Zielkonnektoren anzeigen, 502
Statusleiste, 502
Tipps anzeigen, 502
Übersicht, 502
Vergrößern/Verkleinern, 502
Vorwärts, 502
XBRL-Anzeigeoptionen, 502
Zurück, 502
Arbeitsspeicher-Anforderungen, 633
ATTLIST,
DTD Namespace URIs, 130
Ausgabe,
Alle Ausgabedateien speichern, 500
Alle Lesezeichen löschen, 500
Ausgabedatei neu generieren, 500
Ausgabedatei speichern, 500
Ausgabedatei validieren, 500
Built-In Ausführungsprozessor, 500
C#, 500
C++, 500
Einstellungen für die Textansicht, 500
Java, 500
Lesezeichen einfügen/löschen, 500
Nächstes Lesezeichen, 500

© 2019-2025 Altova GmbH

Ausgabe,
Pretty-Print, 500
SQL/NoSQL-Script ausführen, 500
Vorheriges Lesezeichen, 500
XQuery, 500
XSLT 1.0, 500
XSLT 2.0, 500
XSLT 3.0, 500
auto-number,
als MapForce-Funktion (in core | generator functions), 275
avg,
als MapForce-Funktion (in core | aggregate-Funktionen), 254

B

base-uri,
als MapForce-Funktion (in xpath2 | accessors-Bibliothek), 339
Bearbeiten,
Alle auswählen, 492
Ausschneiden/Kopieren/Einfügen/Löschen, 492
Rückgängig, 492
Suchen, 492
Vorheriges suchen, 492
Weitersuchen, 492
Wiederherstellen, 492
Benutzerdefinierte Funktionen,
aufrufen, 220
bearbeiten, 220
Beispiel, 219
Beispiele, 230, 235
erstellen, 220
importieren, 220
inline, 220
Input-Parameter, 220
kopieren und einfügen, 220
Look-up, 235
löschen, 220
Navigation, 220
Output-Parameter, 220
Parameter, 225
Parameter hinzufügen, 225
Parameterreihenfolge, 225
reguläre, 220
rekursiv, 230
rekursiv aufrufen, 230
rekursive Suche, 230
Strukturen von komplexen Typen (complexTypes), 225
Übersicht, 219
vom Typ complexType, 225
vom Typ simpleType, 225
Vorteile, 219
Benutzeroberfläche, 19
Bereiche, 25
Fenster, 20
Fenster "Meldungen", 24
Leisten, 20
Bereiche,
Ausgabe, 25
DB-Abfrage, 25
Mapping, 25
StyleVision-Ausgabe, 25
XQuery, 25
XSLT, 25
Betriebssysteme für Altova-Produkte, 633
boolean,
als MapForce-Funktion (in core | conversion functions), 261

C

C#,
Integration von MapForce, 461
CDATA,
Abschnitt, 140
ceiling,
als MapForce-Funktion (in core | math functions), 285
char-from-code,
als MapForce-Funktion (in core | string functions), 326
Code Point,
Collation, 188
code-from-char,
als MapForce-Funktion (in core | string functions), 328
Codegenerierung,
Applikation ausführen, 65
bauen, 65
C#, 65
C++, 65
Code erstellen, 65
Code generieren, 65
Java, 65
kompilieren, 65

© 2019-2025 Altova GmbH

Codegenerierung,
XQuery, 65
XSLT, 65
Collation,
Lokale Collation, 188
Sortierkomponente, 188
Unicode Code Point, 188
ComplexType,
sortieren, 188
concat,
als MapForce-Funktion (in core | string functions), 329
contains,
als MapForce-Funktion (in core | string functions), 330
Copyright-Informationen, 636
count,
als MapForce-Funktion (in core | aggregate-Funktionen), 255
current,
als MapForce-Funktion (in xslt | xslt Funktionsbibliothek), 400
current-date,
als MapForce-Funktion (in xpath2 | context functions), 344
current-dateTime,
als MapForce-Funktion (in xpath2 | context functions), 344
current-time,
als MapForce-Funktion (in xpath2 | context functions), 345

D

Datei,
Alle schließen, 489
Alles speichern, 489
als Schaltfläche in einer Komponente, 38
als Schaltfläche in Komponenten, 422
Anmeldeinformationen-Manager öffnen, 489
Auf FlowForce Server bereitstellen, 489
Beenden, 489
Code generieren, 489
Dokumentation generieren, 489
Drucken, 489
Druckereinrichtung, 489
Druckvorschau, 489
Letzte Dateien, 489
Mapping validieren, 489
Mapping-Einstellungen, 489
Neu, 489
Neu laden, 489
Öffnen, 489
Schließen, 489
Speichern, 489
Speichern unter, 489
Zu MapForce Server-Ausführungsdatei kompilieren, 489
Datei/String,
als Schaltfläche in einer Komponente, 38
als Schaltfläche in Komponenten, 422
Datei: (Standard),
als Name des Root-Node, 422
Datei: <dynamisch>,
als Name des Root-Node, 422
Dateien,
mehrere Dateien anhand einer Datenbank, 424
Dateipfade,
absolute, 40
falsche, 40
falsche Referenzen korrigieren, 40
im generierten Code, 43
in Ausführungsumgebungen, 43
relative, 40
relative im Gegensatz zu absoluten, 43
von dateibasierten Datenbanken, 40
Daten sortieren,
Sortierkomponente, 188
Datenelement,
fehlendes, 59
Datenstreaming,
Definition, 35
default-collation,
als MapForce-Funktion (in xpath2 | context functions), 345
distinct-values,
als MapForce-Funktion (in core | sequence functions), 297
divide,
als MapForce-Funktion (in core | math functions), 285
document,
als MapForce-Funktion (in xslt | xslt Funktionsbibliothek), 400
Dokumentebene,
Beispiele für die Integration von MapForce, 461
DoTransform.bat,
mit RaptorXML Server ausführen, 428
DTD,
Quelle und Ziel, 130

Netzwerk-Proxy, 508
XBRL, 508

E

element-available,
als MapForce-Funktion (in xslt | xslt Funktionsbibliothek), 400
Endbenutzer-Lizenzvereinbarung, 636, 638
Enumerationen,
in MapForceControl, 487
equal,
als MapForce-Funktion (in core | logical functions), 278
equal-or-greater,
als MapForce-Funktion (in core | logical functions), 279
equal-or-less,
als MapForce-Funktion (in core | logical functions), 279
Erweiterungsfunktionen für XSLT und XQuery,
Altova-Erweiterungen, 534
Erweiterungsfunktionen in .NET für XSLT und XQuery,
Java-Erweiterungsfunktionen, 624
siehe .NET Erweiterungsfunktionen, 624
Erweiterungsfunktionen in Java für XSLT und XQuery,
Java-Erweiterungsfunktionen, 614
siehe Java-Erweiterungsfunktionen, 614
Erweiterungsfunktionen in MSXSL Scripts, 630
Evaluierungszeitraum,
für Altova-Software-Produkte, 636
von Altova Software-Produkten, 636
exists,
als MapForce-Funktion (in core | sequence functions), 299
Extras,
Aktive Konfiguration, 504
Anpassen, 504
Globale Ressourcen, 504
Menübefehl, 504
Optionen, 504
Symbolleisten und Fenster wiederherstellen, 504
Umgekehrtes Mapping erstellen, 504
XBRL-Taxonomiepakete, 504
Extras | Optionen,
Allgemein, 508
Bearbeiten, 508
Code-Generierung, 508
Datenbank, 508
Debugger, 508
Java, 508
Meldungen, 508

F

false,
als MapForce-Funktion (in xpath2 | boolean functions), 342
Fehlende Datenelemente, 59
Fehler,
Behebung, 35
zu wenig Speicher, 35
Fehlerhafte Verbindungen,
in Datenbanken, 59
in XML-Dateien, 59
nach Ändern des Schemas, 59
Fenster,
Bibliotheken, 20
Bibliotheken verwalten, 20
Design, 517
Dialogfeld "Fenster", 517
Dunkles Design, 517
Helles Design, 517
Horizontal/Vertikal anordnen, 517
Klassisches Design, 517
Mapping, 20
Mehrere Mapping-Fenster, 20
Meldungen, 24
Projekt, 20
Überlappend, 517
Übersicht, 20
Filter,
zum Mapping hinzufügen, 194
Filtern,
Daten aus Komponenten, 194
Datenbanktabellen, 194
first-items,
als MapForce-Funktion (in core | sequence functions), 301
floor,
als MapForce-Funktion (in core | math functions), 286
format-date,
als MapForce-Funktion (in core | conversion functions), 261
format-dateTime,
als MapForce-Funktion (in core | conversion functions), 262
format-number,
als MapForce-Funktion (in core | conversion functions), 266
format-time,

© 2019-2025 Altova GmbH

format-time,
als MapForce-Funktion (in core | conversion functions), 269
Fragezeichen,
fehlende Datenelemente, 59
function-available,
als MapForce-Funktion (in xslt | xslt Funktionsbibliothek), 401
Funktionen, 212
Argumentdatentyp, 212
Benutzerdefinierte Funktion erstellen, 499
Benutzerdefinierte Funktion von Auswahl erstellen, 499
Beschreibung, 212
durchsuchen, 212
Funktion entfernen, 499
Funktionseinstellungen, 499
Grundlagen, 212
hinzufügen, 212
im Fenster "Bibliotheken" suchen, 212
Input-Komponente einfügen, 499
Instanzen im aktiven Mapping suchen, 212
Konstanten, 212
Output-Komponente einfügen, 499
Parameter, 212
Parameter hinzufügen, 212
Parameter löschen, 212

G

generate-id,
als MapForce-Funktion (in xslt | xslt Funktionsbibliothek), 401
generate-sequence,
als MapForce-Funktion (in core | sequence functions), 302
get-fileext,
als MapForce-Funktion (in core | file path functions), 271
get-folder,
als MapForce-Funktion (in core | file path functions), 272
Global, 634
Globale Ressourcen,
Definitionsdatei, 433
Einführung, 432
einrichten, 433
erstellen, 433
Ordner, 440
XML-Dateien als, 438
Grafische Benutzeroberfläche, 19
greater,
als MapForce-Funktion (in core | logical functions), 280
group-adjacent,
als MapForce-Funktion (in core | sequence functions), 303
group-by,
als MapForce-Funktion (in core | sequence functions), 305
group-ending-with,
als MapForce-Funktion (in core | sequence functions), 309
group-into-blocks,
als MapForce-Funktion (in core | sequence functions), 311
group-starting-with,
als MapForce-Funktion (in core | sequence functions), 313
Grundlagen, 29, 31, 45, 63
Kontext, 79
Mapping-Regeln, 77
Mapping-Strategien, 77
Mehrere Zielkomponenten, 92
Prioritätskontext, 87
Sequenzen, 78
Verarbeitungsreihenfolge, 79

H

Hilfe,
Auf Updates überprüfen, 519
Bestellformular, 519
Fragen und Antworten im Web, 519
Index, 519
Inhaltsverzeichnis, 519
Komponenten und Gratistools downloaden, 519
MapForce im Internet, 519
MapForce Training, 519
Registrierung, 519
Software-Aktivierung, 519
Suchen, 519
Support Center, 519
Über MapForce, 519
Hintergrundinformationen, 633

I

If-Else-Bedingungen,
zum Mapping hinzufügen, 194
implicit-timezone,

implicit-timezone,
als MapForce-Funktion (in xpath2 | context functions), 345
Input,
duplizieren, 38
Integrating,
MapForce in Applikationen, 451
Integration,
mit Altova-Produkten, 18
Intelligente Komponentenlöschung, 61
Internet-Verwendung,
in Altova-Produkten, 634
ISO/IEC 10646, 634
is-xsi-nil,
als MapForce-Funktion (in core | node functions), 290
item-at,
als MapForce-Funktion (in core | sequence functions), 315
items-from-till,
als MapForce-Funktion (in core | sequence functions), 316

J

Java, 462
VM-Bibliothekspfad, 512
Java Erweiterungsfunktionen,
Datentypkonvertierungen, Java in Xpath/XQuery, 623
Instanzmethoden, Instanzfelder, 621
Konstruktoren, 620
statische Methoden, statische Felder, 621
Java-Erweiterungsfunktionen,
benutzerdefinierte JAR-Dateien, 619
benutzerdefinierte Klassendateien, 616
Datentyp-Konvertierungen, XPath/XQuery in Java, 622
für XSLT und XQuery, 614

K

Kataloge, 442
anpassen, 447
in DTD, 443
in XML-Schema, 443
Struktur, 445
Umgebungsvariablen, 449
Kommentare,
zu Zieldateien hinzufügen, 140
Komponente,
Daten sortieren, 188
Komponenten,
Aktualisieren, 496
Ausnahme, 493
ausrichten, 38
Datenbank, 493
Datenbank abfragen, 496
Datenbankaktionen, 496
Datenbankobjekte hinzufügen/entfernen/bearbeiten, 496
Duplikat danach einfügen, 496
Duplikat davor einfügen, 496
Duplikat löschen, 496
durchsuchen, 38
EDI, 493
Eigenschaften, 496
Einfache Input-Komponente, 493
Einfache Output-Komponente, 493
Einstellungen, 38
Einstellungen ändern, 38
Excel 2007+-Datei, 493
Filter: Nodes/Zeilen, 493
FlexText-Konfiguration bearbeiten, 496
gelöschte Datenelemente, 59
Grundlagen, 38
IF-Else-Bedingung, 493
Inhalt als CDATA-Abschnitt schreiben, 496
Input-Komponente einfügen, 493
Join, 493
JSON-Schema/Datei, 493
Kommentar, 31
Konstante, 493
Links ausrichten, 496
löschen, 61
Mapping auf EDI X12 997 erstellen, 496
Mapping auf EDI X12 999 erstellen, 496
Menübefehle, 496
Output-Komponente einfügen, 493
Protocol Buffers-Datei, 493
Rechts ausrichten, 496
Root-Element ändern, 496
Schema-Definition in XMLSpy bearbeiten, 496
Sortieren: Nodes/Zeilen, 493
SQL/NoSQL-WHERE/ORDER, 493
Strukturkomponenten, 31, 129, 130
Symbolreferenz, 31
Textdatei, 493
Transformation, 31, 163

© 2019-2025 Altova GmbH

Komponenten,
Übersicht, 31
Variable, 493
Webservice-Funktion, 493
Wertezuordnung, 493
XBRL-Dokument, 493
XML, 131
XML und XML-Schema, 131, 136, 138, 140, 142, 144
XML-Schema, 131
XML-Schema/Datei, 493
zum Mapping hinzufügen, 35
Konstanten,
hinzufügen, 212
Konventionen, 14

L

last,
als MapForce-Funktion (in xpath2 | context functions), 346
last-items,
als MapForce-Funktion (in core | sequence functions), 317
Leisten,
Applikationsstatus, 20
Menü, 20
Symbolleisten, 20
less,
als MapForce-Funktion (in core | logical functions), 280
Lizenz, 638
Informationen, 636
Lizenzierung, 519
Lizenzdateien in den Installer verpacken, 638
Lizenzüberwachung,
in Altova-Produkten, 637
Locale Collation, 188
local-name-from-QName,
als MapForce-Funktion (in lang | QName functions), 295
logical-and,
als MapForce-Funktion (in core | logical functions), 281
logical-not,
als MapForce-Funktion (in core | logical functions), 281
logical-or,
als MapForce-Funktion (in core | logical functions), 282
Lookup-Tabellen,
im Mapping verwenden, 200
Löschen,
fehlende Datenelemente, 59

M

main-mfd-filepath,
als MapForce-Funktion (in core | file path functions), 272
MapForce,
Datentransformationsmodell, 14
Einführung, 14
Integration, 451
Übersicht, 14
MapForceCommand,
in MapForceControl, 468
MapForceCommands,
in MapForceControl, 470
MapForceControl, 471
Beispiele für die Integration auf Dokumentebene, 461
Dokumentation, 451
Integration mittels C#, 461
Objektreferenz, 468
MapForceControlDocument, 478
MapForceControlPlaceHolder, 484
Mapping,
Ausgabe speichern, 104
Ausgabedateieinstellungen, 74
Begriffe, 29
Bestandteile, 29
Code generieren, 104
dynamische Dateinamen, 126
Einstellungen, 74
erstellen, 29, 98
Funktion hinzufügen, 102
Grundlagen, 29
Komponente hinzufügen, 99
Komponenten, 29
Komponententypen, 29
Konnektoren, 29
Quelle, 15
quellorientiert - Mixed Content, 49
speichern, 98
Struktur anzeigen, 99
Szenarien, 16
Terminologie, 29
Transformationssprache auswählen, 98
validieren, 98
Verbindungen, 29
XML-Schema-Version, 74

© 2019-2025 Altova GmbH

Mapping,
Ziel, 15
Mapping-Input,
Mehrere Dateien bereitstellen als, 422
Mapping-Output,
Mehrere Dateien generieren als, 422
max,
als MapForce-Funktion (in core | aggregate-Funktionen), 256
max-string,
als MapForce-Funktion (in core | aggregate-Funktionen), 256
Mehrere XML-Dateien,
anhand einer einzigen XML-Quelle, 424
Menübefehle, 488
Anpassen, 505, 506
Ansicht, 502
Ausgabe, 500
Bearbeiten, 492
Datei, 489
Einfügen, 493
Extras, 504
Extras | Anpassen, 505
Extras | Optionen, 508
Extras | Optionen | Java, 512
Extras | Optionen | Netzwerk-Proxy, 514
Extras | Tastatur, 506
Fenster, 517
Funktion, 499
Hilfe, 519
Komponente, 496
Verbindung, 498
Menüreferenz, 488
mfd-filepath,
als MapForce-Funktion (in core | file path functions), 272
Microsoft SharePoint Server,
Dateien als Komponenten hinzufügen über, 35
min,
als MapForce-Funktion (in core | aggregate-Funktionen), 257
min-string,
als MapForce-Funktion (in core | aggregate-Funktionen), 258
Mixed,
Content mappen, 49
quellorientiertes Mapping, 49
Mixed Content,
mappen, 49
mit Standardverbindungen, 49
mit zielorientierten Verbindungen, 49
modulus,
als MapForce-Funktion (in core | math functions), 286
msxsl:Script, 630
multiply,
als MapForce-Funktion (in core | math functions), 287

N

Namespace URI,
DTD, 130
Namespaces,
benutzerdefinierte, 144
manuell deklarieren, 144
namespace-uri-form-QName,
als MapForce-Funktion (in lang | QName functions), 296
Netzwerk-Proxy,
automatisch, 514
Einstellungen, 514
Konfiguration, 514
manuell, 514
System, 514
Neue Funktionen, 10
Version 2021, 13
Version 2022, 12
Version 2023, 11
Version 2024, 10
node-name,
als MapForce-Funktion (in core | node functions), 292
als MapForce-Funktion (in xpath2 | accessors-Bibliothek), 340
node-name (Funktion),
Alternativen, 405
Node-Namen,
Daten mappen von/auf, 405
normalize-space,
als MapForce-Funktion (in core | string functions), 331
not-equal,
als MapForce-Funktion (in core | logical functions), 283
not-exists,
als MapForce-Funktion (in core | sequence functions), 318
NULL,
Attribut, 138
Werte, 138
Werte in Datenbanken, 138
number,

© 2019-2025 Altova GmbH

number,
als MapForce-Funktion (in core | conversion functions), 270

O

Ordner,
als globale Ressourcen, 440

P

Parameter,
für das Mapping bereitstellen, 164, 168
Parent-Kontext,
Beispiel, 83
Parser,
in Altova-Produkte integrierter, 633
Plattformen für Altova-Produkte, 633
position,
als MapForce-Funktion (in core | sequence functions), 318
Prioritätskontext,
Beispiel, 89
Processing Instructions,
zu Zieldateien hinzufügen, 140
Processing Instructions und Kommentare,
mappen, 49

Q

QName,
als MapForce-Funktion (in lang | QName functions), 295
Quelldatei,
in mehrere Zieldateien aufteilen, 424
Quellorientiertes Mapping,
Mixed Content Mapping, 49

R

RaptorXML Server,
Transformation ausführen, 428
Rechtliches, 636
Regular Expressions,
in Mappings verwenden, 247
remove-fileext,
als MapForce-Funktion (in core | file path functions), 273
remove-folder,
als MapForce-Funktion (in core | file path functions), 273
replace-fileext,
als MapForce-Funktion (in core | file path functions), 274
replicate-item,
als MapForce-Funktion (in core | sequence functions), 321
replicate-sequence,
als MapForce-Funktion (in core | sequence functions), 323
resolve-filepath,
als MapForce-Funktion (in core | file path functions), 274
resolve-uri,
als MapForce-Funktion (in xpath2 | anyURI functions), 341
round,
als MapForce-Funktion (in core | math functions), 287
round-half-to-even,
als MapForce-Funktion (in xpath2 | numeric functions), 370
round-precision,
als MapForce-Funktion (in core | math functions), 288

S

Schema,
Branchenstandard, 130
generieren, 130
vorverpackt, 130
Schema-Manager,
CLI-Befehl Help, 157
CLI-Befehl Info, 157
CLI-Befehl Initialize, 158
CLI-Befehl Install, 158
CLI-Befehl List, 159
CLI-Befehl Reset, 160
CLI-Befehl Uninstall, 160
CLI-Befehl Update, 161
CLI-Befehl Upgrade, 162
CLI-Übersicht, 156
Patch für Schema installieren, 154
Schema deinstallieren, 155
Schema installieren, 154
Schemas nach Status auflisten in, 152
starten, 150
Status von Schemas, 152
Übersicht, 147

© 2019-2025 Altova GmbH

Schema-Manager,
Upgrade für Schema installieren, 154
zurücksetzen, 155
Schlüssel,
Sortierschlüssel, 188
Schlüssel-Wert-Paare,
im Mapping verwenden, 200
set-empty,
als MapForce-Funktion (in core | sequence functions), 324
set-xsi-nil,
als MapForce-Funktion (in core | node functions), 293
SimpleType,
sortieren, 188
skip-first-items,
als MapForce-Funktion (in core | sequence functions), 324
Software-Produktlizenz, 638
Sortieren,
Sortierkomponente, 188
Sortierreihenfolge,
ändern, 188
Sortierschlüssel,
Sortierkomponente, 188
SQLite,
Datenbankpfad im generierten Code in einen absoluten ändern, 43
starts-with,
als MapForce-Funktion (in core | string functions), 332
static-node-annotation,
als MapForce-Funktion (in core | node functions), 293
static-node-name,
als MapForce-Funktion (in core | node functions), 294
Stille Installation,
Lizenzdateien in den Installer verpacken, 638
MSI-Datei ändern, 638
string,
als MapForce-Funktion (in core | conversion functions), 271
als MapForce-Funktion (in xpath2 | accessors-Bibliothek), 340
string-join,
als MapForce-Funktion (in core | aggregate-Funktionen), 259
string-length,
als MapForce-Funktion (in core | string functions), 332
Strukturkomponenten,
XML, 130
XML und XML-Schema, 130
XML-Schema, 130
substitute-missing,
als MapForce-Funktion (in core | sequence functions), 325
substitute-missing-with-xsi-nil,
als MapForce-Funktion (in core | node functions), 294
substring,
als MapForce-Funktion (in core | string functions), 333
substring-after,
als MapForce-Funktion (in core | string functions), 333
substring-before,
als MapForce-Funktion (in core | string functions), 334
subtract,
als MapForce-Funktion (in core | math functions), 288
Suchen,
Datenelemente in Mapping-Komponenten, 38
sum,
als MapForce-Funktion (in core | aggregate-Funktionen), 260
system-property,
als MapForce-Funktion (in xslt | xslt Funktionsbibliothek), 402

T

Tabellendaten,
sortieren, 188
Technische Informationen, 633
Textansicht,
durchsuchen, 71
Einrücklinien, 67
Klappleiste, 67
Klappleisten, 67
Lesezeichen, 67
Pretty-Print-Anzeige, 67
Syntaxfärbung, 67
Textmarkierung, 67
Vergrößern/Verkleinern, 67
Whitespace-Markierungen, 67
Zeilenendemarkierungen, 67
Zeilennummerierung, 67
Zeilenumbruch, 67
tokenize,
als MapForce-Funktion (in core | string functions), 335
tokenize-by-length,
als MapForce-Funktion (in core | string functions), 335
tokenize-regexp,
als MapForce-Funktion (in core | string functions), 336
Transformationen,

© 2019-2025 Altova GmbH

Transformationen,
RaptorXML Server, 428
Transformationssprachen,
BUILT-IN, 16
C#, 16
C++, 16
Java, 16
XQuery, 16
XSLT 1.0, 16
XSLT 2.0, 16
XSLT 3.0, 16
translate (in core | string functions),
als MapForce-Funktion, 338
true,
als MapForce-Funktion (in xpath2 | boolean functions), 342
Tutorials,
Beispieldateien, 96
dynamische Dateinamen, 120
eine Quellkomponente auf eine Zielkomponente, 97
Grundlagen, 97
Input duplizieren, 106
mehrere Quellkomponenten auf eine Zielkomponente, 106
mehrere Quellkomponenten auf mehrere Zielkomponenten, 120
Transformation, 97
verkettetes Mapping, 111
Weiterleitungskomponente, 111

U

UCS-2, 634
UCS-4, 634
UDFs,
and mapping context, 232
Unicode,
Code Point Collation, 188
Konsortium, 634
Standard, 634
unparsed-entity-uri,
als MapForce-Funktion (in xslt | xslt Funktionsbibliothek), 402
URI,
in DTDs, 130
URL,
Dateien als Komponenten hinzufügen von, 35
UTF-8, 634

V

Validator,
in Altova-Produkten, 633
Variablen,
Beispiel für die Verwendung, 185
Beispiele für die Verwendung, 184
DB-basiert, 175
einfache, 175
Geltungsbereich ändern, 181
komplexe, 175
zum Mapping hinzufügen, 177
Verbindungen,
alles kopieren, 49, 54
Alles kopieren (Sub-Einträge kopieren), 498
ändern, 45
Annotation, 56
Arten, 49
beibehalten nach Löschen von Komponenten, 61
Eigenschaften, 498
Einstellungen, 56
Einstellungen für 'Idente Sub-Einträge verbinden', 498
erstellen, 45
fehlende übergeordnete Verbindungen, 45
Fehler nach Bearbeitung des Schemas beheben, 59
Idente Sub-Einträge automatisch verbinden, 498
Idente Sub-Einträge verbinden, 498
Kontextmenü, 57
kopieren, 45
löschen, 45
mixed, 49
obligatorische Inputs, 45
quellorientiert, 49
Quellorientiert (Mixed Content), 498
reparieren, 59
selektiv markieren, 45
Standard, 49
Sub-Einträge kopieren, 49, 52
Tooltips zu Verbindungen anzeigen, 45
Typen, 56
verschieben, 45, 59
zielorientiert, 49
Zielorientiert (Standard), 498
Verbindungsart,
alles kopieren, 54

© 2019-2025 Altova GmbH

Verbindungsart,
mixed, 49
quellorientiert, 49
Standard, 49
Standard mit Mixed Content, 49
Sub-Einträge kopieren, 52
zielorientiert, 49
zielorientiert mit Mixed Content, 49
zielorientierte im Vergleich zu quellorientierten Verbindungen, 49
Vertrieb,
von Altova Software-Produkten, 636
von Altova-Software-Produkten, 636
Visual Studio,
die MapForce ActiveX Controls zur Toolbox hinzufügen, 454

W

WebDAV Server,
Dateien als Komponenten hinzufügen von, 35
Wertezuordnung,
als Mapping-Komponente, 200
Beispiele, 205, 208
Wildcards,
Auswahl, 142
Schema importieren, 142
Wrapper-Schema, 142
xs:any/xs:anyAtrribute, 142
Windows,
Unterstützung für Altova-Produkte, 633

X

XML, 634
BOM, 131
Bytefolge, 131
Dateipfade relativ zur MFD-Datei speichern, 131
Deklaration, 131
digitale Signatur, 131
DTD-Referenz hinzufügen, 131
Kodierungseinstellungen, 131
Komponenteneinstellungen, 131
Komponentenname, 131
min/maxOccurs, 131
Pretty Print, 131
Schema hinzufügen, 131
Schema-Datei, 131
standalone, 131
standalone="yes", 131
StyleVision Power Stylesheet-Datei, 131
Werte in Zieltypen konvertieren, 131
XML-Deklaration, 131
XML-Input-Datei, 131
XML-Output-Datei, 131
XML Parser,
Info, 633
XML-Dateien,
als globale Ressourcen, 438
anhand einer einzigen XML-Quelle, 424
XML-Schema-Manager, 130
xs: any, 142
xs:anyAttribute, 142
xsi:nil,
als Attribut in einer XML-Instanz, 138
XSLT,
benutzerdefinierte Funktionen entfernen, 240
benutzerdefinierte Funktionen hinzufügen, 240
Template Namespace, 240

Z

Z bis A,
Sortierkomponente, 188
Zieldatei,
mehrere Zieldateien anhand einer einzigen Quelldatei, 424