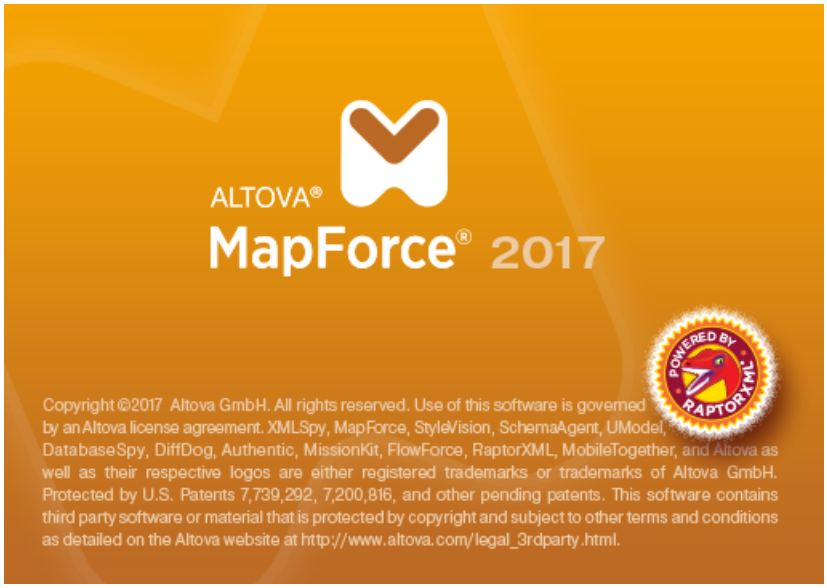
ユーザーマニュアル


ALTOVA®
MapForce® 2017
POWERED BY RAPTORXML
Copyright ©2017 Altova GmbH. All rights reserved. Use of this software is governed by an Altova license agreement. XMLSpy, MapForce, StyleVision, SchemaAgent, UModel, DatabaseSpy, DiffDog, Authentic, MissionKit, FlowForce, RaptorXML, MobileTogether, and Altova as well as their respective logos are either registered trademarks or trademarks of Altova GmbH. Protected by U.S. Patents 7,739,292, 7,200,816, and other pending patents. This software contains third party software or material that is protected by copyright and subject to other terms and conditions as detailed on the Altova website at http://www.altova.com/legal_3rdparty.html.

Altova MapForce 2017 Basic Edition
ユーザーマニュアル

発行日 : 2017

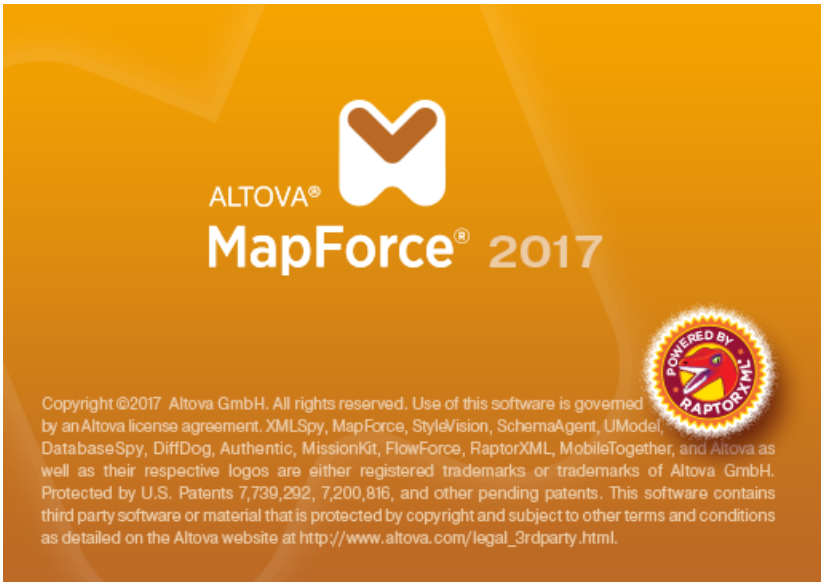
目次

Chapter 1

Altova MapForce 2017 Basic Edition

1 Altova MapForce 2017 Basic Edition

MapForce® 2017 Basic Edition は高度なデータ統合プロジェクトにて決定的な役割を果たす視覚的なデータマッピングツールです MapForce(R) Windows XP/Vista, Windows 7/8/10 とWindows Server 2003/2008/2012/2016 で作動します。



最終更新日 : 201?年 04月 28日

1.1 新規機能

MapForce 2017 リリース 3の新規機能:

- 内でのテキストの検索オプション。**出力** ペインと『**SLT**』ペインが強化されました (次を参照 : テキストビュー内の検索)。また、テキストのハイライトも上記のペインで使用することができます (次を参照 : テキストのハイライト)。
- 内部のアップデートと最適化

MapForce 2017 の新規機能:

- ソースXML からノード名を読み取るを読み取ることは可能でターゲットにこの情報をマップすることができます。また、動的に新規のXML 属性、または、要素をソースから与えられた値をベースにしたターゲット内に作成することもできます。ノード名のマッピング を参照してください。
- 要素レベルでカスタム化された名前空間を持つXML インスタンスファイルを作成することができます (次を参照してください : カスタム名前空間の宣言)
- 内部のアップデートと最適化

MapForce 2016 R2 の新規機能:

- XSLT ペイン内の更に直感的なコードの使用 : 折りたたまれたテキストが、省略シンボルと共に表示され、ツールヒントとして、プレビューすることができます。
- アクティブなマッピング内で関数が使用される全ての箇所を検索することができます (ライブラリウィンドウ 内の関数を右クリックし、**全ての呼び出しを検索する**』を選択します)。
- 内部のアップデートと最適化

MapForce 2016 の新規機能 :

- XSLT 1.0 コードの改善された生成 (生成されたスタイルシートは読み込みやすく、より早く実行することができます)。
- 新規 集計関数が MapForce コアライブラリで使用することができます **`min-string`** と **`max-string`**。これらの関数は文字列のシーケンスから最大値および最小値を取得することができます。

MapForce バージョン 2015 R4 の新規機能 :

- 内部的な更新と最適化

MapForce バージョン 2015R3 の新規機能 :

- XML 出力内の `<?xml ... ?>`宣言を抑制するオプション
- 新しいコンポーネント型 : 単純型出力
- 内部的な更新と最適化

MapForce バージョン 2015 の新規機能 :

- 新規 language 引数はformat-date とformat-dateTime 関数で使用することができます
- 新規 シーケンス 関数 :replicate-item

MapForce バージョン 2014R2 の新規機能：

- 新規 シーケンス 関数: シーケンスの生成、item-at など。
- 出力コンポーネント内の CDATA セクションを定義する機能のためのタイムアウトの値を定義する機能
- コンポーネントの削除後接続を保存する方法
- ターゲットコンポーネント内の必須アイテムの自動ハイライト

MapForce バージョン 2014 の新規機能:

- RaptorXML 検証への統合と XML Schema 1.1への基本的なサポート
- 新規 RaptorXML XSLT エンジンの統合
- XML スキーマワイルドカードへのサポート、xs:any とxs:anyAttribute
- XML ターゲットコンポーネント内のコメントと処理命令へのサポート

MapForce バージョン 2013R2 SP1 の新規機能:

- 高速変換エンジン

MapForce バージョン 2013R2 の新規機能:

- 内部的な更新と最適化

MapForce バージョン 2013 の新規機能:

- 内部的な更新と最適化

MapForce バージョン 2012R2 の新規機能:

- XSLT 2.0、XQuery、内蔵の実行エンジンにて ソートコンポーネント をサポート
- ユーザー定義コンポーネント名

MapForce バージョン 2012 の新規機能:

- ウィンドウマッピングウィンドウにおけるコンポーネントの自動配置
- ターゲット親 ノードへの接続を通知
- マッピングにてコンポーネントのシーケンスを制御するルール

MapForce バージョン 2011R3 の新規機能:

- 中間変数

MapForce バージョン 2011R2 の新規機能:

- ライブラリウィンドウライブラリウィンドウにおける検索機能
- 反転 マッピング
- 拡張可能な IF-ELSE 関数
- Core ライブラリにおけるノード名ならびに解析関数

MapForce バージョン 2011 の新規機能:

- マッピングチェーンにて、中間コンポーネントのプレビュー (パススループレビュー)
- サポートされている全ての言語において dateTime と numbers のフォーマット機能が利用可能に
- 自動連番 機能の強化

MapForce バージョン 2010 リリース 3 の新規機能:

- Nillable 値 ならびに XML インスタンスファイルにおける xsi:nil 属性をサポート
- XML ドキュメントにてターゲットに対する自動キャスティングを無効化

MapForce バージョン 2010 リリース 2 の新規機能:

- 親コネクタの接続時に、対応する子接続を自動的に接続
- 入力文字列の トークン化により更なる処理

MapForce バージョン 2010 の新規機能:

- コンポーネント毎に複数の入力 /出力サポート
- 相対パス サポートのアップグレード
- xsi:type のサポートによる派生型の使用
- 新たに追加されたデータ型システム
- ユーザー定義の関数ナビゲーションの改善
- XML 要素における複合コンテンツの処理を強化

MapForce バージョン 2009 SP1 の新規機能:

- ユーザー定義関数における パラメーターの順序を定義
- XML スキーマに対して妥当ではない XML ファイルの処理
- 全般的 (標準的) なユーザー定義関数における複雑な階層構造の引数サポート

MapForce バージョン 2009 の新規機能:

- EDI HL7 versions 3.x XML をソースならびにターゲットコンポーネントとして使用可能に
- ノードコンテンツやノードのグループ化
- シーケンス内の ノードの位置 をベースにしたデータのフィルタリング
- QName サポート
- コンポーネント内にあるアイテム/ノードの 検索

MapForce バージョン 2008 リリース 2 の新規機能:

- XML ファイルのために XML スキーマを自動生成
- Altova グローバルリソースのサポート

- パフォーマンスの最適化

MapForce バージョン 2008 の新規機能:

- 集計 関数
- Value-Map 検索コンポーネント
- XML 出力オプションの強化: XML 出力の整形、個々のコンポーネントに対してのエンコーディング設定や XML スキーマ参照の省略
- 内部における様々なアップデート

Chapter 2

はじめに

2 はじめに

MapForce の機能、ユーザーインターフェイス、MapForce の基本概念、およびこのドキュメントで使用される規約を紹介します。

2.1 MapForce とは ?

MapForce は、プログラムコードを作成することなく視覚的なドラッグアンドドロップを使用するグラフィカルなユーザーインターフェイスを用いデータを異なるデータ間、スキーマ間で変換するWindows ベースの多目的のIDE (統合された開発環境 統合開発環境) です。また、事実上、MapForce は時際のデータ変換 (またデータマッピング)を行うプログラムコードを生成します。プログラムコードを生成しない場合、 (MapForce Professional または Enterprise Editionsで使用することのできる)MapForce 内蔵の変換言語 を実行します。

MapForce を使用して、便利にファイルベースおよび他のフォーマットの多種のデータへ、またはこれらからからデータを変換することができます。使用する技術に関わりなく、MapForce は自動的にデータの構造を決定し、データのためのスキーマを提供するオプションを提供し、サンプルインスタンスファイルから自動的に生成します。例 :例えば、XML インスタンスファイルがあり、スキーマ定義がない場合 MapForce はスキーマを、XML ファイル内で使用することのできるデータを他のファイルまたはフォーマットでマッピングのために使用できるように生成することができます。

マッピングソースまたはターゲットとしてサポートされるテクノロジーは、以下のとおりです。

MapForce Basic Edition	MapForce Professional Edition	MapForce Enterprise Edition
• XML とXML スキーマ • HL7 バージョン 3x (schema-based)	• XML とXML スキーマ • コンマで区切られた値 (CSV) と固定長フィールドフォーマット (FLF) を含むフラットファイル • データベース (Microsoft Access や SQLite データベースなどを含む全てのメジャーなデータベース)	• XML とXML スキーマ • コンマで区切られた値 (CSV) と固定長フィールドフォーマット (FLF) を含むフラットファイル • マップし、MapForce FlexText を使用して、他のフォーマットに変換することのできるレガシーテキストファイルからのデータ。 • データベース (Microsoft Access や SQLite データベースなどを含む全てのメジャーなデータベース) • (UN/EDIFACT、ANSI X12、HL7、IATA PADIS、SAP IDoc、TRADACOMSなどを含む)EDI フォーマットのファミリー • JSON ファイル • Microsoft Excel 2007 と以降のファイル • XBRL インスタンスファイルとタクソノミ

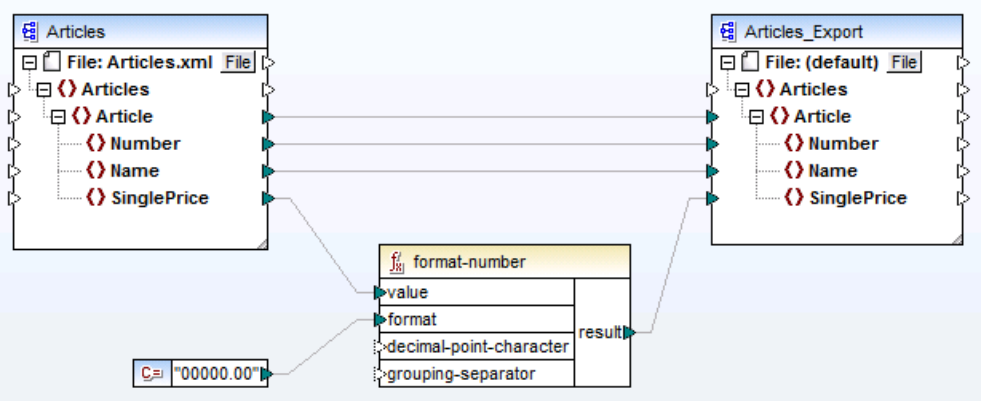
MapForce エディションにより、データの変換言語を以下のように選択することができます。

MapForce Basic Edition	MapForce Professional Edition	MapForce Enterprise Edition
• XSLT 1.0 • XSLT 2.0	• MapForce 内蔵の変換言語 • XSLT 1.0 • XSLT 2.0 • XQuery • Java • C# • C++	• MapForce 内蔵の変換言語 • XSLT 1.0 • XSLT 2.0 • XQuery • Java • C# • C++

全ての変換の結果、および生成された XSLT または XQuery コードをグラフィカルユーザーインターフェイスを移動することなくをプレビューすることができます。デザインまたはマッピングをプレビューする際、MapForce はスキーマおよび変換の整合性を検証し、検証エラーを専用のウィンドウで表示するため、すぐに確認して、エラーの修正を行うことができます。

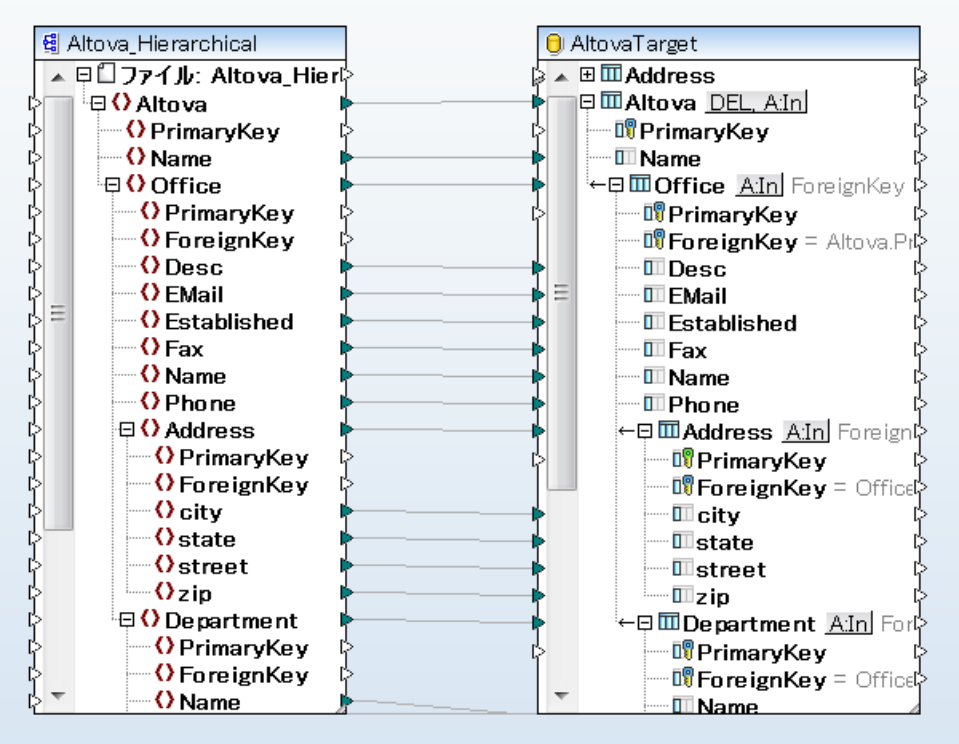
Java、C#、または C++ を変換言語として選択する場合、MapForce は必要とされるプロジェクトとソリューションを生成し、Visual Studio または Eclipse で直接開き、生成されたデータマッピング プログラムを実行することができます。高度なデータ統合のシナリオの場合、自身のコードを用い Altova と MapForce API を使用して、生成されたプログラムを拡張します。

MapForce では、全てのマッピングの変換を視覚的にデザインすることができます。例 : XML の場合、全ての要素と属性を接続することができます、または要素または 属性 他の XML ファイルの要素または属性の XML ファイルにコメントを挿入します。これにより、MapForce にソース要素 (または 属性)からデータを読み込み、ターゲット要素 (または 属性)に書き込むことを命令します。



2 つのXML ファイル間のサンプルデータ変換

同様に、MapForce Professional または Enterprise Editions でデータベースと作業する場合、MapForce のマッピングエリア内のデータベースカラムを確認し、データをマップし、視覚的な接続を作成することができます。Altova MissionKit 製品と同様、MapForce からデータベース接続を設定する場合、柔軟的にデータベースドライバーと接続の型 (ADO、ODBC、または JDBC) を既存のインフラストラクチャとデータマッピングの必要に応じて選択することができます。更に、 SQL クエリを視覚的に作成し、ストアドプロシージャを使用し、または、データベースに直接問い合わせることができます。 (データベース型、エディション、ドライバーによりサポートは異なります)



XML ファイルとデータベース間のサンプルデータ変換

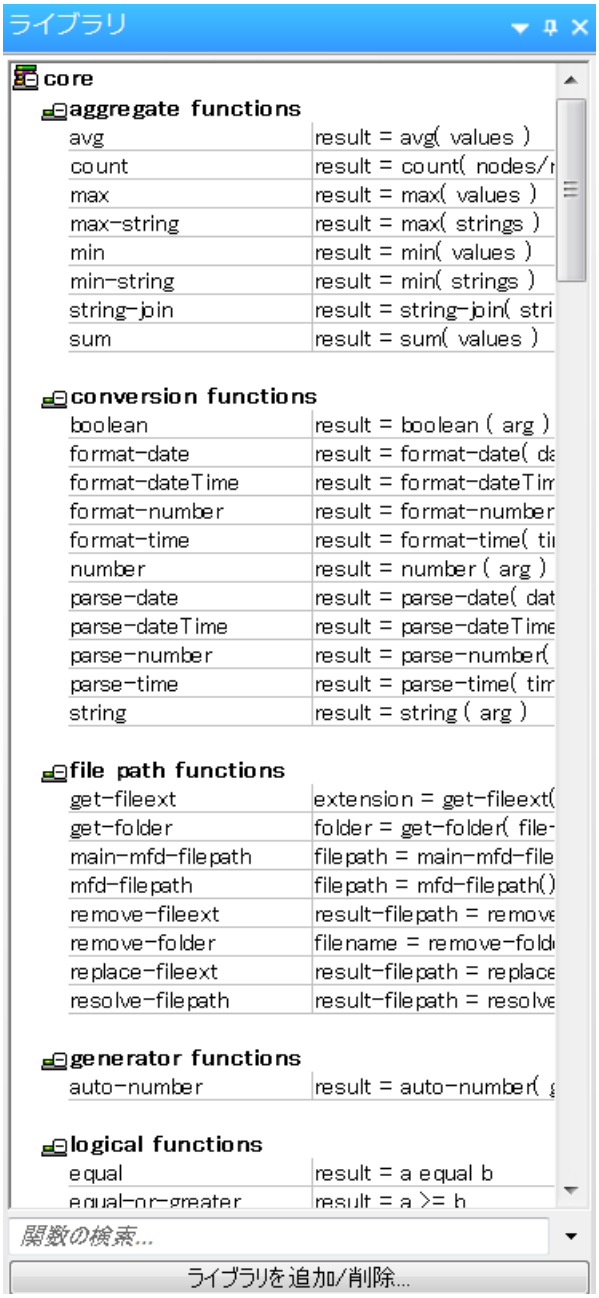
MapForce を使用したマッピングデザインのとても簡単な例として、 「ソースX からデータを読み込み ターゲットY に書き込む"が挙げられます。しかしながら、 「ソースX からデータを読み込み、 ターゲットY に書き込む。そしてソースY からデータを読み込み ターゲットZ へ書き込む"などのMapForce シナリオを簡単にデザインすることができます。これらは 「パススルー"または 「チェーン"マッピングとして知られ、これらにより変換プロセスの中間のステージでデータにアクセスすることができます (例えばファイルとして保存するため)。

MapForce 内で作成することのできるデータマッピングは、単一かつ定義済みファイルだけではありません。同じ変換内で、ディレクトリから複数の入力ファイルを動的に処理することができ、複数の出力ファイルを生成することができます。ですから、 「複数のX ファイルからデータを読み込み、単一のY ファイルに書き込む"または "ファイルX を読み込み複数の ファイルY を生成する"といったシナリオが可能になります。

重要な点は、同じ変換で、使用中のMapForce エディションでサポートされる全てのデータ型の複数の ソースと複数のターゲットを混合できる点です。例 : MapForce Professional またはEnterprise の場合、2つの異なるデータベースを単一のXML ファイルにマージすることが可能です。または、複数のXML ファイルからのデータをマージし、一部のデータを1つのデータベースとして書き込み、一部のデータを他のデータベースとして書き込むことができます。SQL ステートメントをデータベースにコミットする前にプレビューすることができます。

ソースからターゲットへのデータの直接の変換は通常の変換目的ですが、多くの場合、データを特定の方法で処理する必要がある場合があります (例 :並べ替え、グループ化、または フィルター)などを最終プロセスの前に行う必要があります。この理由のため、MapForce は、プログラミング言語の構成を簡素化するその他の関数コンポーネント (規則、変数、SQL-WHERE 条件、フィルターと並べ替えコンポーネントなど)を搭載しています。また、MapForce は、全てのデータ操作をアシストする豊富かつ広範囲の関数ライブラリも搭載しています。

必要であれば、ビルドインライブラリを、MapForce で直接関数 (いわゆるユーザー定義関数、またはUDF) をデザインすることにより、またはXSLT、XQuery、Java、 またはC# 言語で外部的に作成された関数やライブラリを使用して拡張することも可能です



ライブラリペイン (MapForce Basic Edition)

データマッピングデザインファイルの数が多すぎる場合、マッピングをプロジェクトに整理することができます (MapForce Professional と Enterprise エディションで使用することができます)。これによりより簡単なアクセスと管理を行うことができます。重要な点は、プロジェクト内の個別のマッピングのためにコードを生成し、更にプログラムコードをプロジェクト全体から生成することができます。

(MapForce Server API を使用したマッピング変換の実行時などの) 高度なデータ処理のために マッピングをデザインし、ランタイムで値をパス、またはランタイムから単純型文字列の値を取得することができます。この機能は、関数または単純型文字列の値を生成するマッピング全体を素早くテストすることを可能にします。MapForce の Professional と Enterprise エディションは、他のプログラミング言語内で同様に動作する、ランタイムでの文字列の解析とシリアル化を行うコンポーネントも含みます。

MapForce Enterprise Edition では、Web サービス言語定義 (WSDL) ファイルをベースにした SOAP 1.0 と SOAP 2.0 ウェイブサービスを視覚的にデザインすることができます。WSDL 1.0 または WSDL 2.0 Web サービスからデータをマッピング内部から取得することができます。これは、HTTP と HTTPS プロトコールを使用して、ユーザーが WS- Security メカニズムまたは HTTP 認証を使用することを必要とするWeb サービスで使用することができます。

MapForce Professional と Enterprise Editions では、マッピングデザイン ファイルの詳細なドキュメントを HTML、Word 2007+ または RTF フォーマットで生成することができます。 例 :ドキュメントから特定のコンポーネントを包含したり削除することが選択可能です)。

MapForce を他の Altova MissionKit 製品と併せて使用している場合、MapForce は、これらの製品と Altova サーバーベースの製品を以下のテーブルに表示されているように統合します。

MapForce Basic Edition	MapForce Professional Edition	MapForce Enterprise Edition
生成された XSLT を直接 MapForce 内で実行し、データ変換の結果をすぐに確認することができます。パフォーマンスを向上するには、超高速 XML 変換エンジン RaptorXML Server を使用して、マッピングを処理します。		
XMLSpy がインストールされている場合、同じマシンで、関連する MapForce コンテキストから直接 XMLSpy を開き、サポートされるファイル型を便利に開き編集することができます。 (例 : **「コンポーネント | XMLSpy 内でスキーマ定義を編集」** メニューコマンドは XML コンポーネントをクリックすると使用することができます)。		
	データ変換を、直接 MapForce 内で、またはコマンドラインや自動化された実行を使用して異なるマシンやオペレーティングシステムにデプロイし実行することができます。 具体的な方法は、Windows 上でマッピングをデザインし、MapForce Server または FlowForce Server が動作する Windows、Linux、または Mac サーバーマシンで実行することができます。
	StyleVision が同じマシンにインストールされている場合、デザインし、既存の ,StyleVision スタイルシートを再利用し、マッピングの結果を HTML、RTF、PDF、または Word 2007+ ドキュメントとしてプレビューすることができます。

MapForce Professional と Enterprise エディションは、Visual Studio と Eclipse の統合された開発環境のプラグインとしてインストールすることができます。これにより、優先する開発環境を移動することなく、MapForce の機能にアクセスしマッピングをデザインすることができます。

MapForce では、開発環境の概観や使用感を完全にカスタム化できるだけでなく (グラフィカルユーザーインターフェイス) 各技術と各マッピングのコンポーネントの型の他の関連する設定 もカスタム化することができます。例 :

- XML から、または、XML へ マッピングを行う際、スキーマレファレンスを含むか否かを選択することができ、または XML 宣言 が XML 出力ファイル内で抑制されるべきかを選択することができます。生成されたファイルのエンコードを選択することができます (例 :UTF-8
- データベースから、または、データベースへマッピングする際、データベース ステートメントの実行のタイムアウト期間、MapForce がデータベーストランザクションを使用するか、または、コード生成の際データベーススキーマ名からテーブル名を削除するかなど、などの設定を定義することができます。
- XBRL の場合、MapForce が表示する構造ビューを選択することができます (「プレゼンテーション定義リンクベース"ビュー、 「テーブルリンクベース」ビュー、または 「全てのコンセプト」ビューなど)。

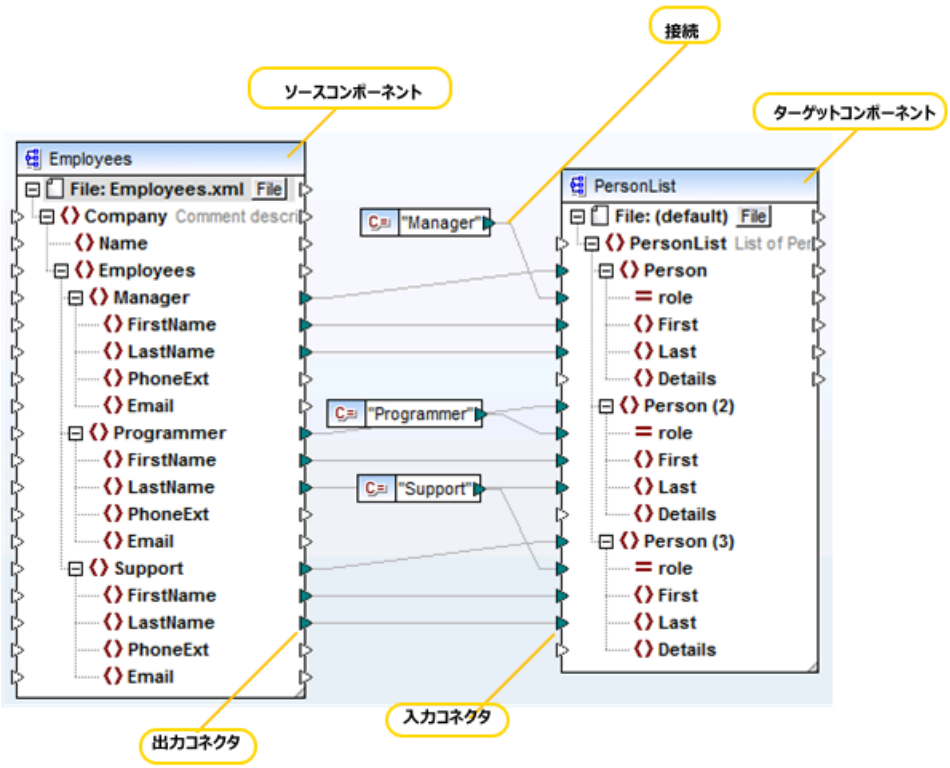
MapForce の全てのエディションは 32-ビットアプリケーションで動作します。更に、MapForce Professional と Enterprise エディションは 64-ビットアプリケーションで動作します。

2.2 基本概念

このセクションはデータマッピングを開始するにあたり必要な基本概念のアウトラインを紹介します。

マッピング

MapForce マッピングデザイン (または、略して "マッピング") は、データがどのように1つのフォーマットから他のフォーマットに変換されるかの視覚的な表現です。マッピングは、データ変換を行うためのMapForce マッピングエリア内で1つのスキーマから他のスキーマに追加するコンポーネントから構成されています。 (例 :XML ドキュメントを1つのスキーマから他のスキーマに変換)。有効なマッピングは、1つまたは複数のターゲットコンポーネントに接続されている、1つまたは複数の複数のソースコンポーネントから構成されています。マッピングを実行して、結果を直接 MapForce 内で確認することができます。コードを生成し、外部で実行することも可能です。MapForce 実行可能ファイルにマッピングをコンパイルし、MapForce Server またはFlowForce Server を使用してマッピングの実行を自動化することもできます。 MapForce は、マッピングを .mfd の拡張子を持つファイルとして保存します。



MapForce マッピングの基本的構造

コンポーネント

MapForce 内では、 "コンポーネント" という用語はデータの構造 (スキーマ) またはデータがどのように変換 (関数) されるかを視覚的に表します。コンポーネントはすべてのマッピングを構築する中心的な役割を果たします。以下は、MapForce コンポーネントの例です :

- 定数

- フィルター
- 条件
- 関数コンポーネント
- EDI ドキュメント (UN/EDIFACT、ANSI X12、HL7)
- Excel 2007+ ファイル
- 単純型 入力コンポーネント
- 単純型 出力コンポーネント
- スキーマおよびDTD

コネクタ

コンポーネントの左右に表示される小さな三角形です。コンポーネントの左に表示されるコネクタはそのコンポーネントのエントリポイントを表します。コンポーネントの右に表示されるコネクタはそのコンポーネントからのデータの終了ポイントを表します。

接続

接続とは2つのコネクタ間に描くことができる線です。接続を描くことにより、MapForce にデータを特定の方法で変換するように命令します。(例 :XML ドキュメントからデータを読み込み、他のXML ドキュメントに書き込みます)。

ソースコンポーネント

ソースコンポーネントとは、MapForce がデータを読み込む元のコンポーネントです。マッピングを実行すると、ソースコンポーネントのコネクタにより与えられたデータを読み込み、必要とされる型に変換し、ターゲットコンポーネントのコネクタに送信します。

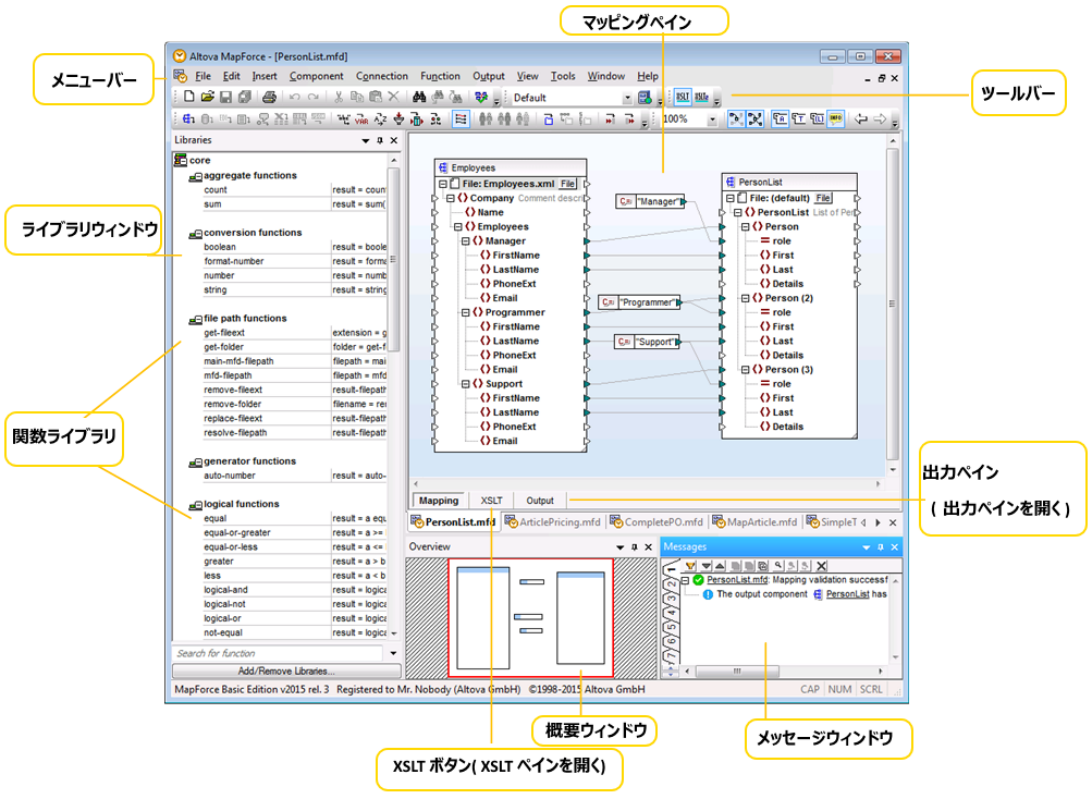
ターゲットコンポーネント

ターゲットコンポーネントとMapForce はがデータを書き込むコンポーネントです。マッピングを実行すると、ターゲットコンポーネントは、外部プログラム内で更に処理するためにMapForce にファイル (または複数のファイル)の生成、または結果を文字列の値として出力するように命令します。ターゲットコンポーネントは、ソース コンポーネントの逆です。

2.3 ユーザーインターフェイスの概要

MapForce のグラフィカルインターフェイスは、統合された開発環境として整理されています。メインのインターフェイス設定は **「ツール| カスタマイズ」** メニューコマンドを使用して行います。

各ウィンドウの右上に表示される ▾ 📌 ✕ ボタンを使用してウィンドウの、表示、非表示、ピン、ドックを行います。ツールバーとウィンドウをデフォルトの状態に復元する場合は、メニューコマンド **「ツール| ツールバーとウィンドウの復元」** を使用します。



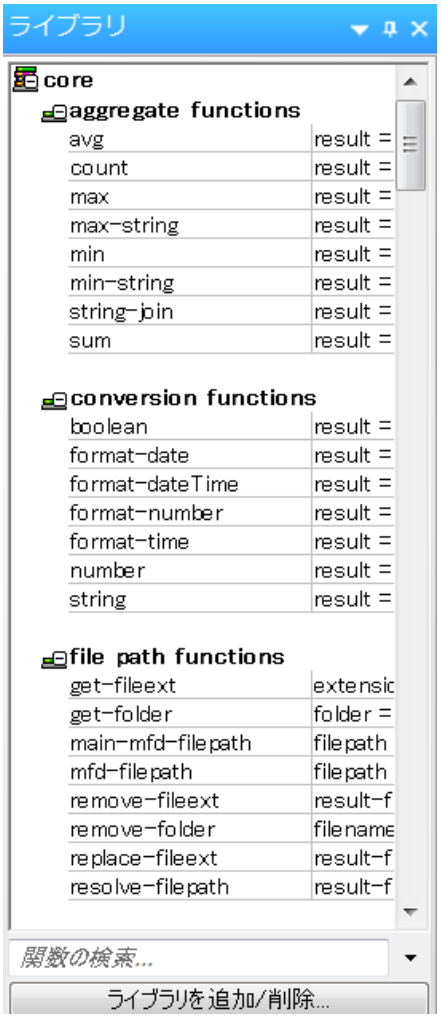
MapForce グラフィカルユーザーインターフェイス (MapForce Basic Edition)

メニューバーとツールバー

メニューコマンドは、メニューアイテムを表示します。各ツールバーは MapForce コマンドのボタンのグループを表します。希望する場所にハンドルをドラッグすることにより、ツールバーを移動することができます。

ライブラリウィンドウ

ライブラリウィンドウは、ライブラリごとに整理された MapForce 内蔵の関数をリストします。使用することのできる関数は選択する変換言語によりことなります。ユーザー定義関数を作成した場合、またはインポートした場合、これらの関数もライブラリウィンドウに表示されます。



名前または詳細による関数の検索は、**ライブラリ**ウィンドウの下のテキストボックスに検索の値を入力することにより行うことができます。現在アクティブなマッピング内で関数の全ての発生を検索するには、関数を右クリックして、コンテキストメニューから**全ての呼び出し**を選択します。また、**ライブラリ**ウィンドウから直接関数データ型と詳細を確認することができます。詳細に関しては、関数と作業を参照してください。

マッピング ペイン

マッピングペインはマッピングをデザインする作業エリアです。 **挿入**」メニュー（コンポーネントをマッピングに追加を参照）からマッピングエリアに（ファイル、スキーマ、コンスタント、変数などの）マッピングコンポーネント追加することができます。ライブラリウィンドウに表示されたマッピングペイン関数をドラッグすることもできます（関数と作業を参照）。

XSLT (XSLT2) ペイン

XSLT ペインは、 **『XSLT**」ボタンをクリックすると、マッピングから生成されたXSLT 1.0 変換コードを表示します。このペインはXSLT を変換言語として選択し、**XSLT** タブ（または**XSLT2** タブ）をクリックすると使用することができます。

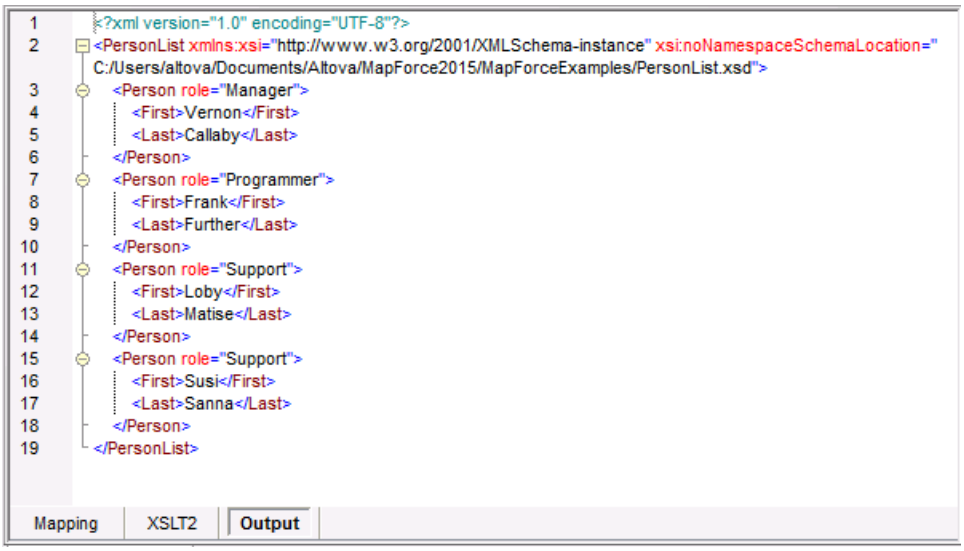
このペインは、ラインの付番とコードの折りたたみ機能を提供します。コードの一部を展開または折りたたむには、ウィンドウの左側の「+」と「-」アイコンをクリックします。折りたたまれたコードは、省略記号と共に表示されます。折りた

たまれたコードを展開することなくプレビューするには、マウスで省略記号をポイントすると、プレビューされるコードを表示したヒントが下に表示されるように開かれます。プレビューされたテキストが大きくヒント内に表示できない場合、ヒントの後に追加表示記号が表示されます。

(インデント、行末マーカーを含む)表示設定を構成するには、ペインを右クリックして、コンテキストメニューから **「テキストビュー設定」**を選択します。または、 **「テキストビュー設定」**( ) ツールバーボタンをクリックします。

出力ペイン

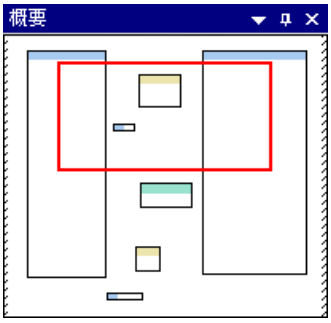
出力ペインは、 **「出力」** ボタンをクリックするとマッピングの変換結果を表示します(例:XML ファイル)。マッピングが複数のファイルを生成する場合、それぞれの生成されたファイルを順番にナビゲートすることができます。



XSLT ペインと同様の仕組みを持つラインの付番とコードの折りたたみ機能を提供します(上を参照)。

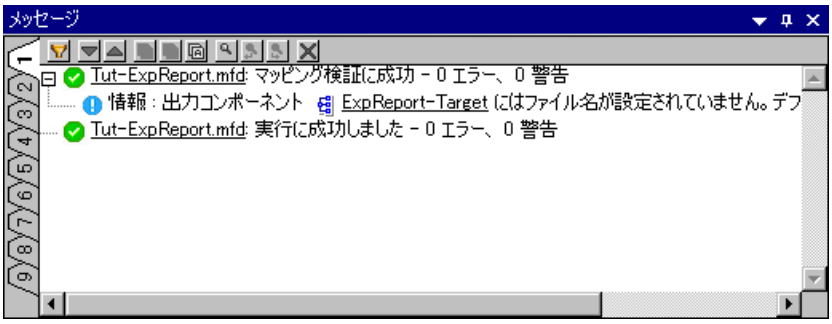
概要ウィンドウ

概要ウィンドウは、マッピングペインの概観図を表示します。マッピングのサイズが大きい場合、マッピングエリアの特定の場所に移動する場合に使用します。マッピングの特定の場所に移動する場合、赤い枠の長方形をクリックアンドドラッグします。



メッセージウィンドウ

メッセージウィンドウにはマッピング処理中（出力のプレビューを参照）に行われる検証結果（マッピングの検証を参照）の警告やエラーメッセージが表示されます。



情報、警告またはエラーメッセージをトリガーしたコンポーネントまたは構造のマッピングエリアをハイライトするには、メッセージウィンドウ内の下線のついたテキストをクリックします。

メッセージウィンドウ内のマッピング実行または検証オペレーションの結果は、以下のアイコンの1つと共に表示されます：

アイコン	説明
	オペレーションに成功しました。
	オペレーションは警告と共に終了しました。
	オペレーションに失敗しました。

メッセージウィンドウは次のメッセージを追加して表示する場合があります：情報メッセージ、警告、およびエラー

アイコン	説明
	情報メッセージを表示します。情報メッセージはマッピングの実行を停止しません。
	警告メッセージを表示します。警告はマッピングの実行を停止しません。例えば、必須の入力コネクタへの接続が作成されなかった場合、警告メッセージが生成されます。このような場合、有効な接続を持つコンポーネントのために出力が生成されます。
	エラーメッセージを表示します。エラーが発生すると、マッピングの実行に失敗し、出力は生成されません。XSLT またはXQuery コードのプレビューの不可能です。

メッセージウィンドウ内の他のボタンにより以下のアクションを実行することができます：

アイコン	説明
	重要度（情報メッセージ、エラー、警告）別にメッセージをフィルターします。 **「すべて選択」**を選択して、全ての重要度を選択します（これはデフォルトの振る舞いです）。 **「チェックをすべて解除」**を選択して、フィルターからすべての重要度を解除します。この場合、全般的な実行または検証の状況に関するメッセージのみが表示されます。
	選択を次の行に移動させる。
	選択を前の行に移動させる。
	選択された行をクリップボードにコピーする。
	ネストされたラインを含む選択されたラインをクリップボードにコピーします。
	メッセージウィンドウの全てのコンテンツをクリップボードにコピーします。
	メッセージウィンドウ内で特定のテキストを検索します。オプションで、特定の用語を検索する場合は、**「単語の完全マッチ」**を選択します。単語の大文字と小文字を区別する場合**「大文字と小文字を区別」**を選択します。
	特定のテキストを現在選択されているラインから最後まで検索します。
	特定のテキストを現在選択されているラインから最初まで検索します。
	メッセージウィンドウをクリアする。

複数のマッピングファイルと同時に作業する場合、個々のマッピングのための個別のタブ内で情報、警告またはエラーを表示する必要があることがあります。この場合、マッピングを実行または検証する前に、メッセージウィンドウの左横にある番号の付いたタブをクリックします。

アプリケーションステータスバー

アプリケーションウィンドウの下部にあるアプリケーションステータスバーには、アプリケーションに関する情報が表示されます。ツールバーアイコンに対してマウスポインターを重ねると、ツールチップ情報がこのステータスバーに表示されます。アプリケーションステータスバーとデータベースクエリステータスバーは別のものです。64ビットバージョンのMapForceを使用している場合、アプリケーション名の後に (x64) という文字列が加えられます。32ビットバージョンでこのような表示は行われません。

2.4 規約

サンプルファイル

このドキュメントで参照される(.mfd 拡張子 と他の伴うインスタンスファイルを持つファイル)データマッピングデザインファイルの多くは次のフォルダーで検索することができます:

- (My) Documents\Altova\MapForce2017\MapForce Examples
- (My) Documents\Altova\MapForce2017\MapForce Examples\Tutorials

(マイ)ドキュメント フォルダーの場所はWindows のバージョンにより異なります

Windows 10 Windows 8 Windows 7 Windows Vista Windows Server 2008 Windows Server 2012	`C:\Users\<username>\Documents`
Windows XP Windows Server 2003	`C:\Documents and Settings\<username>\My Documents`

MapForce に伴うサンプルマッピングとインスタンスファイルは動作の多くのアスペクトを紹介し、MapForce を使用するに当たり色々と試すことが奨励されます。オリジナルのサンプルへ直接変更を加えることを希望しない場合は、変更前にバックアップを作成することが奨励されます。

グラフィカルユーザーインターフェイス

このドキュメントに伴うイメージ (スクリーンショット)の一部は、使用中のMapForce エディションでは、使用することのできないグラフィカルユーザーインターフェイスの要素を含む場合があります。関連したコンテキストで、イメージは通常ソースマッピングデザイン (*.mfd) ファイルの名前、画像が作成されたMapForce のエディションを含みます。

Chapter 3

チュートリアル

3 チュートリアル

MapForce チュートリアルは、MapForce のデータの基本変換機能短い時間でを理解することをお手伝いします。チュートリアルの目的は全てのMapForce 機能を説明することではありませんが、MapForce の基本を順序を追って説明します。ですから、チュートリアルを順番どおりにフォローすることが奨励されます。コンセプトは順を追って複雑になるため、各コンセプトを次のコンセプトに移動する前に理解することが重要です。XML およびXML スキーマに対する基本的な知識はとても有利です。

XML を新しいスキーマに変換

このチュートリアルは、コードを書く必要なくXSLT 2.0 言語を使用してXML 構造から他にデータを変換する方法について説明します。また、 MapForce シーケンスとアイテム、マッピング接続作成、関数の使用方法、マッピングの検証およびプレビュー、と出力結果をディスクに保存する方法などについて学ぶことができます。

複数のソースから1つのターゲットにマップ

このチュートリアルは、異なるスキーマを持つ2つのXML ファイルからのデータの読み込み方法や単一のターゲットXML ファイルへのマージの方法を説明します。また、各マッピングコンポーネントの名前とインスタンスファイルの変更方法や入力の複製の方法などについて学ぶことができます。

複数のターゲットスキーマとの作業

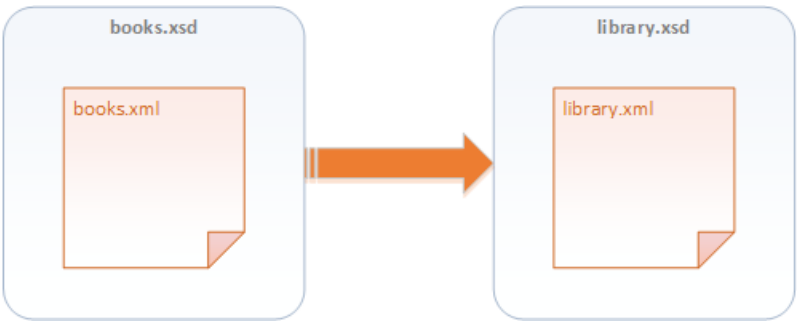
このチュートリアルは、2つ以上のターゲット出力を作成する更に複雑なマッピングとの作業方法について説明します。具体的な方法は、同じマッピング内でブックレコードのリストを保管するファイル、最初のXML ファイル内のブックのサブセットのみを含む他のXML ファイルを生成し、発行年によりフィルターします。データのフィルタリングをサポートするには、 **「フィルター」**コンポーネント、関数、数値定数を使用します。

ファイルを動的に処理と生成する

このチュートリアルは、同じフォルダーに存在する複数のXML インスタンスファイルからデータを読み込み複数のXML ファイルに書き込む方法を説明します。また、XML とスキーマ宣言 の削除および、関数を使用して、文字列を連結しファイル拡張子を抽出する方法についても学ぶことができます。

3.1 XML を新しいスキーマに変換

このチュートリアルは、MapForce 開発環境の基本を学びつつ、2つの XML ファイル間のデータの変換方法について説明します。両方の XML ファイルはブックのリストを保管しますが、その要素は、異なる方法により名前を付けられ整理されます。 (すなわち、2つのファイルは異なるスキーマを持ちます)。



データ変換の抽象的なモデル

下にリストされるコードは、データソースとして使用されるファイルからのサンプルのデータを表示しています (簡略化するために、XML と名前空間宣言は省略されています)。

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

books.xml

ターゲット (デスティネーション) ファイルではデータは以下のように表示されます:

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

```
      <publish_year>1912</publish_year>
   </publication>
</library>
```

library.xml

ソースとターゲットXML 内の一部の要素の名前は、同じではありません。このセクションの目的は、ソースファイル (<author>、<title>、<category>、<year>) 内の同等の要素から、ターゲットファイルの<author>、<title>、<genre> と<publish_year> 要素を表示することです。ソースXML ファイル内の属性 id は、ターゲットXML ファイル内の<id> 要素にマップされる必要があります。最後に、ターゲットファイルの<last_updated> 要素をファイルが最後に更新された日付と時刻として表示するように設定する必要があります。

必要とされるデータ変換を行うには、以下のステップを行います。

ステップ 1：XSLT2 を変換言語として選択する

これを以下の方法で行うことができます：

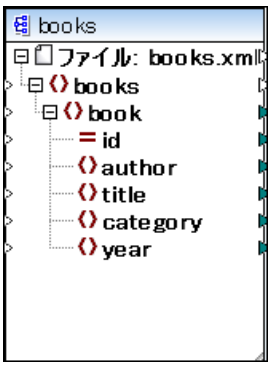
- **「XSLT2」**（ ）ツールバーボタンをクリックします。
- **「出力」** メニューから **「XSLT 2.0」** をクリックします。

ステップ 2：ソース XML ファイルをマッピングに追加する

このマッピングのためのソースXML ファイルは、は、以下のパスで見つけることができます：**<Documents>\Altova\MapForce2017\MapForceExamples\Tutorial\books.xml。** ファイルを以下の方法でマッピングに追加することができます：

- **「XML スキーマ /ファイルの挿入」**（ ）ツールバーボタンをクリックします。
- **「挿入」** メニューから **「XML スキーマ/ファイル」** をクリックします。
- XML ファイルをウィンドウエクスプローラーからマッピングエリアにドラッグします。

ファイルはマッピングエリアに追加され、構造を一見することができます。MapForce では、この構造はマッピングコンポーネントとして知られており、または単純にコンポーネント. コンポーネント内の要素を、折りたたむ (⊟) と展開するアイコン (⊞) をクリックすることにより、または数字キーパッドの **「+」** と **「-」** キーを押すことによりすることにより拡張することができます。



マッピング コンポーネント

マッピングペイン内でコンポーネントを移動するには、コンポーネントヘッダーをクリックして、マウスを新しい場所にドラッグします。コンポーネントのサイズを調整するには、コンポーネントの角をドラッグします ⌟。角をダブルクリックするとMapForce は自動的にサイズを調整します。

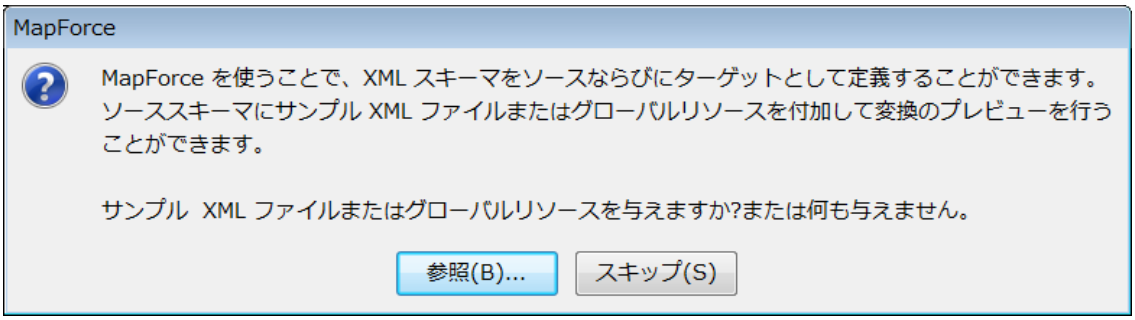
トップレベルノード は、ファイル名を表します。このサンプルの場合、は、XML インスタンスファイルの名前を表示します。構成内の、XML 要素は アイコンにより表示されており、XML 属性は アイコンにより表示されています。

コンポーネントの両側に表示されている小さな三角形は、(左側にある場合)データ入力を、(右側にある場合)データ出力を表します。MapForce では、それぞれ入力コネクタ、出力コネクタと呼ばれます。

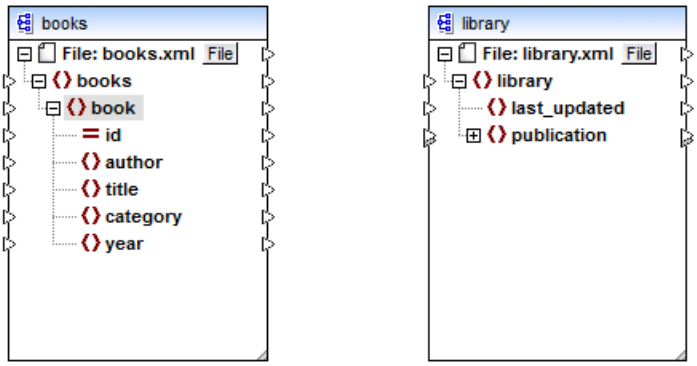
ステップ 3:ターゲット XML スキーマをマッピングに追加する

ターゲット XML を生成するには、既存の XML スキーマファイルを使用します。実生活のシナリオでは、このファイルは第三者パーティから提供される場合があり、またはXMLSpy などのツールを使用してユーザーが作成することができます。XML データのためのスキーマファイルが存在しない場合、MapForce は伴うスキーマまたはスキーマレファレンスを持たない XML ファイルをマッピングに追加すると、スキーマを生成するようにプロンプトします。

このサンプルでは、以下で見つけることのできる既存のスキーマファイルを使用します:**<Documents>\Altova\MapForce2017\MapForceExamples\Tutorial\library.xsd。** マッピングに追加するには、ソース XML ファイルと同様のステップを踏んでください。(**「XML スキーマ /ファイルの挿入」**( )ツールバーボタンをクリックします)。MapForce にインスタンスファイルを提供するようにプロンプトされると、**「スキップ」**をクリックします。



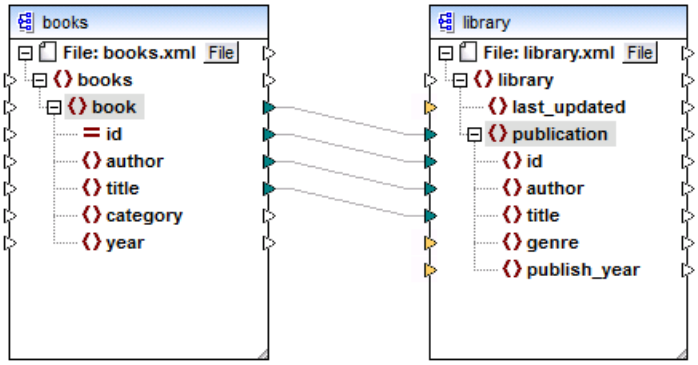
この時点では、マッピングデザインは、以下のようになります:



ステップ 4:接続の作成

ソース XML ファイルのそれぞれの<book> 内で、ターゲット XML ファイル内に新しい<publication> を作成します。 ですから、ソースコンポーネント内の<book> 要素とターゲットコンポーネント内の<publication> 要素を接続します。マッピング接続を作成するには、 <book> 要素の右の出力コネクタ(小さな三角形)をクリックし、ターゲット内の <publication> 要素の入力コネクタにドラッグします。

これを行うと、MapForce は自動的に、ターゲットファイル内に同じ名前を持つソースファイル内の<book> の子の全ての要素を接続します。ですから、4つの接続が同時に作成されます。この振る舞いは「子要素の自動接続」と呼ばれ、必要に応じて無効化またはカスタム化することができます。



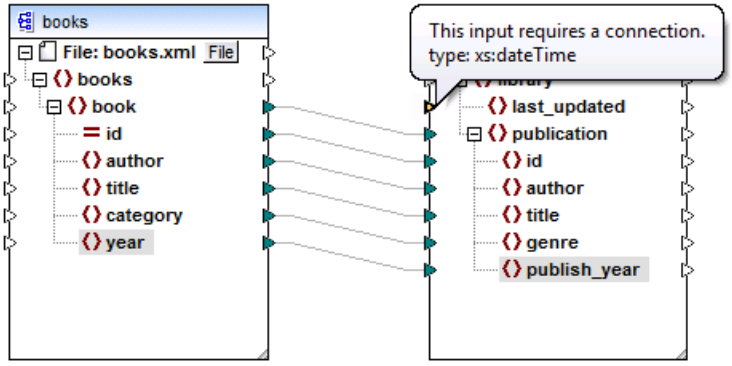
「マッチする子を自動的に接続する」の振る舞いを以下の方法で有効化または無効化することができます：

- **「子の自動接続の切り替え」**（ ）ツールバーボタンをクリックします。
- **「接続」**メニューから**「マッチする子を自動的に接続する」**をクリックします。

ターゲットコンポーネントの入力コネクタの一部が、MapForce により、必須アイテムを表示する、オレンジ色にハイライトされていることにご注意してください。XML ファイルの妥当性を確認するために、必須アイテムの値を以下のように提供してください：

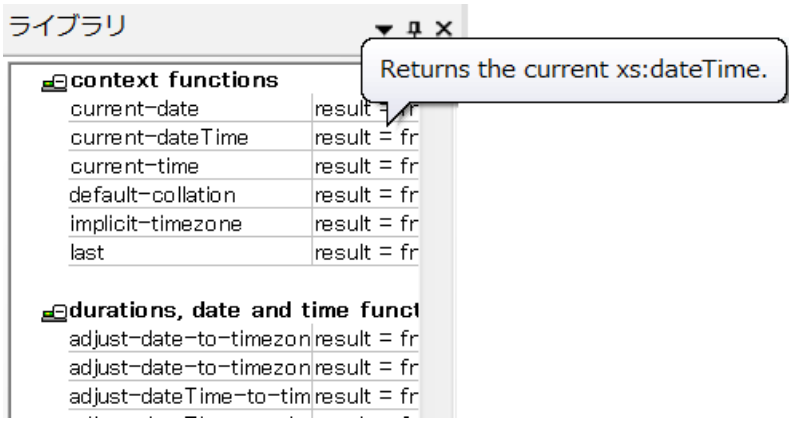
- ターゲット内の<genre> 要素とソース内の<category> 要素を接続する。
- ターゲット内の<publish_year> 要素とソース内の<year> 要素を接続する。

最後に、<last_updated> 要素への値を提供する必要があります。入力コネクタにマウスをポイントすると、要素は xs:dateTime 型であることが表示されます。ヒントを表示するには、**「ヒントの表示」**（ ）ツールバーボタンが有効化されている必要があることにご注意してください。



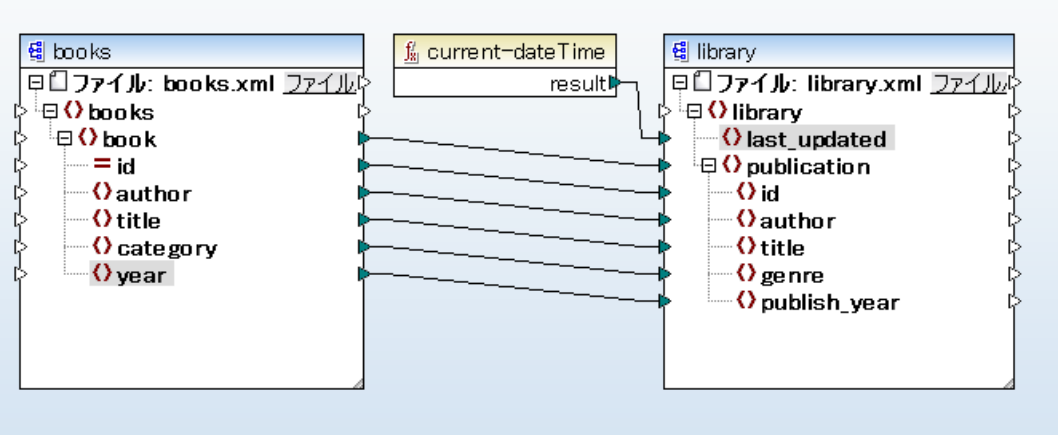
「データ型の表示」（ ）ツールバーボタンをクリックすることにより、常に各アイテムのデータ型を表示することができます。

現在の日時（すなわち、`xs:dateTime` の値）を関 XSLT 数の日付と時刻で取得することができます。マッピングに対するXSLT 関数を検索するには、ライブラリウィンドウの下の部分にあるテキストボックスに「date」を入力します。



上に表示されるように、マウスを関数の結果の部分にかざすと、詳細が表示されます。ヒントを表示するには、**「ヒントの表示」**（ ）ツールバーボタンが有効化されていることを確認してください。

マッピングに関数を追加するには、関数をマッピングペインにドラッグし、`<last_updated>` 要素の出力を入力に接続します。



(**books.xsd** スキーマを持つ)**books.xml** インスタンスファイルのデータから(**library.xsd** スキーマを持つ) **library.xml** ファイルへデータを変換するMapForce マッピングデザイン(または「マッピング」)を作成することに成功しました。各コンポーネントのヘッダーをダブルクリックすると、コンポーネント設定ダイアログボックス内のこれらと他の設定を確認することができます。

ソースのためのコンポーネント設定

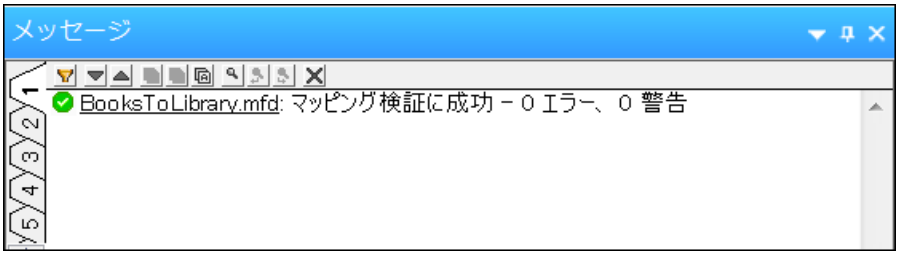
ターゲットのコンポーネント設定

ステップ 5：マッピングを検証し保存する

マッピングの検証は、マッピングを実行する前に、マッピングエラーの可能性を確認し修正するための任意のステップです。マッピングが妥当であるか確認するため、以下を行います：

- **「ファイル」**メニューから**「マッピングの検証」**をクリックします。
- **検証」**（ ）ツールバーボタンをクリックします。

「メッセージ」ウィンドウは、検証の結果を表示します：



メッセージウィンドウ

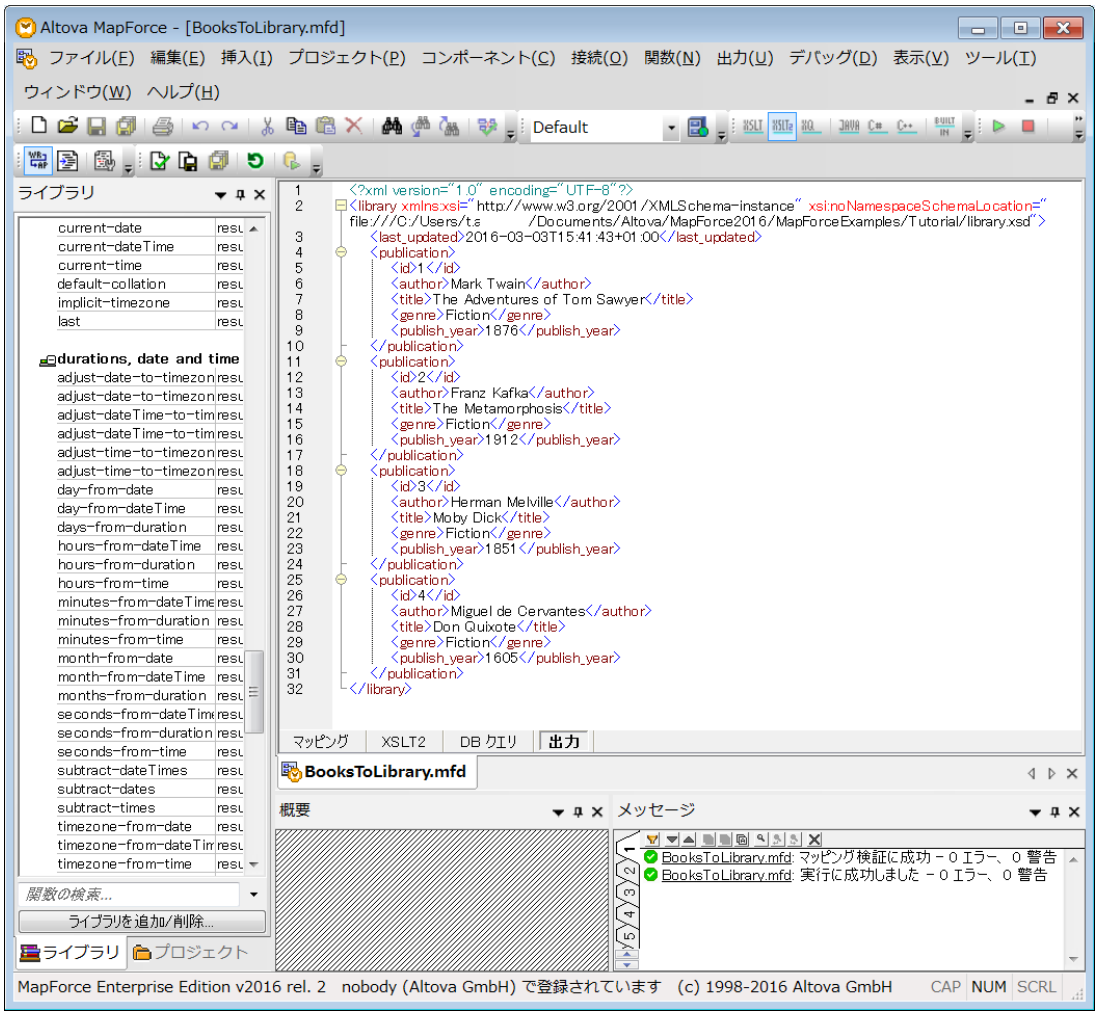
この時点で、ファイルにマッピングを保存します。マッピングを保存するには、以下を行います：

- **「ファイル」**メニューから**「保存」**をクリックします。
- **保存」**（ ）ツールバーボタンをクリックします。

このチュートリアルで作成されたマッピングは以下で検索することができます：**<Documents>\Altova\MapForce2017\MapForceExamples\Tutorial\\BooksToLibrary.mfd**。ですから、これから先、自分で作成したマッピングを継続して使用するか、または**BooksToLibrary.mfd** ファイルを使用することができます。

ステップ 6：マッピングの結果のプレビュー

マッピングの結果の結果を直接でMapForce でプレビューすることができます。これを行うには、マッピングペインの下にある、**出力」** ボタンをクリックします。MapForce は、変換を実行し、マッピングの結果を **出力」** ペインに表示します。



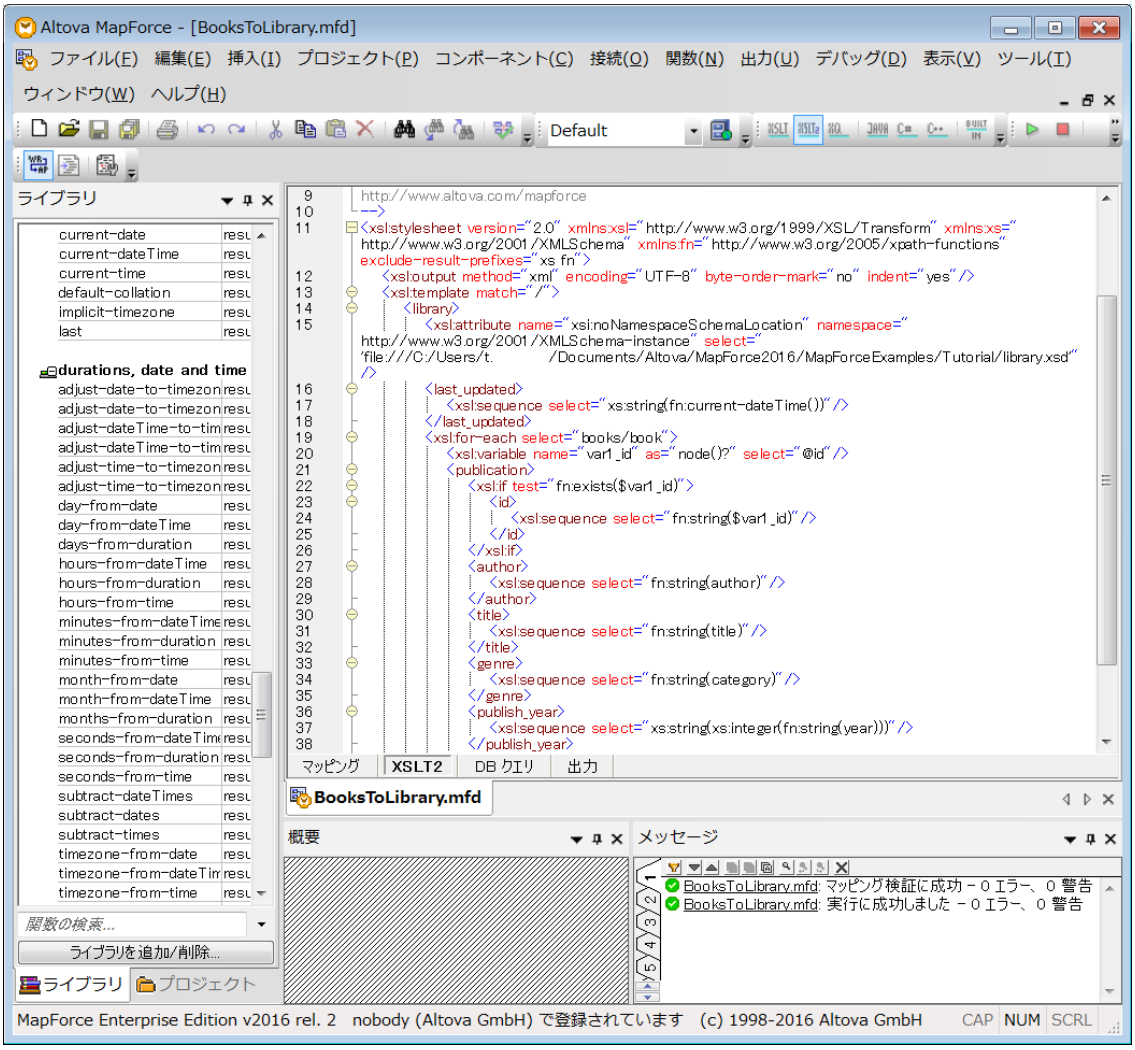
出力ペイン

MapForce 内で変換の結果を確認することができます。

デフォルトでは、 **出力** ペイン内にプレビューのために表示されたファイルはディスクに書き込まれません。その代わり、MapForce は一時ファイルを作成します。 **出力** ペインに表示されたファイルをディスクに保存するには、メニューコマンド **出力 | 出力ファイルの保存** を選択するか、または **生成された出力を保存** ( ) ツールバーボタンをクリックします。

MapForce が一時ファイルではなく最終のファイルに直接的に出力を書き込むように構成するには、 **ツール | オプション | 全般** に移動し、 **最終出力ファイルにファイルを書き込む** チェックボックスを選択します。チュートリアルを実行中は、故意にではなくもとのチュートリアルファイルを上書きする可能性があるため、このオプションを有効化することは奨励されません。

変換を行う生成されたXSLT コードもプレビューすることができます。コードをプレビューするには、マッピングペインの下にある、 **XSLT2** ボタンをクリックします。



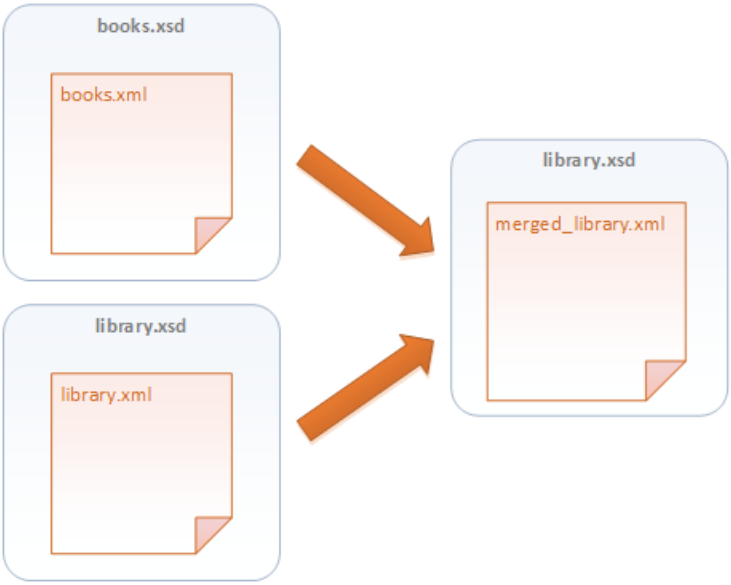
XSLT2 ペイン

XSLT2 コードを生成しファイルに保存するには、メニューアイテム **「ファイル | コード生成 | XSLT 2.0」** を選択します。プロンプトされると、生成されたコードが保存されるフォルダーを選択してください。コード生成が完了すると、デスティネーションファイルは、以下の2つのファイルを含みます：

1. ターゲットスキーマにちなんで名づけられたXSLT 変換ファイル (このサンプルでは、**MappingMaptolibrary.xslt**)。
2. **DoTransform.bat** ファイル。**DoTransform.bat** ファイルによりRaptorXML Server 内でのXSLT 変換の実行を有効化することができます。(詳細に関しては、http://www.altova.com/ja/raptorxml.html を参照してください)。

3.2 複数のソースから1つのターゲットにマップ

前のチュートリアルでは、ソースファイル (**books.xml**) からデータをターゲットファイル (**library.xml**) に変換しました。ターゲットファイル (**library.xml**) はマッピングの実行前は存在しておらず、マッピングの変換により生成されました。**library.xml** ファイル内にすでにデータが存在すると仮定し、このデータを**books.xml** から変換されたデータとマージするとします。このチュートリアルの目的は、**merged_library.xml** という名前のファイルを生成するマッピングをデザインすることです。生成されたファイルは、以下の2つのソースからのデータを含みます: **books.xml** ファイルと**library.xml** ファイル。ソースとして使用されるファイル (**books.xml** と**library.xml**) は異なるスキーマを持つことに注意してください。ソースファイルが同じスキーマを持つ場合、異なるアプローチを使用してデータをマージすることができます (ファイルを動的に処理と生成するを参照)。



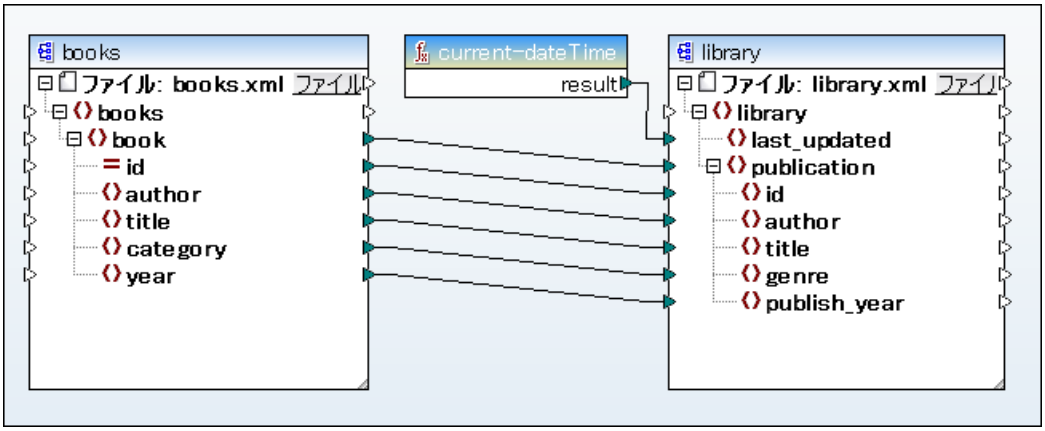
データ変換の抽象的なモデル

チュートリアルの目的を達成するためには、次のステップを踏みます。

ステップ 1: マッピングデザインファイルの準備

このチュートリアルは、**<Documents>\Altova\MapForce2017\MapForceExamples\Tutorial** フォルダーからの**BooksToLibrary.mfd** マッピングを開始点として使用します。このマッピングはXML を新しいスキーマに変換するチュートリアル内で作成済みです。チュートリアルを開始するには、MapForce で**BooksToLibrary.mfd** ファイルを開き、新しい名前で保存します。

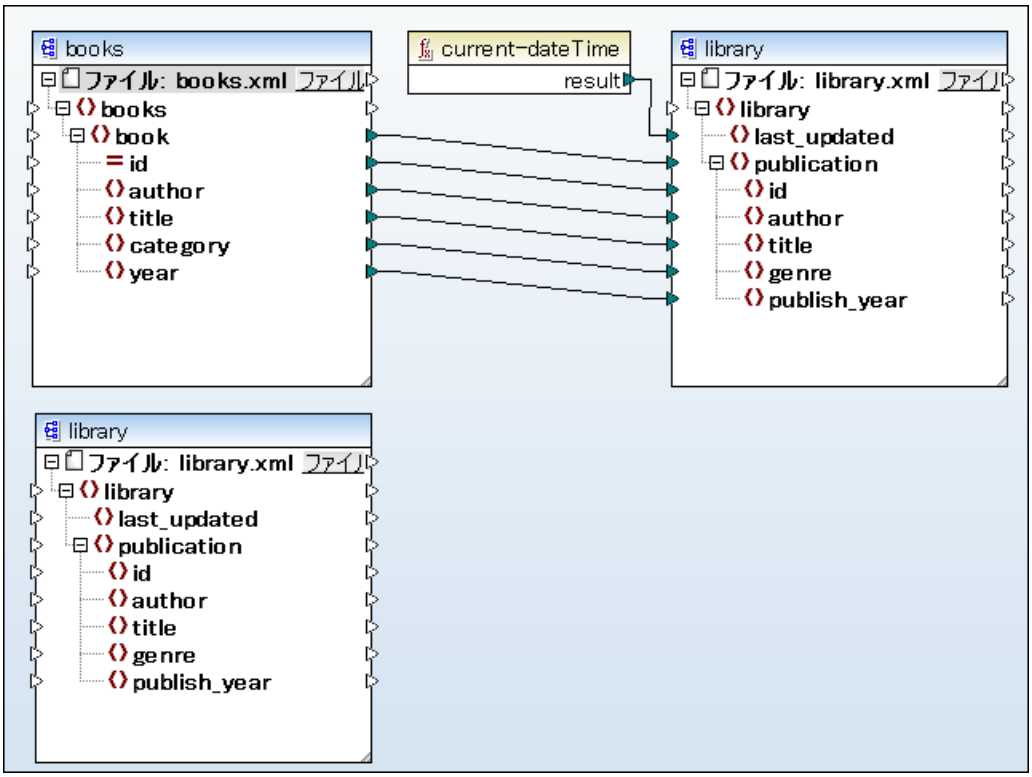
> 複数のファイルを参照するため、**<Documents>\Altova\MapForce2017\MapForceExamples\Tutorial** フォルダー内に新しいマッピングが保存されることを確認してください。



BooksToLibrary.mfd (MapForce Basic Edition)

ステップ 2 : 2番目のソースコンポーネントの作成

最初に、ターゲットコンポーネントを選択して、（**Ctrl + C**」を押して）コピーし、同じマッピング内に（**Ctrl + V**」を押して）貼り付けます。 新しいコンポーネントのヘッダーを**books** コンポーネントの下にドラッグします。



マッピングには2つのソースコンポーネントと:**books** と**library**、と 1つのターゲットコンポーネントがあります :**library**。

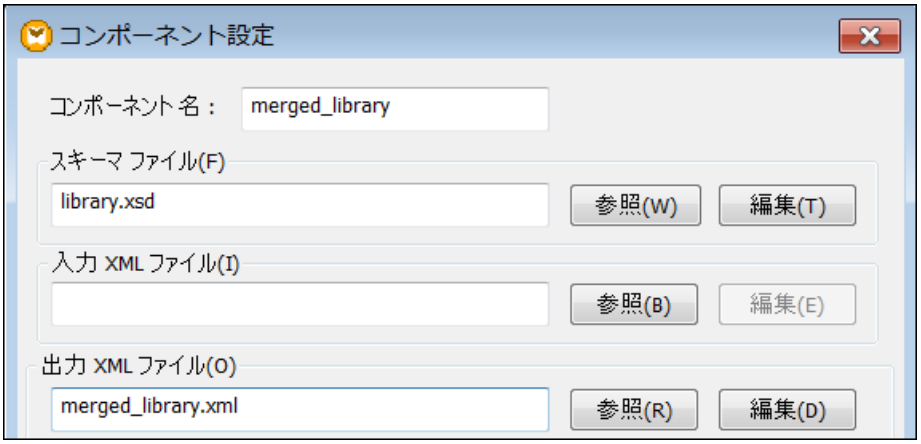
マッピングコンポーネントを4つの (左、右、上、下) 方向に移動することができます。しかしながら、ソースコンポーネントをターゲットコンポーネントの左に配置することにより、マッピングの作成を簡単にし、また、他のユーザーによる読み込みやすく理解しやすくなります。このドキュメントで説明されている全てのマッピングにもこの規則は適応されており、また、MapForce インストールに伴うサンプルマッピングファイルにも同様の規則が適用されています。

ステップ 3 : 入力 /出力ファイルの検証とセット

前のステップでは、新しいソースコンポーネントはターゲットコンポーネントからコピーし、貼り付けられましたので同じ設定を継承します。名前入力 /出力インスタンスファイルが正確に設定されるように、各コンポーネントのヘッダーをダブルクリックし、コンポーネント設定ダイアログボックス内で、下に表示されるように各コンポーネントの入力 /出力ファイルを検証し、名前を変更します。

*最初のソースのコンポーネント設定 (**books**)*

*2番目のソースのコンポーネント設定 (**library**)*

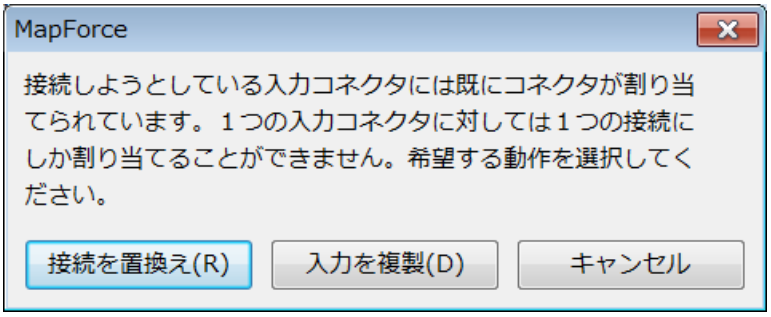


*ターゲットのコンポーネント設定 (**merged_library**)*

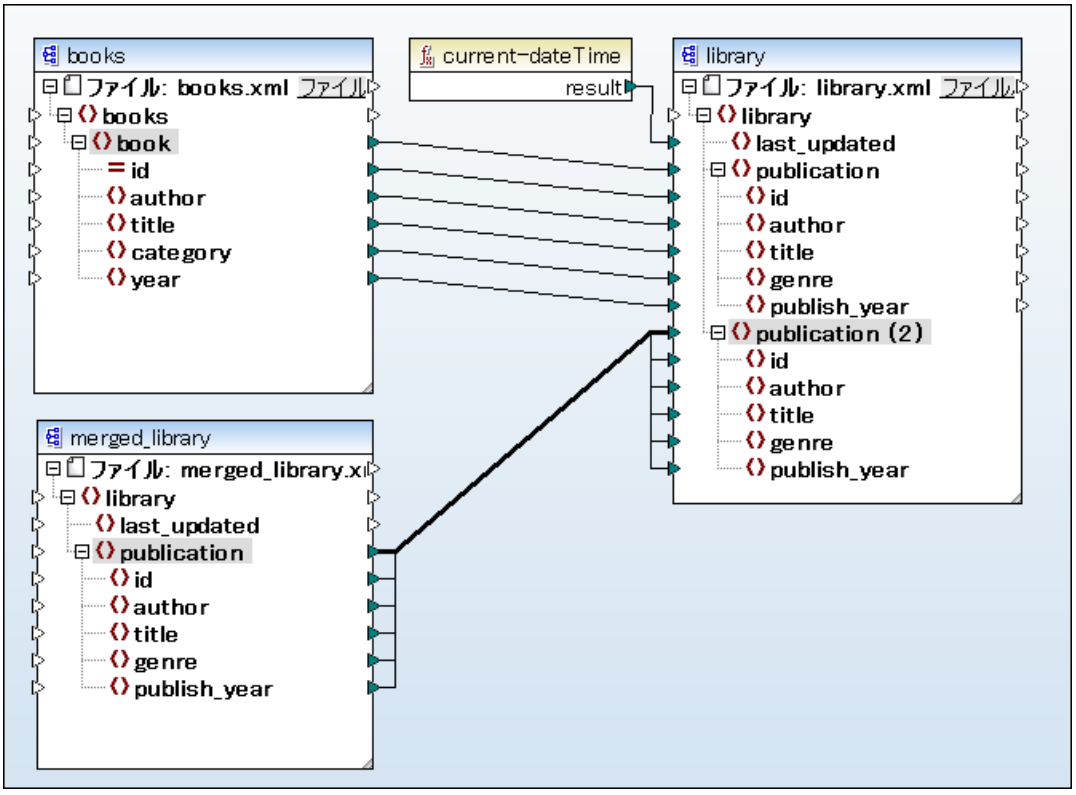
上に表示されるように、最初のソースコンポーネントは**books.xml** からデータを読み込みます。2番目のソースコンポーネントは**library.xml** からデータを読み込みます。2番目のソースコンポーネントは**merged_library.xml** というファイルのターゲットコンポーネント出力データを最後に読み込みます。

ステップ 4：接続の作成

MapForce に2番目のソースからターゲットにデータを書き込むように命令するには、ソース**library** 内コンポーネントの `publications` アイテムの出力コネクタ (小さな三角形) をクリックし、ターゲット**library** コンポーネント内のアイテムの入力コネクタにドラッグします。ターゲット入力コネクタには既に接続があり、次の通知メッセージが表示されます。



このチュートリアルでは、接続の置換えが目的ではなく、目的は2つのソースからのデータのマッピングです。ですから、**「入力の複製」**をクリックします。これを行うことにより、ターゲットコンポーネントに新しいソースからのデータを受け入れるように構成することができます。マッピングは、以下のようになります：



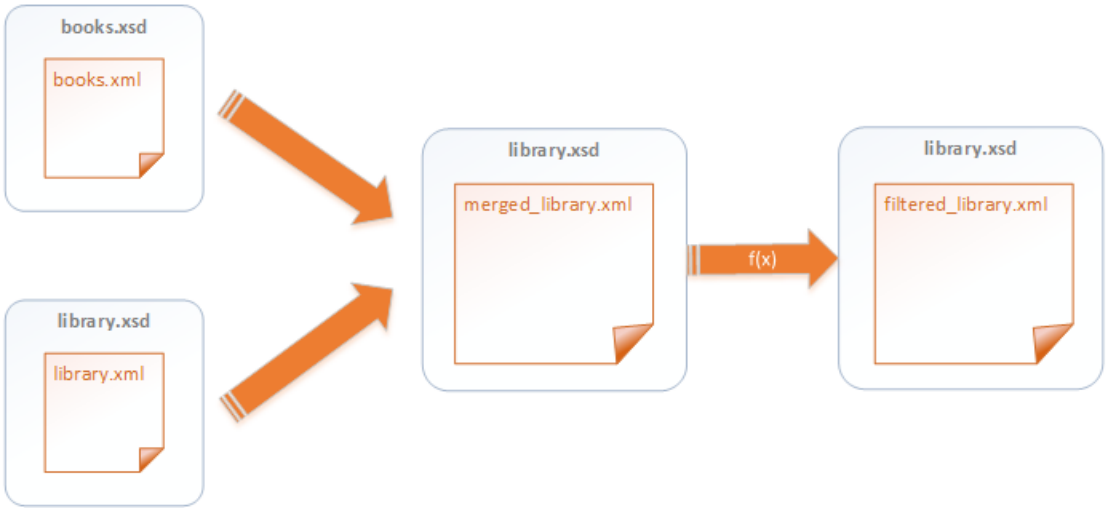
ターゲットコンポーネント内の`publication`アイテムが複製されました。新規`publication(2)`ノードはソース**library**コンポーネントからのデータを受け入れます。更に重要な点は、マッピング内でこのノードは`publication(2)`と表示されますが、結果XMLファイル内での名前は、想定された目的である`publication`となります。

マッピングペインの下の**「出力」**ボタンをクリックして、マッピングの結果を確認します。**library.xml**と**books.xml**ファイルのデータが、新規ファイル**merged_library.xml**にマージされたことを確認してください。

3.3 複数のターゲットスキーマとの作業

前のチュートリアルの、複数のソースから1つのターゲットにマップでは、複数のソーススキーマから単一のターゲットスキーマへのデータのマッピングについて作成方法について説明しました。2つのソースからのブックのレコードを保管する **merged_library.xml** と呼ばれるファイルを作成しました。他の部署からこのXML ファイルのサブセットを提供するように要請があったと仮定します。特に、1900年以降に出版された書籍を含むXML ファイルを提出するとします。

既存の**MultipleSourcesToOneTarget.mfd** マッピングを使用し、完全なXML ライブラリとフィルターされたライブラリ必要に応じて、を生成することができます。変更を加えることができます。



データ変換の抽象的なモデル

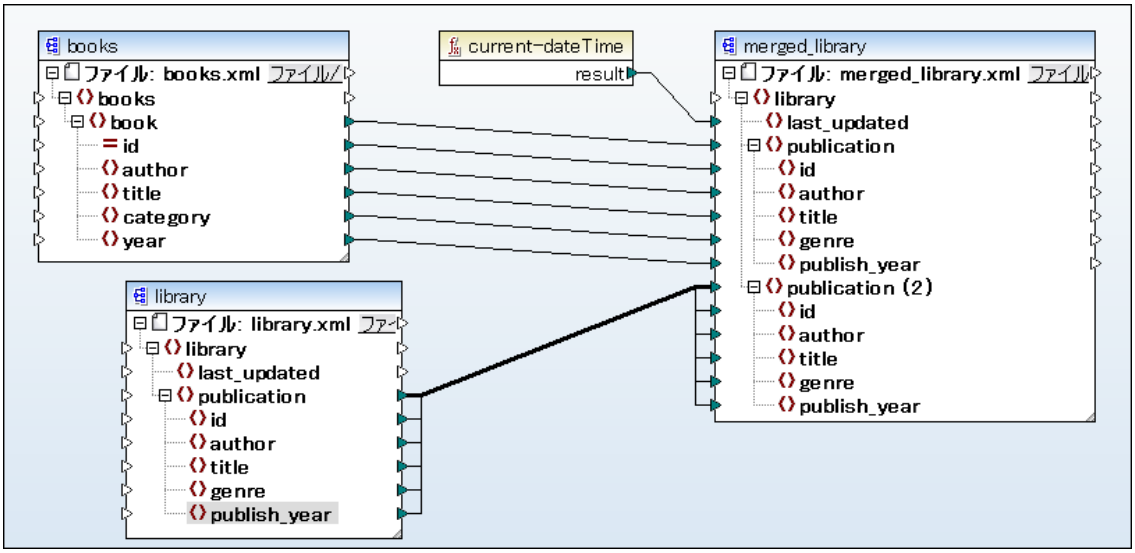
上の図では、データは2つの異なるスキーマ (**books.xsd** と**library.xsd**) が**merged_library.xml** と呼ばれる単一のXML ファイルにマージされます。第2に、データはフィルター関数を使用して変換され、**filtered_library.xml** と呼ばれるXML ファイルを作成する、次のコンポーネントにパスされます。「中間」コンポーネントは、データターゲットとソースとしての役割を果たします。MapForce では、このテクニックは「チェーンマッピング」と呼ばれ、このチュートリアルの主な内容です。ます。

このチュートリアルのゴールは、**merged_library.xml** と**filtered_library.xml** を必要に応じて生成することです。チュートリアルの目的を達成するためには、次のステップを踏みます。

ステップ 1：マッピングデザインファイルの準備

このチュートリアルは、**<Documents>\Altova\MapForce2017\MapForceExamples\Tutorial** フォルダーからの**MultipleSourcesToOneTarget.mfd** マッピングを開始点として使用します。このマッピングは複数のソースから1つのターゲットにマップチュートリアル内で作成済みです。チュートリアルを開始するには、MapForce で **MultipleSourcesToOneTarget.mfd** ファイルを開き、新しい名前で保存します。

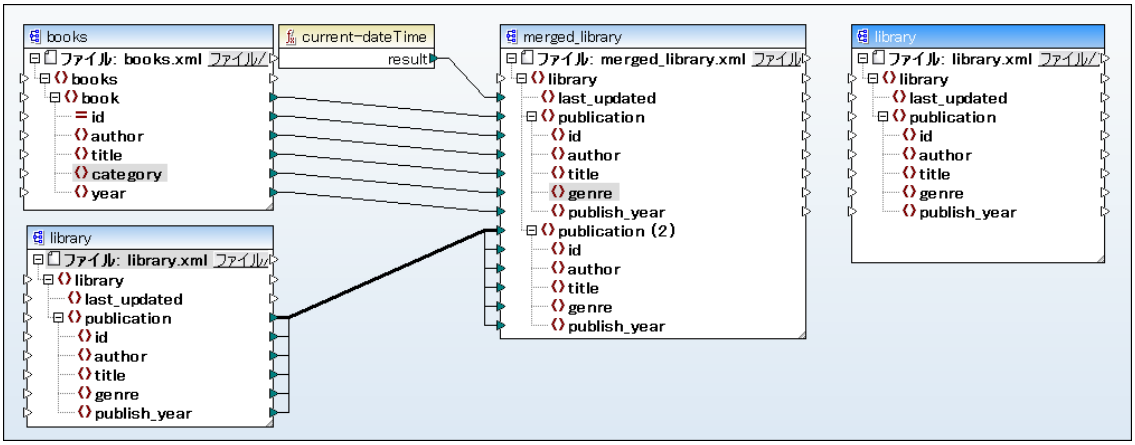
> 複数のファイルを参照するため、<Documents>\Altova\MapForce2017\MapForceExamples\Tutorial\ フォルダー内に新しいマッピングが保存されることを確認してください。



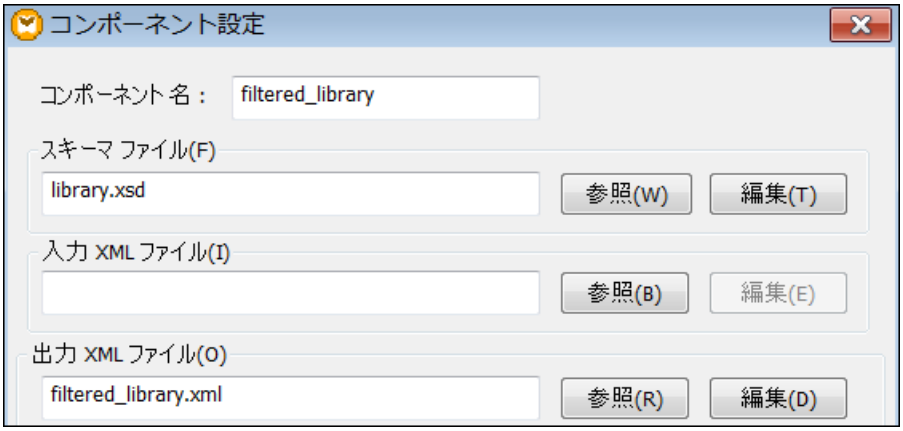
MultipleSourcesToOneTarget.mfd (MapForce Basic Edition)

ステップ 2：2番目のターゲットコンポーネントの追加と構成

2番目のターゲットコンポーネントを追加するには、**「XML スキーマ /ファイルの挿入」**（ ）ツールバーボタンをクリックし、**<Documents>\Altova\MapForce2017\MapForceExamples\Tutorial** フォルダーにある **library.xsd** ファイルを開きます。サンプルインスタンスファイルを提供するようにプロンプトされた場合、**「スキップ」**をクリックします。マッピングは、以下のようになります：



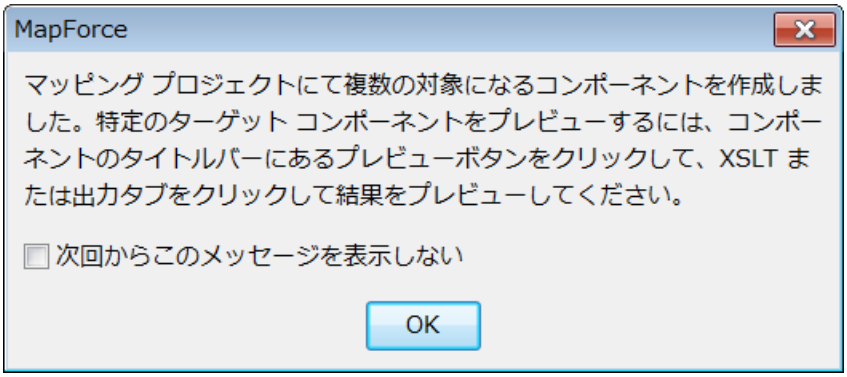
上に表示されるように、マッピングには2つのソースコンポーネントと**:books** と**library**、と 2つのターゲットコンポーネントがあります。ターゲットコンポーネントを区別するために、2番目のターゲットコンポーネントを**filtered_library** に名前を変更し、生成されるXML ファイルの名前も変更します。これを行うには、右端にあるコンポーネントをダブルクリックして、コンポーネントの設定を次のように編集します：



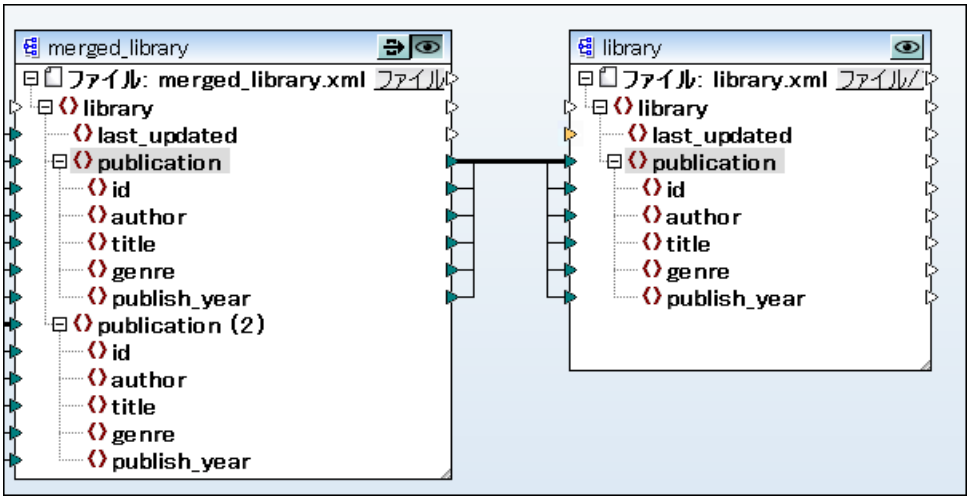
コンポーネントの新しい名前は**filtered_librar** となり、出力 XML ファイルには**filtered_library.xml** という名前が付けられます。

ステップ 3：接続の作成

merged_library 内のアイテム**publication** から**filtered_library** 内のアイテム**publication** への接続を作成します。これを行う際、通知メッセージが表示されます。

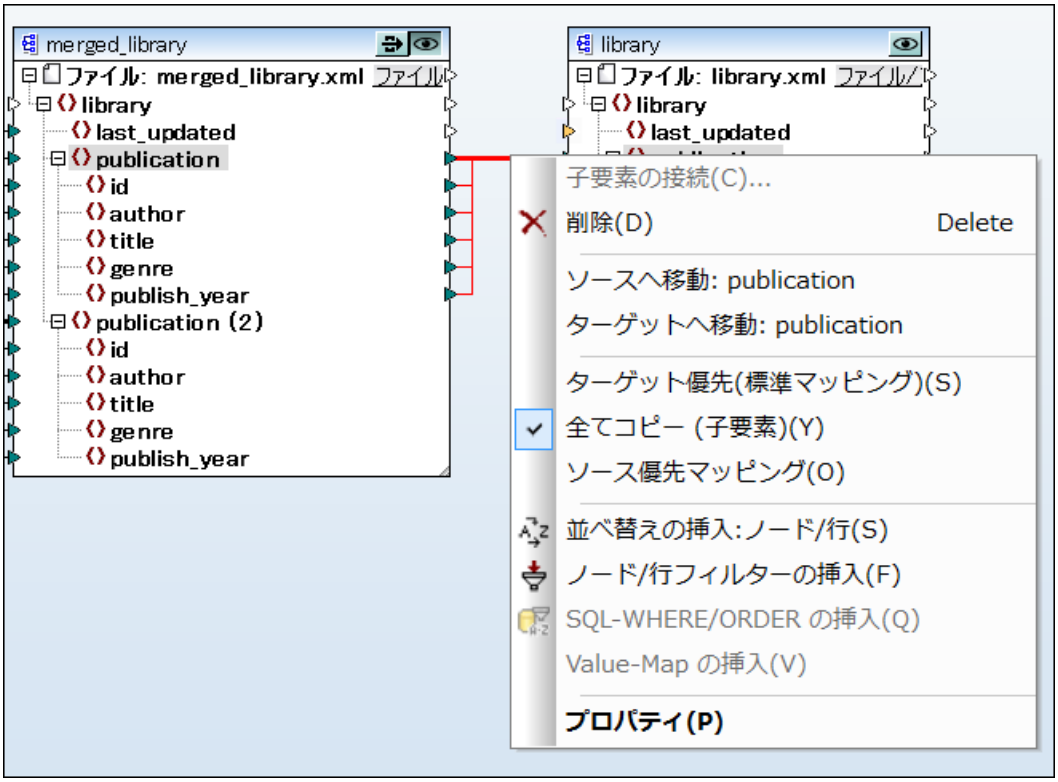


「OK」をクリックします。次の新規のボタンがターゲットコンポーネントの右上に表示されます：**「プレビュー」**（ ）と **「パススルー」**（ ）。次のステップでこれらのボタンは使用され、説明されます。

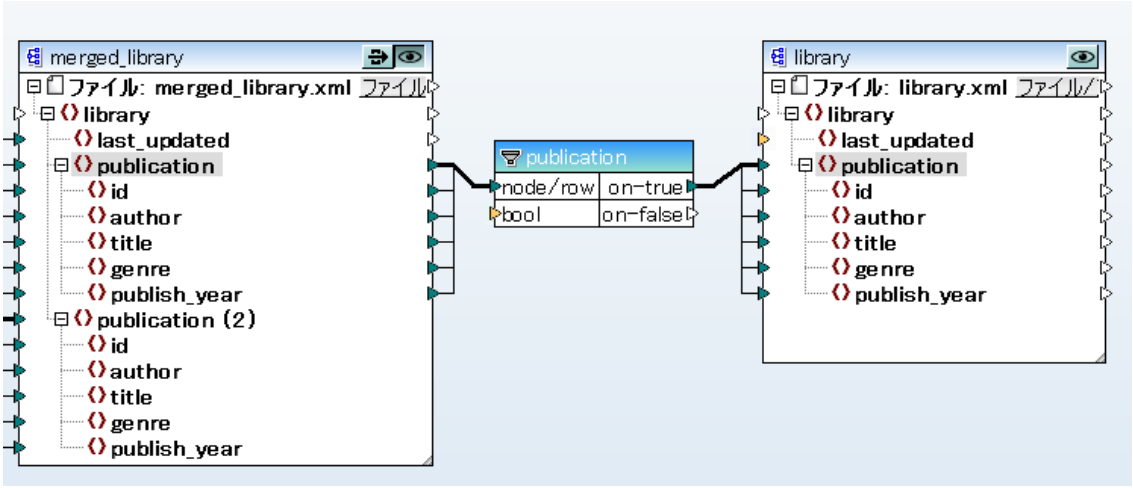


ステップ 4：データのフィルター

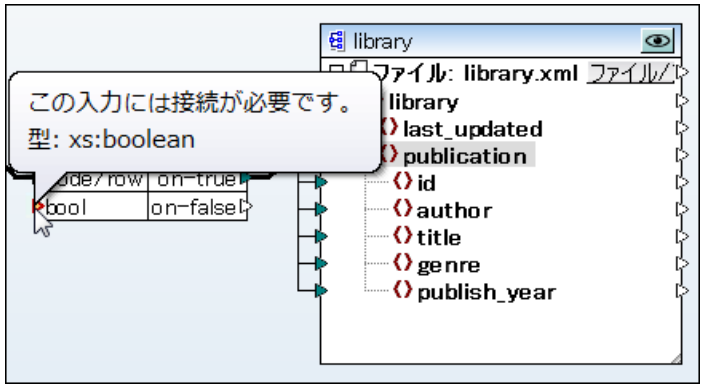
filtered_library に提供する前にデータをフィルターするには、**「フィルター」**コンポーネントを使用します。フィルターコンポーネントを追加するには、**merged_library** と**filtered_library**、間の接続を右クリックして、コンテキストメニューから、**「Node/row フィルターの挿入」**を選択します。



フィルターコンポーネントがマッピングに追加されました。



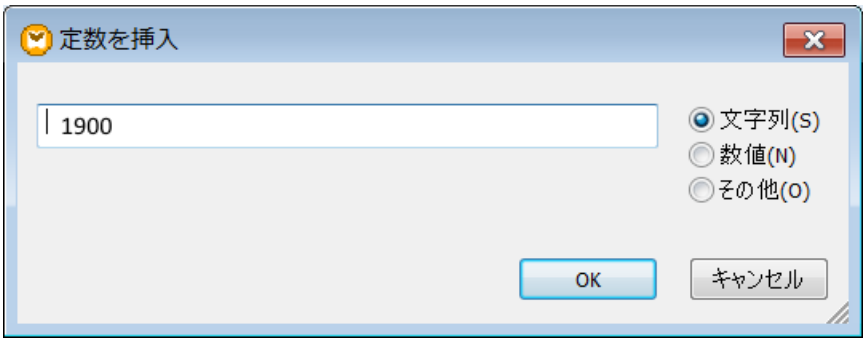
上に表示されるように、ブール入力コネクタは、入力が必要なことを示すオレンジ色にハイライトされます。コネクタ上にマウスをポイントすると、入力の型 `xs:boolean` が必要なことがわかります。ヒントを表示するには、**「ヒントの表示」**( )ツールバーボタンが有効化されている必要があることに注意してください。



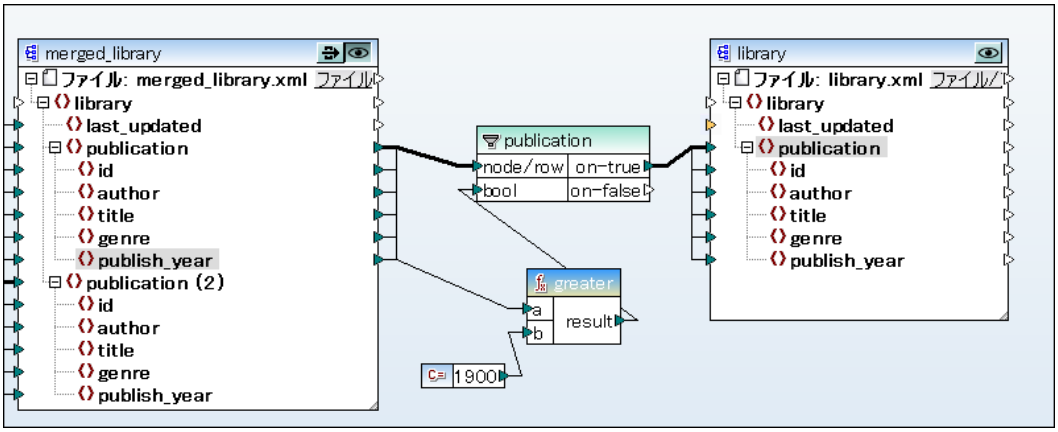
フィルターコンポーネントは`true` または`false` を返す条件を必要とします。ブール型の条件は、現在の**publication** シーケンスのデータがターゲットにコピーされると`true` を返します。条件が、`false` を返すと、データはコピーされません。

このチュートリアルで使用される条件は1900年以降に出版された書籍をフィルターします。条件を整えるために、以下を行います：

1. 1900 の値を持つ定数の数値の型を追加します。(**挿入**」メニューから**定数**」をクリックします)。**数値**」を型として選択します。



2. ライブラリウィンドウ内で、関数 **greater** を検索しマッピングペインにドラッグします。
3. **greater** 関数へと**greater** 関数から以下に表示されるように、マッピングを接続します。これを行うことにより、MapForce に以下を行うように命令します：「`publish_year` (出版年) が 1900以降の場合は、現在の`publication` ソースアイテムを`publication` ターゲットアイテム」にコピーします。



ステップ 5：各ターゲットコンポーネントの出力をプレビューし保存する

ターゲットコンポーネントの出力をプレビューし保存することができます。複数のターゲットコンポーネントが同じマッピングに存在する場合、 **「プレビュー」**（ ）ボタンをクリックすることにより、プレビューの対象を選択することができます。 **「プレビュー」**ボタンが押された状態（ ）の場合、特定のコンポーネントが現在プレビューのために有効化されていることを示します（そして、この特定のコンポーネントが「プレビュー」ペイン内に出力を生成します）。一度に1つのコンポーネントのみがプレビューのために有効化することができます。

ですから、**merged_library**（すなわち、「中間」）コンポーネントの出力を確認し保存する場合は、以下を行います：

1. **merged_library** コンポーネントの **「プレビュー」**ボタン（ ）をクリックします。
2. **「出力」**ボタンマッピングペインの下のをクリックします。
3. 出力をファイルに保存するには、 **「出力」** メニューから **「出力ファイルの保存」**をクリックします。

filtered_library コンポーネントの出力を表示して保存するには以下を行います：

1. **merged_lilbrary** コンポーネント上の **「パススルー」**ボタン（ ）をクリックします。
2. **filtered_library** コンポーネント上の **「プレビュー」**ボタン（ ）をクリックします。
3. マッピングペインの下の **「出力」** ボタンをクリックします。
4. 出力をファイルに保存するには、 **「出力」** メニューから **「出力ファイルを保存」**をクリックします。

「パススルー」（ ）ボタン? をクリックまたはクリックしないことにより、複数のターゲットコンポーネントを持つマッピングに大きな違いをもたらします。このボタンが押された状態（ ）である場合、MapForce は、データに中間コンポーネントをパススルーさせ、マッピング全体の残りをプレビューすることができます。

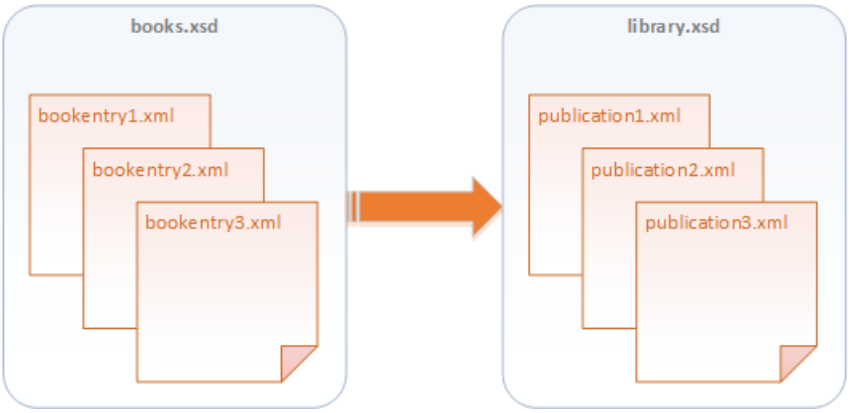
merged_library と**filtered_library** 間のマッピングの部分のみをプレビューするには、ボタン（ ）をリリースします。後者の場合、エラーが生成されます。中間コンポーネントはデータを読み込むべき妥当な入力 XML ファイルを持たないため、この振る舞いは予期されます。問題を解決するには、 コンポーネントのヘッダーをダブルクリックし、編集して、妥当な入力 XML ファイルを提供します。

複数のターゲット コンポーネントを持つマッピングのデザインが終わると、このチュートリアルの目的である、各ターゲットの出力を確認して保存することができます。パススルー コンポーネントに関しての詳細は、 チェーンマッピング / パススルーコンポーネントを参照してください。

3.4 ファイルを動的に処理と生成する

このチュートリアルは、複数のソースXML ファイルからデータを読み込み、同じ変換内で複数のターゲットファイルに書き込む方法を説明します。このテクニックを説明するために、次の目的を持つマッピングを作成します：

1. 同じディレクトリ内の複数のXML ファイルからのデータを読み込む。
2. 新規のXML スキーマに各ファイルを変換する。
3. 各ソースXML ファイルのために、新しいスキーマの下で新規ターゲットファイルを生成する。
4. 生成されたファイルからXML と名前空間宣言を削除する。



データ変換の抽象的なモデル

3つのソースXML ファイルをサンプルとして使用します。ファイルの場所は**<Documents>\Altova\MapForce2017\MapForceExamples\Tutorial** フォルダーで確認することができ**bookentry1.xml**、**bookentry2.xml**, と **bookentry3.xml** という名前が付けられています。これらのファイルはそれぞれ単一のブックを保管します。

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

bookentry1.xml

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

bookentry2.xml

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

bookentry3.xml

ソースXML ファイルは次のフォルダーで使用することのできる**books.xsd** スキーマを使用します :**<Documents>\Altova\MapForce2017\MapForceExamples\Tutorial**。ソースファイルを新規 XML スキーマに変換するには、同じフォルダーで見つけることのできる)**library.xsd** スキーマを使用します。変換の後、マッピングは3つのファイルを新しいスキーマに従い生成します。(下のコードリストを参照)。生成されたファイルが以下のようになるよう マッピングを構成します :**publication1.xml**、**publication2.xml**、と**publication3.xml**。XML 宣言と名前空間が削除されなければならない点に注意してください。

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

publication1.xml

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

publication2.xml

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

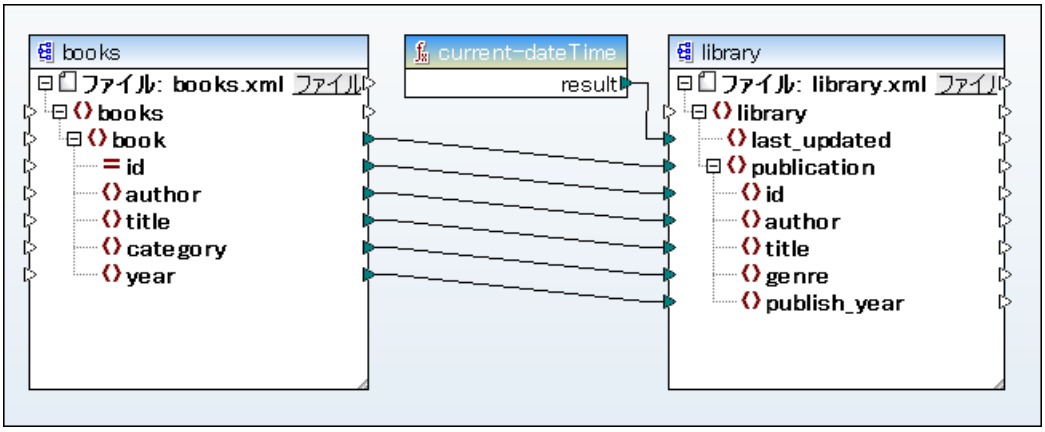
publication3.xml

チュートリアルの目的を達成するために、次のステップを踏みます。

ステップ 1：マッピングデザインファイルの準備

このチュートリアルは、開始のポイントとして**<Documents>\Altova\MapForce2017\MapForceExamples\Tutorial** フォルダーから**BooksToLibrary.mfd** マッピングを使用しています。すでにXML を新しいスキーマに変換するチュートリアル内でこのマッピングをデザインしています。開始するには、MapForce で**BooksToLibrary.mfd** ファイルを開き、新しい名前で同じフォルダーに保存します。

> 複数のファイルを参照するため、**<Documents>\Altova\MapForce2017\MapForceExamples\Tutorial** フォルダー内に新しいマッピングが保存されることを確認してください。



BooksToLibrary.mfd (MapForce Basic Edition)

ステップ 2：入力を構成する

MapForce に複数のXML インスタンスファイルを処理するように命令するには、ソースコンポーネントのヘッダーをダブルクリックします。コンポーネント設定ダイアログボックス内に入力ファイルとして**bookentry*.xml** を入力します。

コンポーネント設定 ダイアログボックス

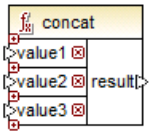
ファイル名内のアスタリスク（*）ワイルドカード文字は、マッピング入力に**bookentry-** プレフィックスを伴う全てのファイルを使用するように命令します。パスは相対的であるため、MapForce はマッピングファイルと同じディレクトリ内の全ての **bookentry-** ファイルを検索します。マッピングファイル .*ワイルドカード文字を使用しているため、必要であれば、絶対パスを入力することもできます。

ステップ 3：出力を構成する

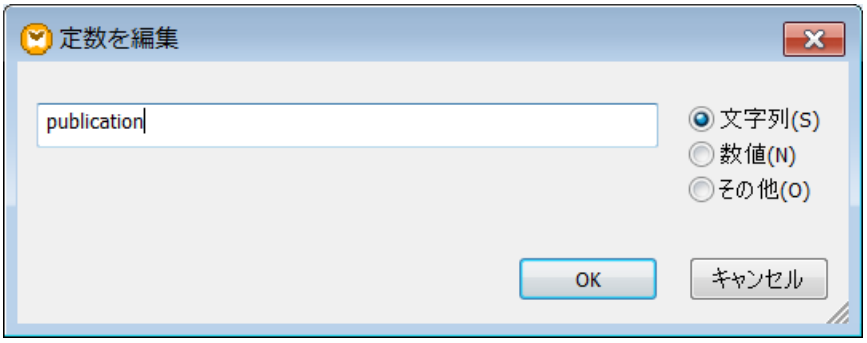
各出力ファイルのファイル名を作成するには、**concat** 関数 を使用します。この関数は引数に与えられた全ての値を連結します。

concat 関数を使用してファイル名を作成する：

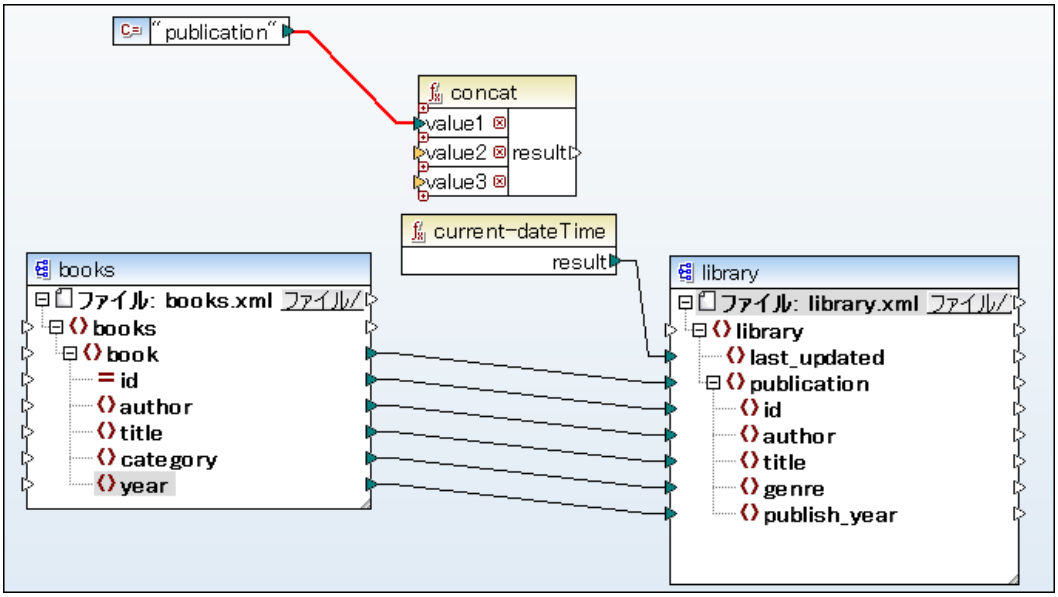
1. **concat** 関数 をライブラリウィンドウで検索し、マッピングエリアにドラッグします。デフォルトでは、この関数はパラメータが2つあるマッピングに追加されます。しかし、必要であれば新しいパラメーターを追加することができます。関数コンポーネント内の **「パラメーターの追加」**（⊡）シンボルをクリックして、3番目のパラメーターを追加します。**「パラメーターの削除」**（⊠）シンボルをクリックするとパラメーターは削除されることに注意してください。



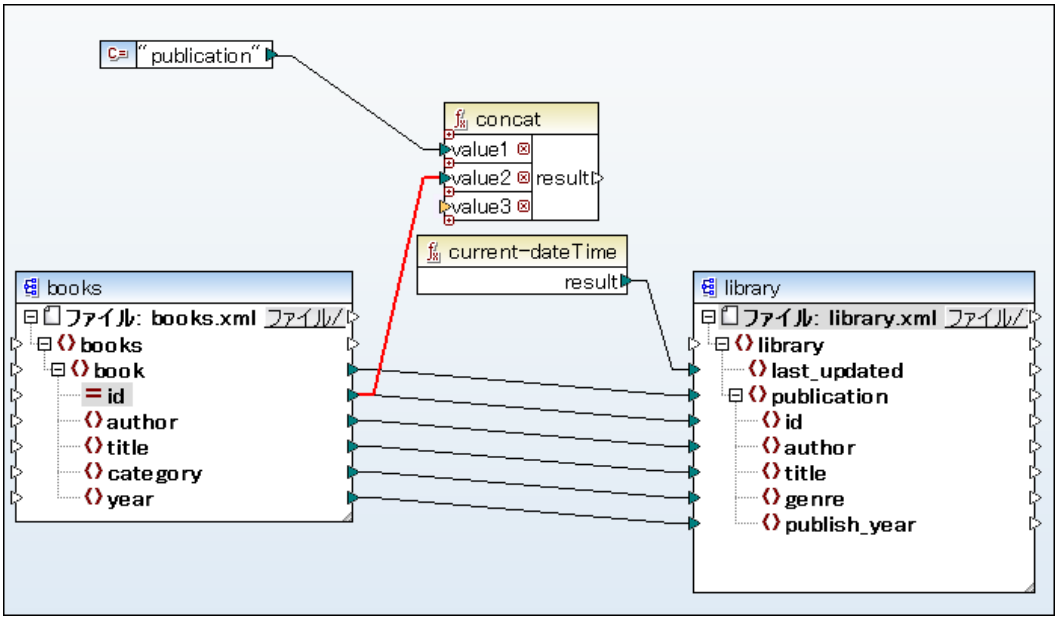
2. 定数を入力するには、（**挿入」** メニューから **定数」** をクリックします）。値を提供するようにプロンプトされると、"publication" を入力して、**文字列** オプションを変更しないでそのままにしておきます。



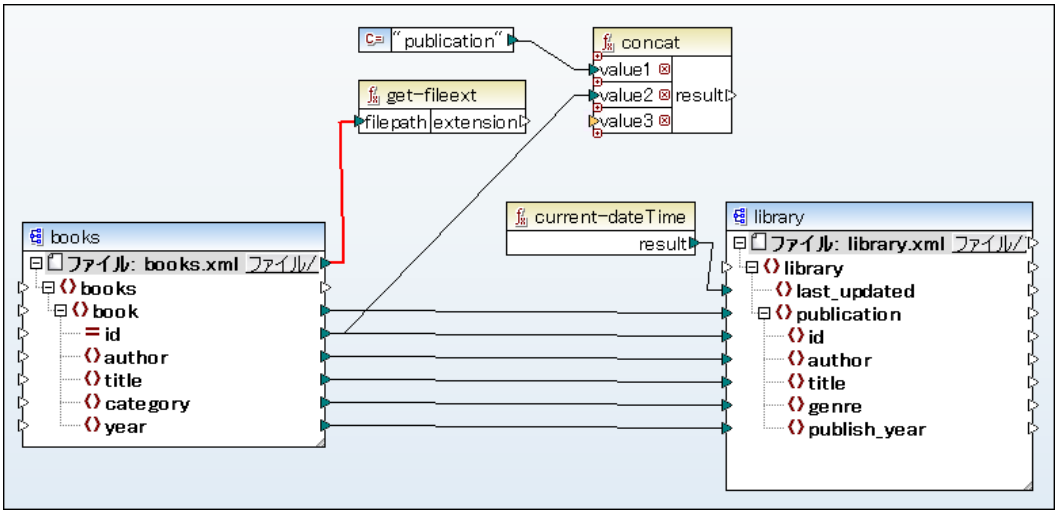
3. **concat** 関数の**value1** と定数をと接続します。



4. ソースコンポーネントの `id` 属性をconcat 関数の**value2** と接続します。



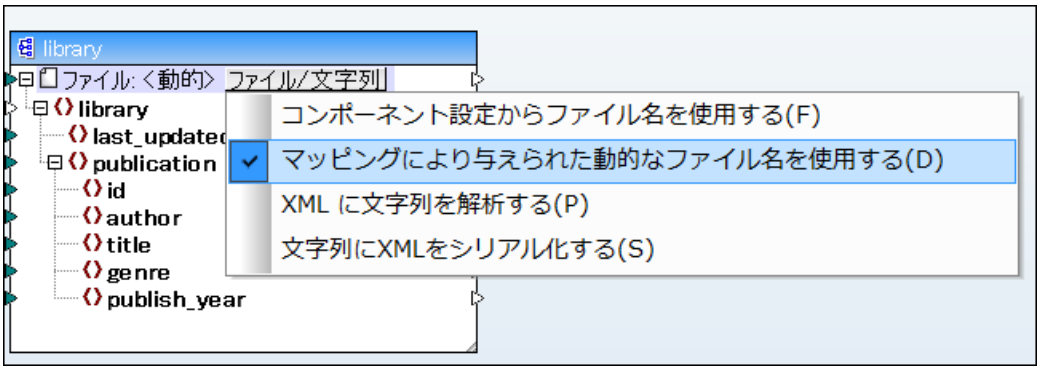
5. ライブラリウィンドウ内で`get-fileext` 関数 を検索し、マッピングエリアにドラッグします。ソースコンポーネント (**File: books.xml**) のトップノードからこの関数の**filepath** パラメータへ接続を作成します。`get-fileext` 関数の結果から`concat` 関数の**value3** への接続を作成します。これを行うことにより、ソースファイル名から拡張子の部分 (この場合、 xml) を抽出することができます。



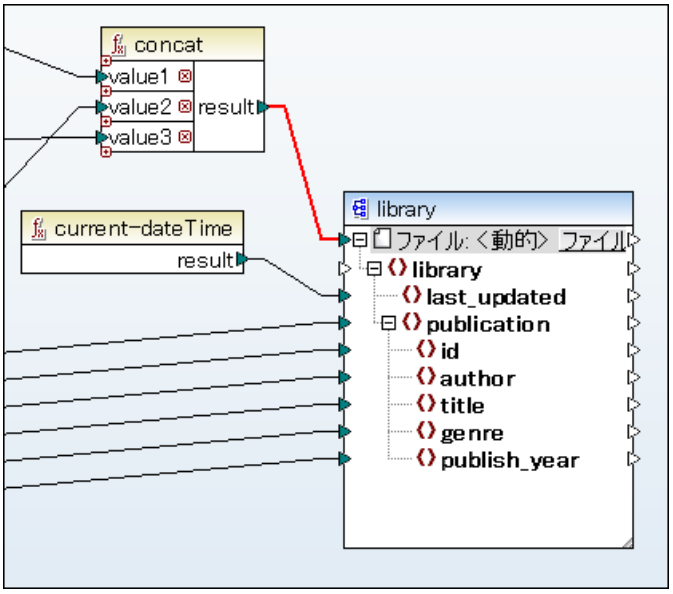
concat 関数のパラメーターに、連結されると生成されるファイル名（例：**publication1.xml**）を作成する3つの値を提供しました：

パート	サンプル
定数 "publication" は定数文字列の値 "publication" を与えます。	publication
ソースXML ファイルの属性 id は、各ファイルのための一意の識別しの値を与えます。これは全てのファイルが同じ名前で生成されないようにするためです。	1
get-fileext 関数 は、生成されるファイルの拡張子を返します。	.xml

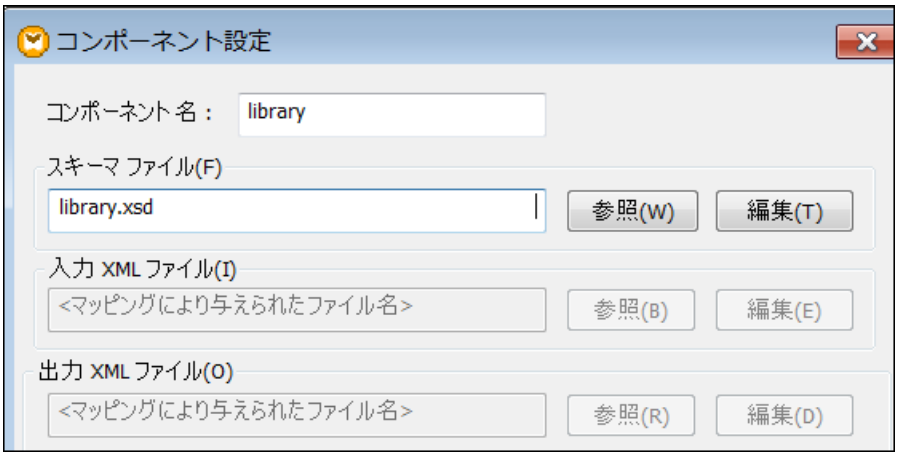
MapForce にマッピングが実行される際に実際にファイル名を作成するように命令することができます。これを行うには、**「ファイル」**（File）をクリックする、または ターゲットコンポーネントの **「ファイル文字列」**（File/String）ボタンから **「マッピングから与えられた動的なファイル名を使用する」** を選択します



マッピングにより与えられた名前を伴うMapForce にインスタンスファイルを動的に生成するように命令しました。この特別な例では、**concat** 関数により名前が作成されました。ですから、**concat** 関数の結果とターゲットコンポーネントの **File: <dynamic>** ノードを接続します。

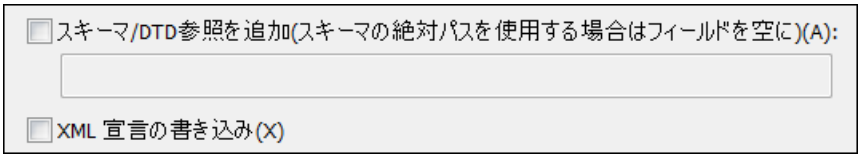


この時点でターゲットコンポーネントヘッダーをダブルクリックすると、**「入力 XML ファイル」**と**「出力 XML ファイル」**テキストボックスが無効化されており、値が**<マッピングにより与えられたファイル名>**を示していることに気がつくはずです。



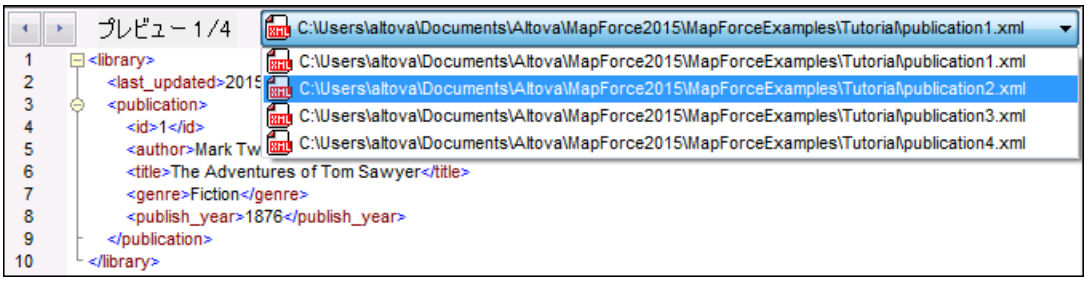
これは、インスタンスファイル名を動的にマッピングから提供したことを示し、コンポーネント設定内で定義することは関連性がありません。

最後に、ターゲットからXML 名前空間とスキーマ宣言を削除する必要があります。これを達成するには、コンポーネント設定ダイアログボックスの**「スキーマ /DTD レファレンスの追加 ..」**と**「XML 宣言の書き込み」**チェックボックスから選択を解除します。



結果と生成されたファイル名を確認するために、マッピングを実行します。 このマッピングは複数の出力ファイルを生成します。出力ペインの左上の左と右ボタンを使用して出力ファイル全体を移動することができます。 または横にあるドロップダウ

リストからファイルを選択します。



Chapter 4

一般的なタスク

4 一般的なタスク

このセクションは、マッピングとの作業、コンポーネント、接続などのMapForce の全般的なタスクとコンセプトを説明します。

4.1 マッピングとの作業

MapForce マッピングデザイン (または、略して "マッピング") は、データがどのように1つのフォーマットから他のフォーマットに変換されるかの視覚的な表現です。マッピングは、データ変換を行うためのMapForce マッピングエリア内で1つのスキーマから他のスキーマに追加するコンポーネントから構成されています。 (例 :XML ドキュメントを1つのスキーマから他のスキーマに変換)。有効なマッピングは、1つまたは複数のターゲットコンポーネントに接続されている、1つまたは複数の複数のソースコンポーネントから構成されています。マッピングを実行して、結果を直接 MapForce 内で確認することができます。コードを生成し、外部で実行することも可能です。MapForce 実行可能ファイルにマッピングをコンパイルし、MapForce Server またはFlowForce Server を使用してマッピングの実行を自動化することもできます。 MapForce は、マッピングを .mfd の拡張子を持つファイルとして保存します。

新規マッピングの作成方法 :

1. 以下を行います :
   - o **「ファイル」** メニューから **新規作成」** をクリックします。
   - o **新規作成」** ( ) ツールバーボタンをクリックします。

マッピングが作成されました。しかしながら、空のため何も起こりません。最低でも妥当なマッピングは2つの接続されたコンポーネント を必要とします。ですから、次のステップではマッピングにコンポーネントを追加し (マッピングにコンポーネントを追加する を参照) 2つのコンポーネントの間に接続を描きます。(接続との作業を参照)。

4.1.1 マッピングにコンポーネントを追加する

MapForce 内では、 "コンポーネント" という用語はデータの構造 (スキーマ) またはデータがどのように変換 (関数) されるかを視覚的に表します。コンポーネントはすべてのマッピングを構築する中心的な役割を果たします。以下は、MapForce コンポーネントの例です :

- 定数
- フィルター
- 条件
- 関数コンポーネント
- EDI ドキュメント (UN/EDIFACT、ANSI X12、HL7)
- Excel 2007+ ファイル
- 単純型 入力コンポーネント
- 単純型 出力コンポーネント
- スキーマおよびDTD

マッピングにコンポーネントを追加するには以下を行います :

- **挿入」** メニューから追加するコンポーネントの型 (例 : **「XML スキーマ/ファイル」** をクリックします。
- Windows ファイルエクスプローラーからファイルをドラッグしてマッピングエリアにドロップします。このオペレーションは互換性のあるファイルベースのコンポーネントのみに対して使用することができます。
- コンポーネントの挿入 ツールバー内の適切なボタンをクリックします。

コンポーネントの挿入 ツールバー (MapForce Enterprise Edition)

それぞれのコンポーネントの型には、特定の目的と振る舞いがあります。コンポーネントの型が必要な箇所では、MapForce は、コンテキストウィザードステップまたはダイアログボックスで手順を踏んで、説明を行います。例 :XML スキーマを追加する場合、通知ダイアログボックスがインスタンスファイルをオプションで選択するように促します。

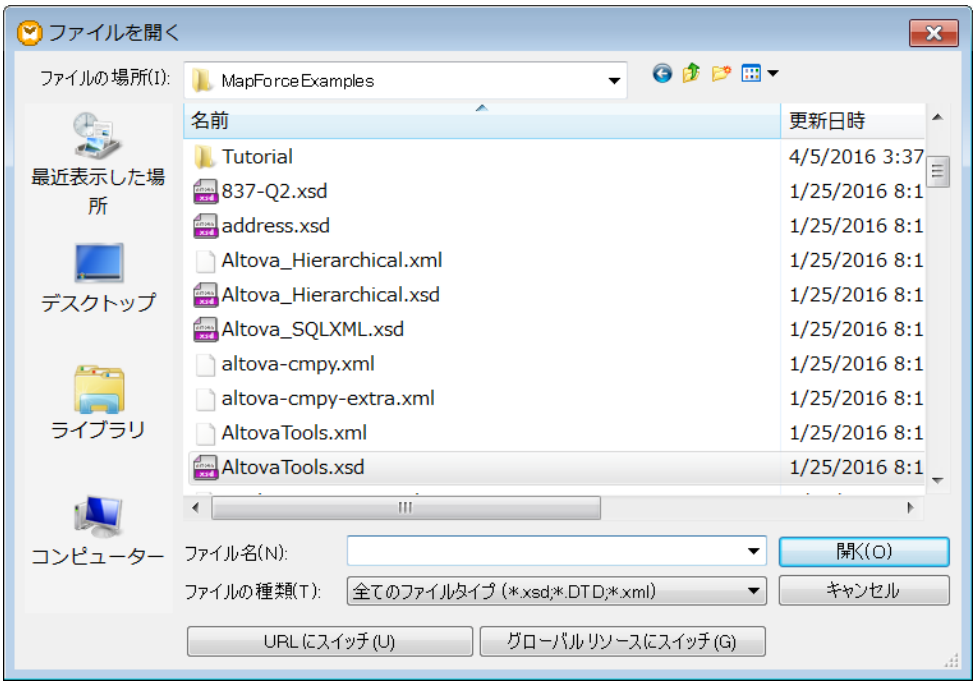
コンポーネントについての説明は、 コンポーネントとの作業を参照してください。マッピングソースまたはターゲットとしてサポートされる個々の技術に関しては、データソースとターゲットを参照してください。データを一時的に保管または変換する MapForce ビルドインコンポーネントに関しては、または (フィルタリングまたは並べ替えなどの) マッピングのデザインを参照してください。

4.1.2 URL からコンポーネントを追加する

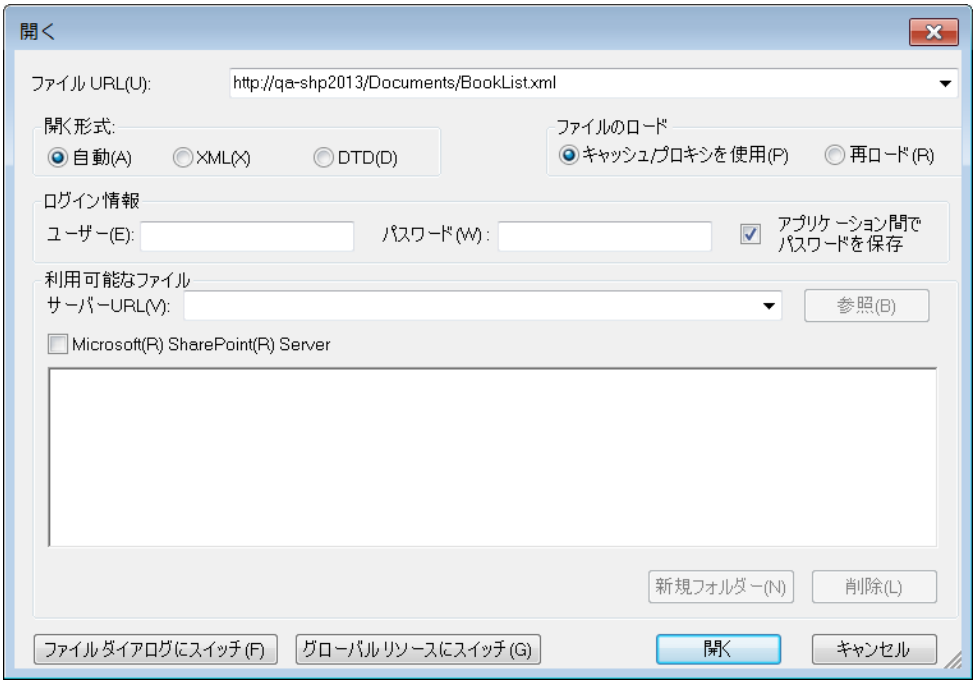
マッピングコンポーネントとしてローカルファイルを追加するだけでなく、URL からファイルを追加することができます。この操作はコンポーネントをソースコンポーネントとして追加した場合のみサポートされます。 (すなわち、マッピングは、リモートファイルからデータを読み込みます)。

URL からコンポーネントを追加する:

1. **挿入**」メニューから追加するコンポーネントの型を選択します (例 :**XML スキーマ/ファイル**)。
2. **開く**」ダイアログボックスから「**URL にスイッチ**」 をクリックします。



3. 「**ファイル URL**」テキストボックスにファイルの URL を入力して、 **開く**」をクリックします。



「ファイルURL」テキストボックス内のファイル型がステップ 1 で指定されたファイル型と同じであることを確認してください。

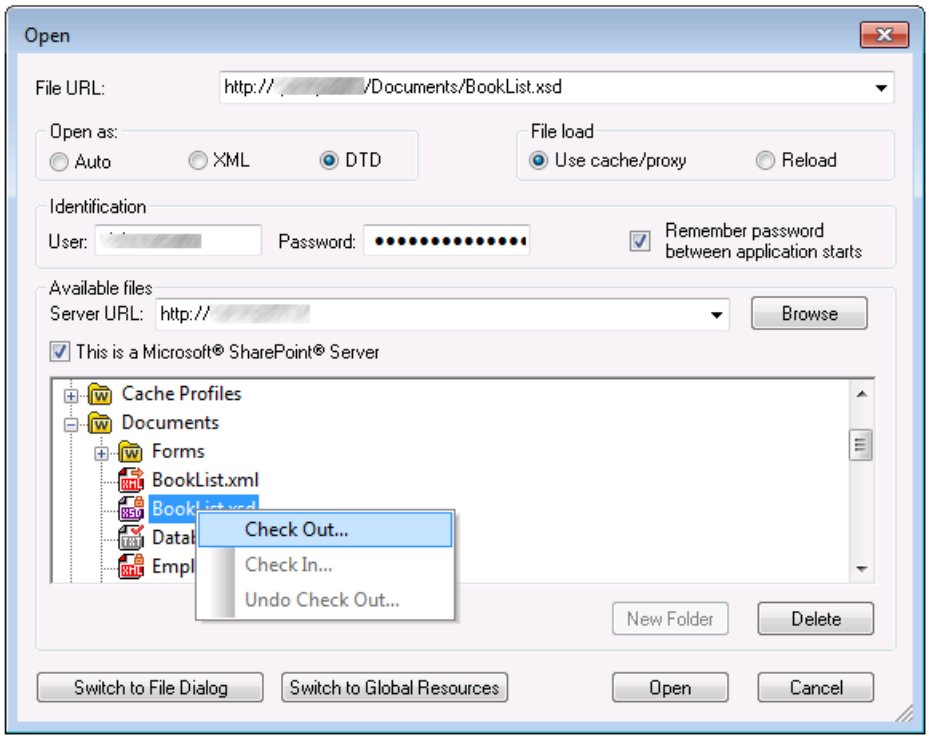
サーバーがパスワード認証を必要とする場合、ユーザー名とパスワードを入力するようにプロンプトされます。次回 MapForce を起動する際に使用できるように、ユーザー名とパスワードを保存するには、「開く」ダイアログボックス内の **「アプリケーション間でパスワードを保存」**チェックボックスを選択します。

「開く形式」設定は、ファイルを開く際のパーサーのための文法を定義します。デフォルトと奨励されるオプションは **「自動」**です。

アップロードするファイルに変更が加えられない場合、**「キャッシュ/プロキシの使用」**オプションを選択して、データをキャッシュし、ロードをスピードアップします。それ以外の場合、マッピングを開くたびにファイルを再ロードする場合は、**「再ロード」**を選択します。

Web Distributed Authoring and Versioning (WebDAV) をサポートするサーバーに関しては、**「サーバーURL」**テキストボックスにサーバーURL を入力した後にファイルを **「参照」**をクリックして、参照することができます。参照は全てのファイルの型を表示しますが、上記のステップ 1 で指定されたファイル型と同じ型が選択されていることを確認してください。それ以外の場合、エラーが発生します。

サーバーがMicrosoft SharePoint Server の場合、**「これは Microsoft SharePoint Server です」**チェックボックスをチェックしてください。これを行うことにより、プレビューエリア内のファイルのチェックインとチェックアウトの状態が表示されます。MapForce を使用中に他のユーザーがファイルを編集できないようにするためには、ファイルを右クリックして、**「チェックアウト」**を選択します。ユーザーにより以前チェックアウトされたファイルをチェックするには、ファイルを右クリックして、**「チェックイン」**を選択します。



開くダイアログボックス (URL に切り替えるモード)

4.1.3 変換言語の選択

データ変換言語として次を選択することができます :

- XSLT 1.0
- XSLT 2.0

変換言語を選択するには、以下を行います :

- **出力」** メニューから変換に使用する言語をクリックします。
- 言語選択ツールバー内の言語名 をクリックします。

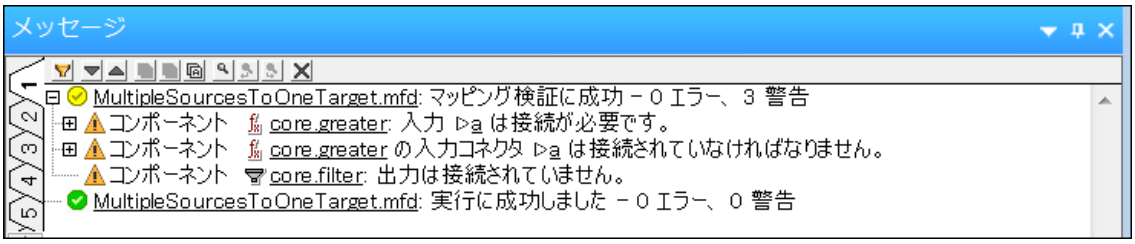
4.1.4 マッピングの検証

MapForce は変換結果をプレビューするために **出力」** タブをクリックすると、マッピングを自動的に検証します。結果をプレビューする前にマッピングのみを検証することもできます。この機能は、マッピングのエラーと警告をマッピングを実行する前に認識し修正するために役にたちます。マッピングの実行は、処理されたデータにより更なるランタイムエラーまたは警告を生成する場合があります。 例 属性にマップされた値の上書きなど。

マッピングを正確に検証するには、以下を行います :

- **「ファイル」**メニューから**「マッピングの検証」**をクリックします。
- **検証」**( ) ツールバーボタンをクリックします。

「メッセージ」ウィンドウは、検証の結果を表示します。例：



メッセージウィンドウ

マッピングを検証するには、MapForce は、マッピングの妥当性をチェックし、（無効または見つからない接続、サポートされないコンポーネントの種類などの）検証結果は、以下のステータスアイコンと共にメッセージウィンドウに表示されます：

アイコン	意味
	変換は成功しました。
	変換は警告と伴い完了しました。
	変換は失敗しました。

「メッセージ」ウィンドウは以下のメッセージの型を表示する場合があります：情報メッセージ、警告、とエラー。

アイコン	意味
	情報メッセージを表示します。情報メッセージはマッピングの実行を停止しません。
	警告メッセージを表示します。警告はマッピングの実行を停止しません。警告メッセージは以下の場合に生成される可能性があります。例：必須の入力コネクタへの接続が作成されなかった場合。このような場合、出力は、有効な接続が存在する場所で、これらのコンポーネントのために生成されます。
	エラーを表示します。エラーが発生すると、マッピングの実行に失敗すると、出力は生成されません。XSLT または XQuery コードのプレビューを行うこともできません。

マッピングエリア内の情報、警告または、エラーメッセージをトリガーしたコンポーネントまたは構造をハイライトするには、メッセージウィンドウ内の下線のついたテキストをクリックします。

データを変換するコンポーネントに関しては、（関数または変数など）MapForce 検証は以下のように動作します：

- 必須の**入力コネクタ**が接続されていない場合、エラーメッセージが生成され、変換は停止されます。
- **出力コネクタ**が接続されていない場合、警告が生成され、変換プロセスは続行されます。問題のあるコンポーネントとそのデータは無視され、ターゲットドキュメントにマップされません。

個別のタブ内にそれぞれの検証の結果を表示するには、メッセージウィンドウの左横にある番号の付いたタブをクリックします。これはとても役にたちます。例：複数のマッピングファイルと同時に作業する場合。

メッセージウィンドウ内の他のボタンにより以下のアクションを実行することができます：

- メッセージを型別にフィルターします（例：エラーまたは警告のみを表示）。

- エントリを上下に移動します。
- メッセージテキストをクリップボードにコピーします
- ウィンドウ内で特定のテキストを検索します
- メッセージウィンドウをクリアします

メッセージウィンドウの全般的な情報に関しては、ユーザーインターフェイスの概要を参照してください。

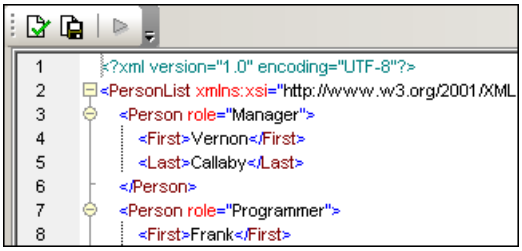
4.1.5 マッピング出力の検証

「出力」 タブをクリックした後、マッピングをプレビューする場合は、「出力」ペインで出力の結果を確認することができます。この出力を関連するスキーマに対して検証することができます。例：マッピング変換がXML ファイルを生成する場合、結果XML ドキュメントはXML スキーマに対して検証されます。

XML ファイルに関しては、コンポーネント設定ダイアログボックスの **「スキーマorDTD レファレンスの追加」** フィールド内 のインスタンスファイルに関連するスキーマを指定することができます(XML コンポーネント設定 を参照)。生成されたXML 出力により参照されるスキーマファイルをパスは指定します。これにより、マッピングの実行時、出力インスタンスが検証されることができます。このフィールドにhttp:// 、絶対または相対パスを入力することができます。 **「スキーマorDTD レファレンスの追加」** フィールドを選択しない場合、スキーマに対する、出力ファイルの検証は不可能です。このチェックボックスを選択し、空白のままにしておくと、検証が終わると、コンポーネント設定ダイアログボックスのスキーマファイル名が出力に生成されます。

コンポーネント設定を開くには、以下を行います：

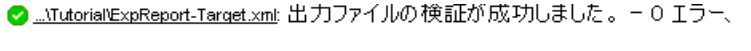
- **「出力の検証」** ツールバーボタンをクリックします。



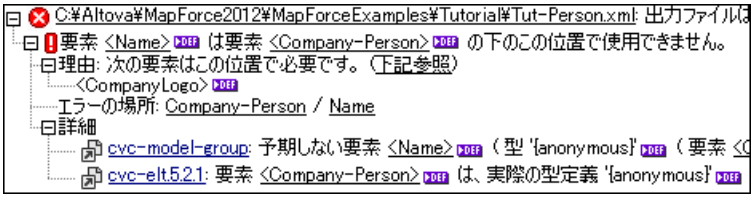
- **「出力」** メニューから **「出力ファイルの検証」** をクリックします。

メモ： **「出力の検証」** ボタンと対応するメニューコマンド（**「出力 | 出力ファイルの検証」**）は、出力ファイルがスキーマに対する検証をサポートする場合のみ有効化されます。

検証の結果はメッセージウィンドウ内に表示されます。例：



検証に失敗した場合、メッセージは発生したエラーの詳細情報を表示します。



検証メッセージは更に詳しい詳細情報を含む多くのハイパーリンクを含みます。:

- ファイルパスをクリックすると、MapForce の **出力** タブの結果の出力が開かれます。
- <ElementName> リンクをクリックすると、 **出力** タブ内の要素がハイライトされます。
- PDF アイコンをクリックすると、XMLSpy (インストールされている場合) 内の要素の定義が開かれます。
- 詳細サブセクションのハイパーリンク例 :cvc-model-group) をクリックすると、http://www.w3.org/ Web サイト上の対応する検証のルールについての詳細が開かれます。

4.1.6 出力のプレビュー

MapForce マッピングと作業する場合、結果出力を、外部のプロセッサーまたはコンパイラーを使用して生成されたコードを実行またはコンパイルすることなく、プレビューすることができます。全般的には、生成されたコードを外部で処理する前に、MapForce 内で変換出力をプレビューすることはよい考えです。

マッピングの結果をプレビューする場合、MapForce は、マッピングを実行して、結果出力を出力ペインに表示します。

出力ペインにデータが準備されると、必要に応じて検証し保存します (マッピング出力の検証を参照)。また、 **検索** コマンドを使用して、 (**Ctrl + F** キーの組み合わせ) 簡単に特定のテキストのパターンを出力ファイル内で検索することができます。

マッピングの実行に関するエラー、警告、情報メッセージはメッセージウィンドウに表示されます (ユーザーインターフェイスの概要を参照)。

変換の出力をプレビューする方法 :

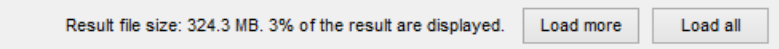
- マッピングウィンドウ下の **出力** タブをクリックします。MapForce は、言語ツールバー内で選択された変換言語を使用してマッピングを実行し、結果の出力を使用して「出力」ペインを作成します。

変換の出力を保存するには、以下を行います :

- **出力** メニューから **出力ファイルの保存** をクリックします。
- **生成された出力の保存** ツールバーボタンをクリックします。

部分的な出力のプレビュー

大きな出力ファイルをプレビューする場合、MapForce は、出力ペインに表示されるデータの量を制限します。具体的には、MapForce は、出力ペイン内でファイルの一部を表示します。そして、 **更にロード ..** ボタンがペインの下の部分に表示されます。 **更にロード ..** ボタンは現在表示されているデータのファイルの部分の横に表示されます。



メモ : 出力ペイン内にファイル全体がロードされると、 **整形出力** ボタンが有効化されます。

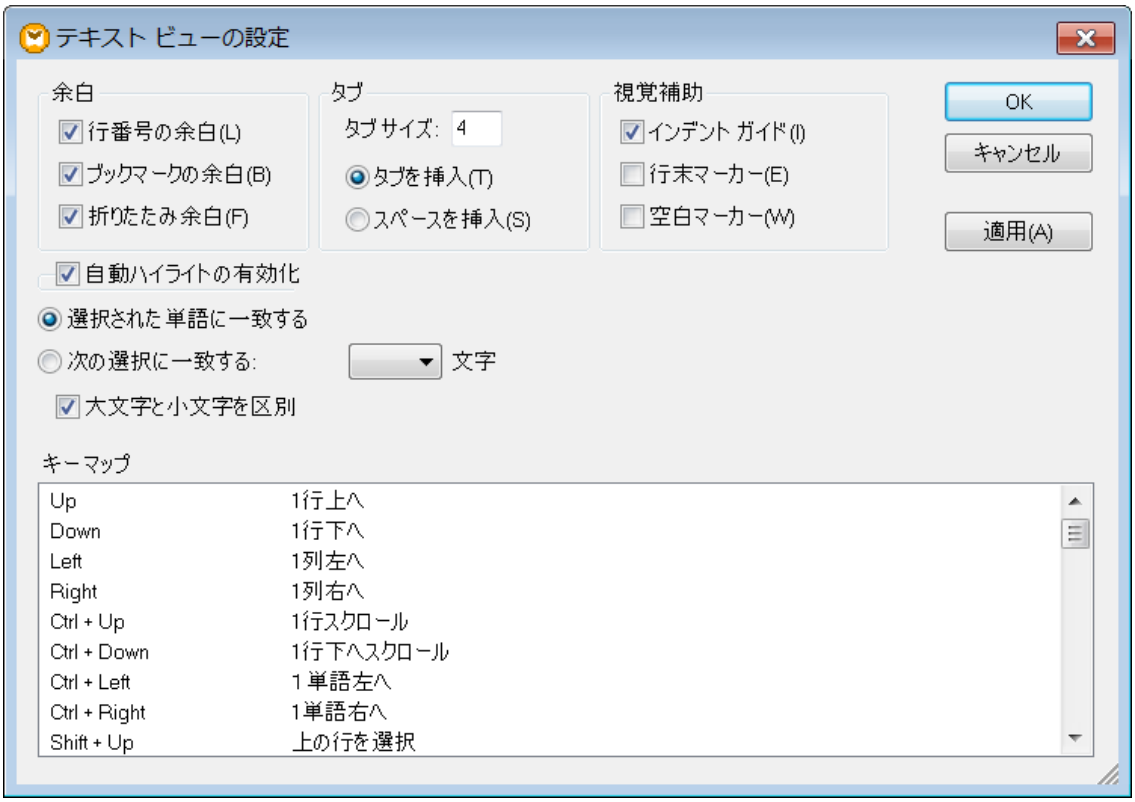
オプション ダイアログボックスの **全般** タブからプレビューの設定を構成することができます (MapForce オプションの変更を参照)。

4.1.7 テキストビューの機能

出力 ペインと **XSLT** ペインには、テキストをより簡単に表示するための複数の視覚的な補助が搭載されています。以下のような機能が含まれます :

- 行番号
- 構文の色
- ブックマーク
- ソースの折りたたみ
- インデントのガイド
- 行末と空白文字のマーカー
- ズーム
- 整形出力
- ワードラップ
- テキストのハイライト

適用できる箇所では、上記の機能を**テキストビュー設定** ダイアログボックスから切り替え、または、カスタム化することができます。**テキストビュー設定** ダイアログボックス内の設定は、アクティブなドキュメントだけではなく、アプリケーション全体に適用されます。



テキストビュー設定 ダイアログボックス

テキストビュー設定 ダイアログボックスを開くには、以下を行います：

- **出力** メニューから**テキストビュー設定**を選択します。
- **テキストビュー設定** ツールバーボタンをクリックします。
- 出力 ペインを右クリックして、コンテキストメニューから、**テキストビュー設定**を選択します。

ナビゲーションの補助の一部は、テキストビューツールバー、アプリケーションメニュー、キーボードショートカットから切り替えることができます。

テキストビューツールバー

適用することのできるショートカットへの参照は、上記の**テキストビュー設定**ダイアログボックスの「キーマップ」セクションを確認してください。

行番号

行番号は、**テキストビュー設定**ダイアログボックス内で切り替えることのできる行番号マージンで表示されます。テキストのセクションが折りたたまれると、折りたたまれたテキストの行番号は表示されません。

構文の色

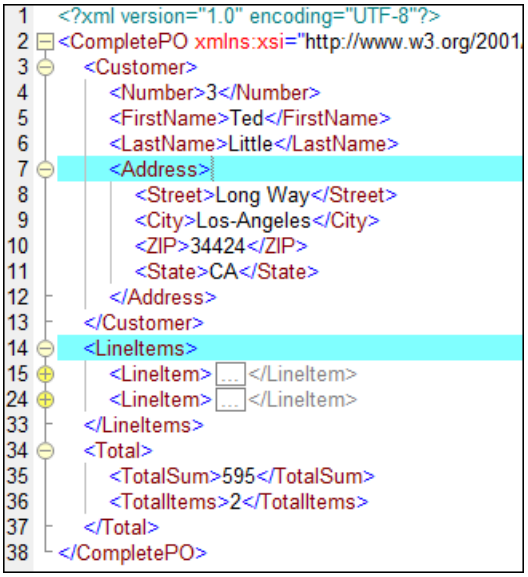
構文の色分けはテキストの構文の値に従い適用されます。例えば、XML ドキュメント内では、XML ノードが要素、属性、コンテンツ、CDATA セクション、コメント、処理命令、ノード名（そして一部の場合ではノードのコンテンツ）により異なる色分けが使用されています。

ブックマーク

ドキュメント内のラインは参照とアクセスのためにブックマークとして使用することができます。ブックマークマージンオンに設定されていると、ブックマークは、ブックマークマージンに表示されます。

```
1   <?xml version="1.0" encoding="UTF-8"?>
2   <CompletePO xmlns:xsi="http://www.w3.org/2001
3     <Customer>
4       <Number>3</Number>
5       <FirstName>Ted</FirstName>
6       <LastName>Little</LastName>
7       <Address>
8         <Street>Long Way</Street>
9         <City>Los-Angeles</City>
10        <ZIP>34424</ZIP>
11        <State>CA</State>
12      </Address>
13    </Customer>
14    <LineItems>
15      <LineItem> ... </LineItem>
24      <LineItem> ... </LineItem>
33    </LineItems>
34    <Total>
35      <TotalSum>595</TotalSum>
36      <TotalItems>2</TotalItems>
37    </Total>
38  </CompletePO>
```

それ以外の場合、ブックマークされたラインは水色でハイライトされています。



ブックマークマージンは、**テキストビュー設定** ダイアログボックス内で切り替えることができます。

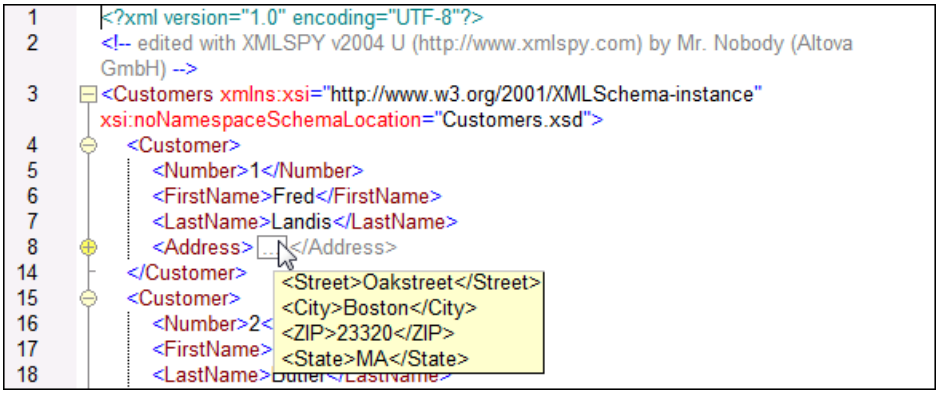
ブックマークを以下のコマンドを使用して編集、および、移動することができます：

- **ブックマークの挿入/削除** (Ctrl + F2)
- **次のブックマークへ移動** (F2)
- **前のブックマークへ移動** (Shift + F2)
- **全てのブックマークを削除** (Ctrl + Shift + F2)

上記の **出力** メニュー内で使用することができます。 **出力** (または、**XSLT**、または、**XQuery**) ペインを右クリックすると、コンテキストメニューを使用してブックマークコマンドを使用することができます。

ソースの折りたたみ

ソースの折りたたみは、ノードの展開と折りたたみ、およびソースの折りたたみマージン内に表示されるノードを表しています。テキストビュー設定 ダイアログボックス内でマージンを切り替えることができます。展開するには、または、テキストの一部を折りたたむには、ウィンドウの左側の "+" と "-" ノードをクリックします。折りたたまれたコードは、省略記号と共に表示されます。これにより、プレビューされているコードを表示するヒントが開かれます。ヒントの終わりには、省略文字が表示されています。



全ての折りたたみの切り替え コマンドは、全てのノードの展開された、または、折りたたまれたフォームを切り替えます。

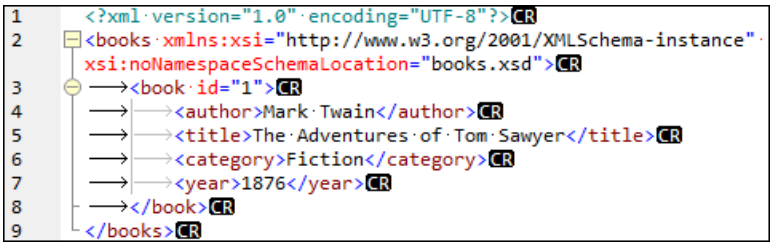
インデントのガイド

インデントのガイドは、ラインのインデントを示す垂直な点線です。**テキストビュー設定** ダイアログボックス内で切り替えることができます。

メモ: 挿入タブ と **スペースの挿入** オプションは、**出力 | XML テキストの整形出力** オプションを使用すると、効果が与えられます。

行末のマーカーと空白文字のマーカー

行末 (EOL) マーカーと空白文字マーカーは、**テキストビュー設定** ダイアログボックス内で切り替えることができます。下のイメージは、行末と空白文字のマーカーが表示されている状態を表示しています。矢印は、タブ文字を表し、"CR" は改行を表し、点は空白文字を表しています。



ズームインとズームアウト

Ctrl キーを長く押して、スクロール (マウスのホイールをスクロール) することによりズームイン、とズームアウトを行うことができます。または、**Ctrl** キーを押しながら "-"、または、 "+" キーを押してもズームイン、および、ズームアウトすることができます。

整形出力

XML テキストの整形出力 コマンドは、アクティブな XML ドキュメントを構成された形式で表示するために、テキストビュー内で再フォーマットします。デフォルトでは、各子ノードが親ノードより4つの空白文字によりオフセットで表示されます。これは、**テキストビュー設定** ダイアログボックスによりカスタム化することができます。

XML ドキュメントを整形出力するには、**出力 | XML テキストの整形出力** メニューコマンドを選択、または、**整形出力** ツールバーボタンをクリックします。

ワード ラップ

現在アクティブなドキュメント内で、ワード ラップを切り替えるには、**出力 | ワードラップ** メニューコマンドを選択、または **ワードラップ** ツールバーボタンをクリックします。

テキストのハイライト

テキストを選択すると、ドキュメント内で選択されたテキストに一致する箇所が自動的にハイライトされます。選択箇所は薄い青でハイライトされ、一致する箇所は薄いオレンジでハイライトされます。選択箇所と一致する箇所は、灰色のマーカーによりスクロールバー上に表示されます。現在のカーソルの位置は、スクロールバー上で青いマーカーにより表示されています。

テキストのハイライトをオンにするには テキストビュー設定 ダイアログボックス内の **自動ハイライトの有効化** を選択します。選択範囲は、文字全体、または、固定された文字数に設定することができます。大文字と小文字の区別も指定することができます。

文字の選択範囲は、選択範囲の文字の最初の文字から開始する、一致する文字の最小数を指定することができます。例えば、2つ、または、2文字以上の文字が一致するように選択することができます。この場合、1つの文字の選択は、一致とは考えられません。2文字以上の一致のみが一致と考えられます。ですから、この場合、`t` を選択すると、一致は表示されません。`ty` を選択すると、`ty` に一致するか表示されます。 `typ` を選択すると、`typ` に一致するすべてのか表示されます。

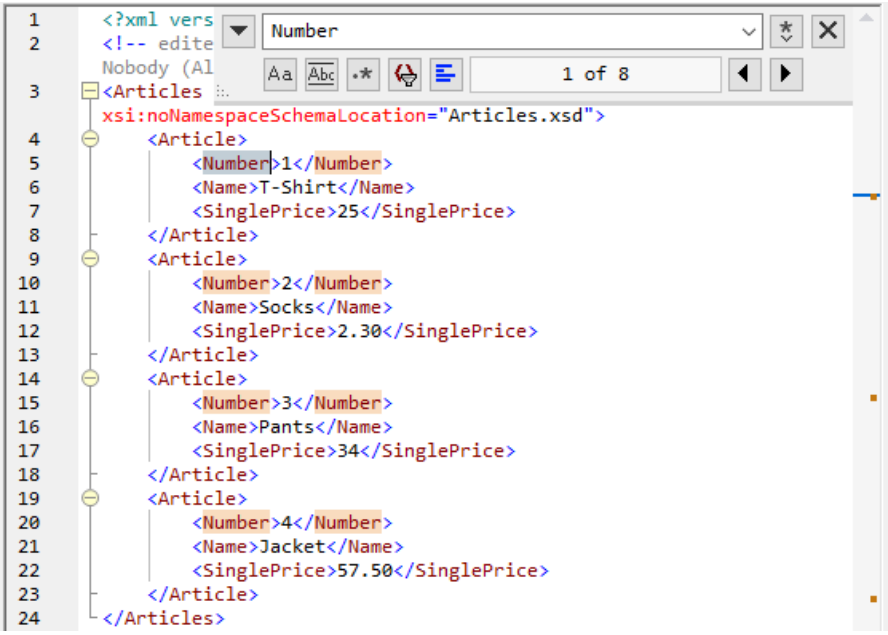
文字の検索に関しては、文字を区切るために以下が考慮されます：角かっこのない要素名、要素タグの角かっこ、属性名、引用符無しの属性の値。

4.1.8 テキストビュー内の検索

Output出力 ペインと**XSLT** ペイン内のテキストは、広範囲のオプションの組み合わせと視覚的な補助を用いて、検索をすることができます。

検索を開始するには、**Ctrl+F** を押します (または、メニューコマンド **編集 | 検索** を選択します)。ダイアログ内に入力された検索用語を、ドキュメント全体で、または、選択された範囲で検索することができます。

- 検索する文字列を入力、または、コンボボックスを使用して最近使用した10件の文字列を選択します。
- 検索する文字列を入力、または、選択すると、全ての一致がハイライトされ、スクロールバー内で一致がベージュ色で表示されます。
- 現在選択されている一致は、他の一致と異なる色で表示されます。一致の位置はスクロールバー上で濃い青のカーソルマーカーで表示されます。
- 検索用語フィールド内に一致の総数が、現在選択されている一致のインデックス位置と共に表示されます。例えば、`2 の 4` は、4つ中の2番目の一致が選択されていることを示しています。
- 右下の **前へ** **(Shift+F3)** と **次へ** **(F3)** ボタンを選択することにより、1つの一致から次の一致へ両方向に移動することができます。



検索ダイアログを閉じるには、右上の**閉じる** ボタンをクリック または、**Esc** を押します。

以下の点に注意してください：

- 検索ダイアログにモードは存在しません。これは、検索ダイアログは、テキストビューを使用する場合でも、開き続けることができます。
- ダイアログボックスを開く前にテキストが選択されている場合、選択されたテキストは、自動的に検索用語フィールドに挿入されます。
- 選択範囲内で検索を行うには、以下を行います：(i) 選択範囲をマークします (ii) *選択範囲内で検索* オプションをオンにして、検索範囲をロックします。 (ii) 検索用語を入力します。他の選択範囲内を検索するには、現在の選択範囲を、*選択範囲内を検索* オプションをオフにしてアンロックします。そして、新規の選択範囲を*選択範囲内を検索* オプションをオンにして切り替えます。
- 検索ダイアログが閉じられると、現在の検索は**F3** を押すことにより順方向検索し、**Shift+F3** を押すことにより逆方向検索します。検索ダイアログは再度表示されます。

検索オプション

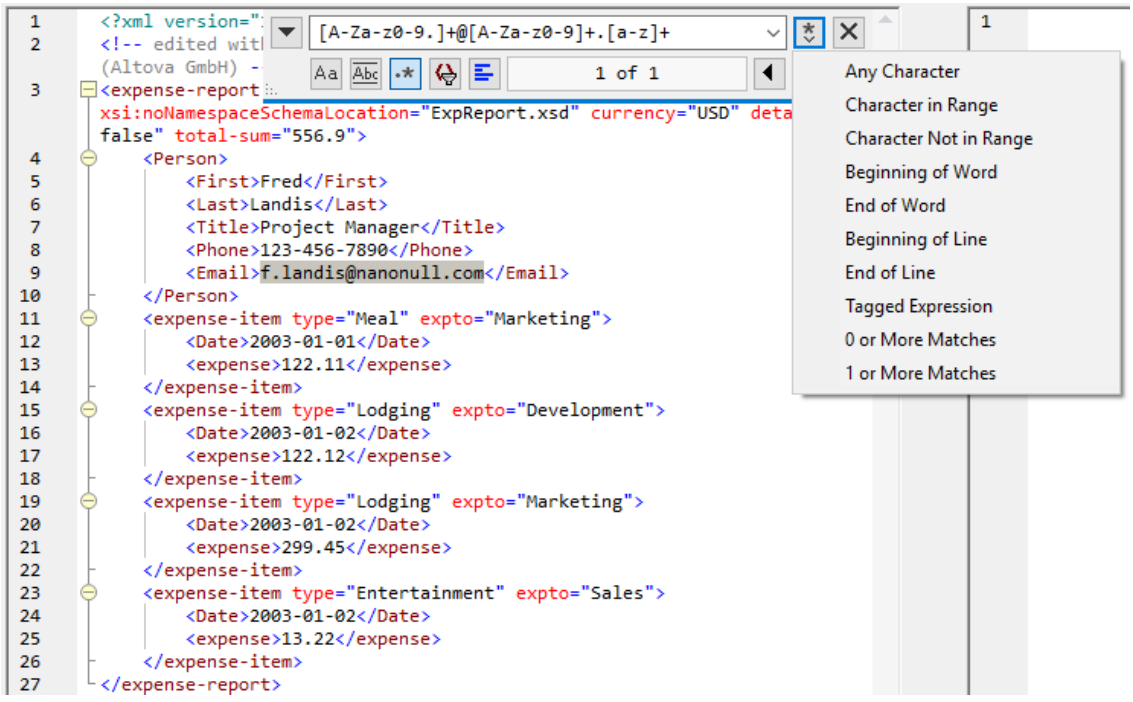
検索の条件は、検索フィールドの下にあるボタンにより指定することができます。オプションがオンになっている場合、ボタンの色は青に変更されます。以下のオプションから選択することができます：

オプション	アイコン	詳細
大文字と小文字を区別する	Aa	切り替えられると、大文字と小文字を区別する検索が行われます（「Address」は「address」とは異なります）。
単語単位で検索	Abc	テキスト内の文字のみが一致されます。例えば、入力文字列 *fit* に対して、**単語単位を一致する** がオンにされていると、単語 *fit* のみが検索文字列に一致します。*fitness* 内の *fit* は、一致しません。
正規表現	.*	オンに切り替えられると、検索用語は、正規表現として読み取られます。正規表現の

		使用」を参照してください。
アンカーの検索		検索用語が入力されると、ドキュメント内の一致がハイライトされ、一致の内の1つが現在の選択としてマークされます。 **アンカーの検索** の切り替えは、最初の選択がカーソルの位置に対して相対的かを決定します。 **アンカーの検索** がオンに切り替えられると、現在選択されている一致が選択され、現在のカーソルの場所の次の一致に一致します。 **アンカーの検索** がオフに切り替えられていると、現在選択されている一致はドキュメントの最初から数えて最初の一致が一致します。
選択範囲内の検索		オンに切り替えられると、現在のテキストの選択範囲をロックし検索を選択されている範囲に制限します。それ以外の場合、ドキュメント全体が検索されます。テキストの新しい範囲を選択する前に、選択範囲内の検索 オプションをオフに切り替えて現在の選択範囲のロックを解除します。

正規表現の使用

正規表現 (regex) を使用して、テキスト文字列を検索することができます。これを行うには、最初に *正規表現* .* オプションをオンに切り替えます。これにより、検索用語フィールド内のテキストが正規表現として評価されるために検索用語内のテキストを指定することができます。次に、検索用語フィールド内に正規表現を入力します。正規表現の作成をヘルプするために、検索用語フィールドの右にある **正規表現 ビルダー** ボタンをクリックします。ビルダー内のアイテムをクリックし、対応する正規表現メタデータを検索フィールドに入力します。下のスクリーンショットは、電子メールアドレスを検索する単な正規表現を示しています。



正規表現メタ文字のカスタムセットは、テキストを検索し置き換える際にサポートされます。

.	全ての文字に一致します。これは単一の文字のプレースホルダです。
\(abc\)	\(and \) メタ文字は、タグ付けされた式の開始と終了をマークします。タグ付けされた式は、後で参照

	(バックレファレンス)するために一致する箇所をタグ(「記録」)する際に役にたちます。タグ付けされた式は、正規表現内の .NET フレーバーの一致したサブ式(インデックスされたグループ)に類似しています。9個までのサブ式をタグ付け(そして後から参照)することができます 例えば、**\(the\) \1** は、文字列 the the に一致します。この式は、以下のように説明することができます:以前に一致したタグ付けされた箇所が後に続き、スペース文字が後に続く、文字列「the」を一致、(および、タグされた箇所として記録)します。
\n	n が1 から9 の箇所では、n は、最初から9番目のタグ付けされた箇所を指します(上を参照してください)。
\<	単語の先頭に一致します。
\>	単語の末尾に一致します。
\	バックスラッシュが後に置かれている文字をエスケープします。すなわち、式 \x は、文字 x を文字通り使用することができます。例えば、\[は、文字のセットの開始としてではなく、[として解釈されます。
[...]	このセット内の文字に一致します。例えば、**[abc]** は、a、b または c の文字に一致します。範囲を使用することができます:例えば、小文字のために **[a-z]** を使用します。
[^...]	このセット内では内文字に一致します。例えば、**[^A-Za-z]** は、アルファベット文字以外の文字に一致します。
^	(セット内で使用されていない限り)行頭に一致します。
$	行末に一致します。例えば、**A+$** は、行末の A に一致します。
*	前の式のゼロ、または複数の発生に一致します。例えば、**Sa*m** は Sm、Sam、Saam、Saaam などに一致します。
+	前の式の1つの、または複数の発生に一致します。例えば、**Sa+m** は Sam, Saam, Saaam などに一致します。

特殊文字の検索と

正規表現 オプション .* が有効化されていると想定して、テキスト内の特殊文字を検索することができます

- \t (タブ)
- \r (キャリッジリターン)
- \n (新しいライン)
- \\ (バックスラッシュ)

例えば、タブ文字を検索するには、**Ctrl + F** を押して、.* オプションを選択し、検索ダイアログボックス内に **\t** を入力します

4.1.9 XSLT コードのプレビュー

XSLT 1.0 または XSLT 2.0 を変換言語として選択している場合、MapForce により生成された XSLT コードをプレビューすることができます。(変換言語の選択を参照)。

変換された XSLT 1.0 (または XSLT 2.0) コードをプレビューするには、以下を行います:

- XSLT 1.0 コードをプレビューするには、マッピングウィンドウの下の **XSLT** タブをクリックします。
- XSLT 2.0 コードをプレビューするには、マッピングウィンドウの下の **XSLT2** タブをクリックします。

メモ: XSLT (または XSLT2) タブは、XSLT (または XSLT2 をそれぞれ) 変換言語として選択すると使用すること

ができます。

4.1.10 XSLT コードの生成

XSLT コードを生成する:

1. メニュー「ファイル| コードの生成 | XSLT 1.0 (またはXSLT 2.0)」を選択します。
2. 生成されたXSLT ファイルを保存するフォルダーを選択し、 **「OK」**をクリックします。MapForce は、コードを生成し、メッセージウィンドウ内にオペレーションの結果を表示します。

生成されたxslt ファイルの名前には、 **<A>MapTo<B>.xslt** の書式を持ちます:

- "<A>" は、マッピング設定内の **アプリケーション名** フィールドの値です (次を参照してください マッピング設定の変更).
- "<B>" は、ターゲットマッピングコンポーネントの名前です。この値を変更するには、ターゲットコンポーネントの設定を開き、 **コンポーネント名** フィールドの値を変更してください (次を参照してください コンポーネント設定の変更)。

XSLT ファイルが保存されるフォルダーは、RaptorXML Server によりデータ変換を行うことができる **DoTransform.bat** というバッチファイルを含みます。

RaptorXML Server を使用して変換を行う:

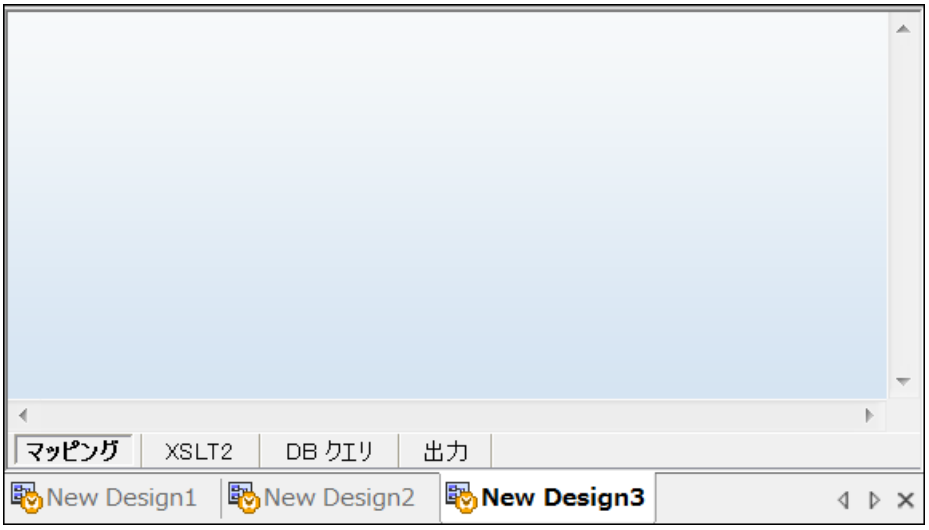
1. RaptorXML をダウンロードページからダウンロードしインストールします (https://www.altova.com/ja/download-trial-server.html)。
2. 上の操作で指定した出力フォルダーに入りされている**DoTransform.bat** バッチファイルを起動します。

RaptorXML がインストールされた場所を環境変数のPath 変数へ追加する必要がある場合があります。 ウェブサイト上のドキュメントページでRaptorXML ドキュメントを確認することができます (https://www.altova.com/ja/documentation.html)。

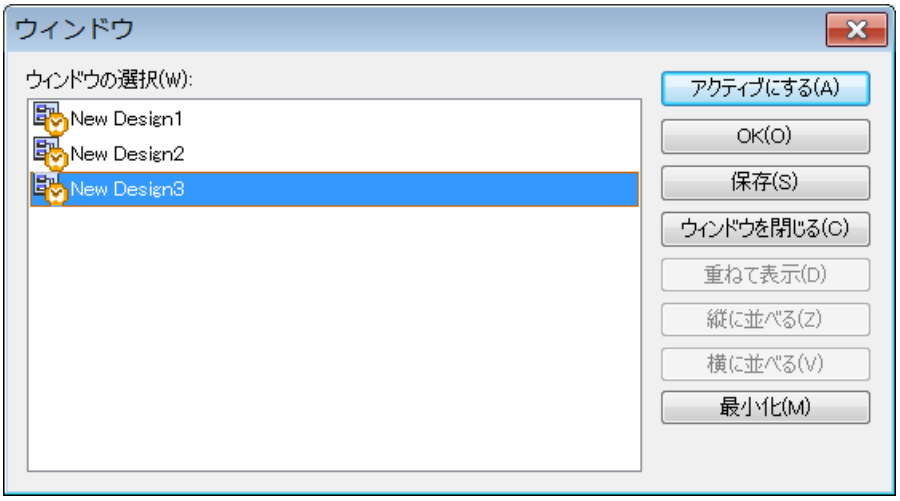
4.1.11 複数のマッピングウィンドウとの作業

MapForce は、複数のドキュメントインターフェイス (Multiple Document Interface) (MDI) を使用します。MapForce で開く各マッピングファイルには個別のウィンドウがあります。これにより、複数のマッピングウィンドウと作業、または並べ替えやサイズ調整をMapForce メイン (親) ウィンドウで行うことができます。で行うことができます。全ての開かれているウィンドウを標準ウィンドウのレイアウトを使用して以下のように並べ替えることができます:上下に並べる、左右に並べる、重ねて表示。

複数のマッピングがMapForce 内で開かれている場合、マッピングペインの下の部分に表示されているタブを使用して素早く切り替えることができます。



ウィンドウ管理オプションは、**『ウィンドウ』**メニューと**『ウィンドウ』**ダイアログボックスで使用することができます。**『ウィンドウ』**ダイアログボックスから、アクションを実行、または現在開かれているマッピングウィンドウで(保存、閉じる、または最小化を含む)アクションを使用することができます。



ウィンドウダイアログボックス

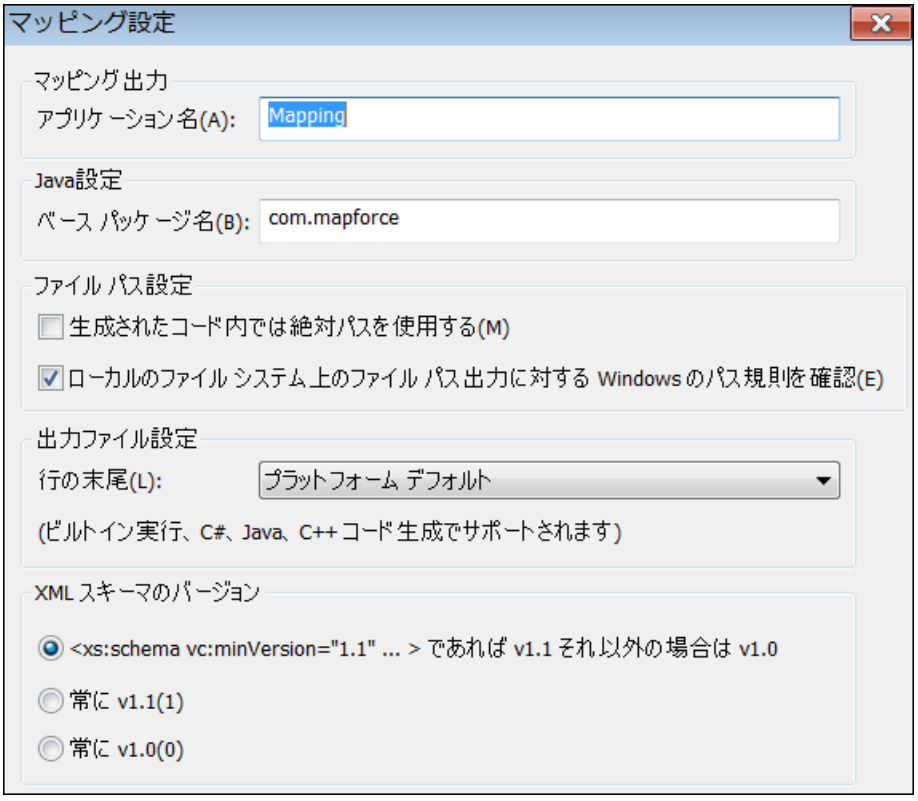
『ウィンドウ』ウィンドウを使用して、ウィンドウダイアログボックスを開くことができます。ウィンドウダイアログボックスから複数のウィンドウを選択するには、必要なエントリを**『Ctrl』**キーを押しながらクリックしてください。

4.1.12 マッピング設定の変更

マッピング設定ダイアログボックスから現在アクティブなマッピングデザインファイルのドキュメント特有の設定を変更することができます。この情報は *mfd ファイルに保管されます。

マッピング設定ダイアログボックスの開きかた:

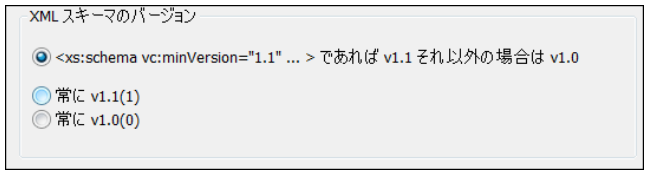
- **『ファイル』**メニューから**『マッピング設定』**をクリックします。



マッピング設定ダイアログボックス

使用することのできる設定は、以下のとおりです。

アプリケーション名	生成される変換ファイルにて使用されるXSLT 1.0/2.0 ファイル名プレフィックス が定義されます。
生成されたコード内でパスを絶対パスにする	生成されたコード内 でファイルパスが相対または絶対パスであるかを定義します。詳細に関しては、生成されたコード内のパスについてを参照してください。
Windows パスがファイルパスの規則であることを確認	ローカルのファイルシステム上のファイルパス出力 ...チェックボックスにより、Windows のパス規則が使用されます。XSLT2 またはXQuery）により出力が行われる場合、現在処理されているファイル名は、document-uri 関数により取得され、内部的にはローカルファイルに対してfile:// というパス形式のURI が使用されます。 このチェックボックスにチェックが入れられている場合、file:// 形式のURI パスがWindows ファイルパス(例："C:\...") へ自動的に変換され、それ以降の処理を単純化することができます。
XML スキーマバージョン	マッピングファイルで使用されたXML スキーマバージョンを定義することができます。バージョン 1.0 または 1.1に準拠するスキーマを**ロード**するかを定義します。すべてのバージョン 1.1特有の機能は現在サポートされています。 *xs:schema vc:minVersion="1.1"* 宣言が存在する場合、バージョン 1.1 が使用されます。それ以外の場合は、 1.0が使用されます。



XSD ドキュメントが`vc:minVersion` 属性 または`1.0` または`1.1` 以外の`vc:minVersion` 属性の値を持たない場合、XSD 1.0 がデフォルトのモードです。

メモ: `vc:minVersion` 属性 と`xsd:version` 属性を混同しないでください。前者は、XSD バージョン番号を示し、後者はドキュメントバージョン番号を示します。

既存のマッピング内のこの設定を変更することは、選択されたスキーマバージョンの全てのXML スキーマを再ロードすることを意味し、妥当性を変更する可能性があります。

4.2 コンポーネントとの作業

コンポーネントは、MapForce 内のマッピングデザインで中心的な要素です。全般的に、「コンポーネント」という用語は、データソースとデータターゲットとしての役割を果たすオブジェクト、または中間処理の段階にあるマッピング内のデータに簡単な名称を与えるために役に立ちます。

コンポーネントには以下の 2つの主なカテゴリがあります :構造コンポーネントと変換コンポーネント。

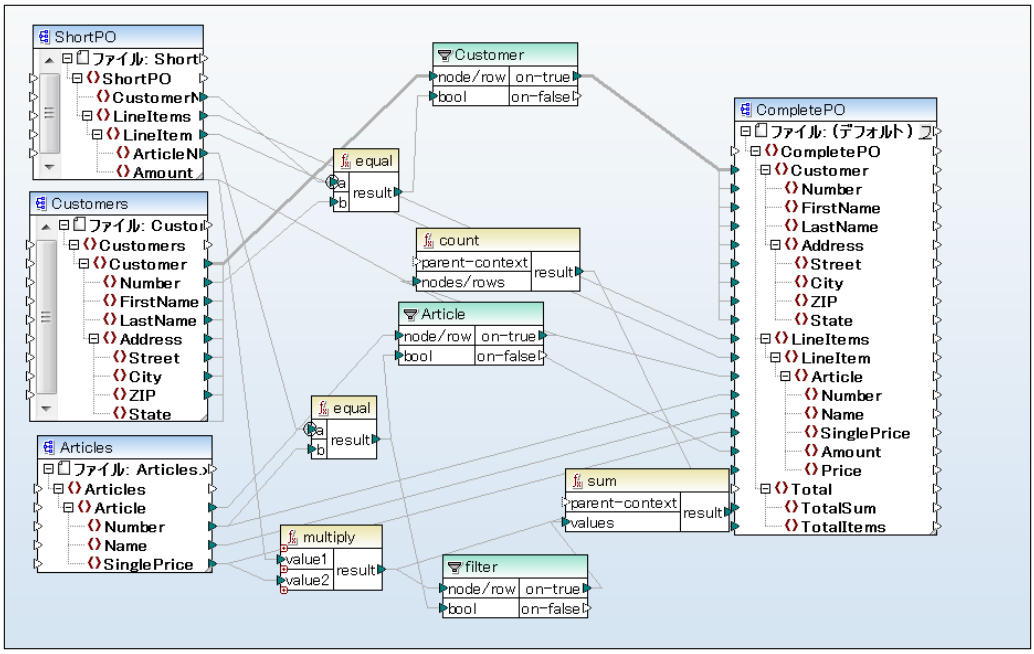
構造コンポーネントは、データの抽象的な構造とスキーマを表します。 例 : (メニューコマンド **挿入 | XML スキーマ/ファイル**) を使用して) マッピングエリアにXML ファイルを追加すると、マッピングコンポーネントになります。構造コンポーネントとその特化についての詳細は、 データソースとターゲットを参照してください。一部の例外を除いて、構造コンポーネントは、アイテムとシーケンスから構成されます。 アイテムはマッピングユニットの最も低いレベルです (例 :XML ファイル内の単一の属性、または 単純型の要素)。シーケンスはアイテムのコレクションです。

変換コンポーネントは、データを変換 (例 :関数)、または変換の手助け (例 :定数または変数) をします。これらのコンポーネントを使用して、複数のデータ変換タスクを達成する方法に関しては、 マッピングのデザインを参照してください。

構造コンポーネントを使用することにより、ファイルまたはその他のソースからデータを読み込み、また、ファイルまたはその他のソースへデータを書き込み、マッピング処理内の中間の段階で (例 : プレビューするために) データを保存することができます。この結果、構造コンポーネントには、以下の3つの型があります :

- ソース。マッピングエリアの左側に配置し、MapForce にデータを読み込むように命令し、コンポーネントをソースとして宣言します。
- ターゲット。マッピングエリアの右側に配置し、MapForce にデータを書き込むように命令し、コンポーネントをソースとして宣言します。
- パススルー。これは特別なコンポーネントの型でソースとターゲットとしての役割を果たします。 (更に詳しい情報に関しては、 チェーンマッピング / パススルーコンポーネントを参照)。

マッピングエリアでは、コンポーネントは三角形で表示されます。次のサンプルマッピングは、 3 つのソースコンポーネント、1つの XML コンポーネント、ソースに書き込まれる前にデータが通過する複数の変換コンポーネント (関数とフィルター) を表しています。



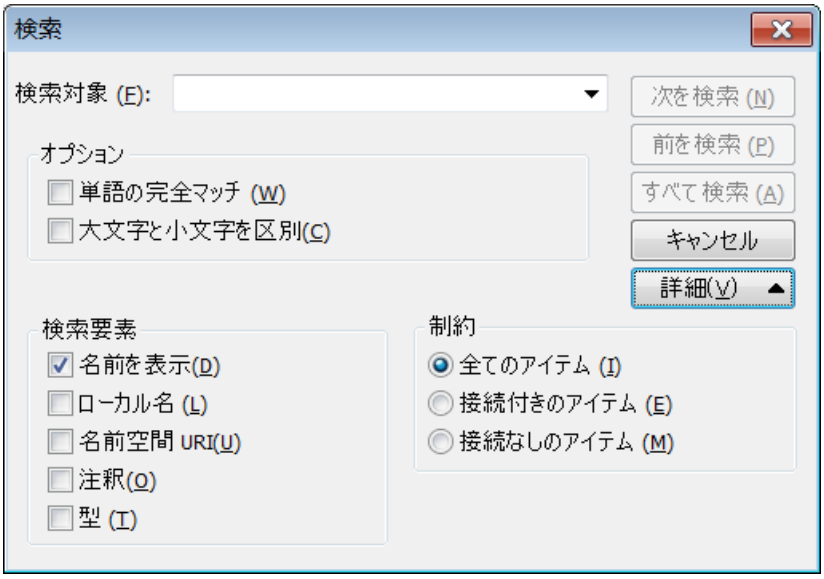
CompletePO.mfd

このマッピングサンプルは以下のパスで見つけることができます：**<Documents>\Altova\MapForce2017\MapForceExamples\CompletePO.mfd**。

4.2.1 コンポーネント内の検索

特定のノード/アイテムをコンポーネント内で検索する方法：

1. 検索するコンポーネントをクリックします。「CTRL+F」キーをクリックします。
2. 検索用語を入力して、**「次を検索」**をクリックします。

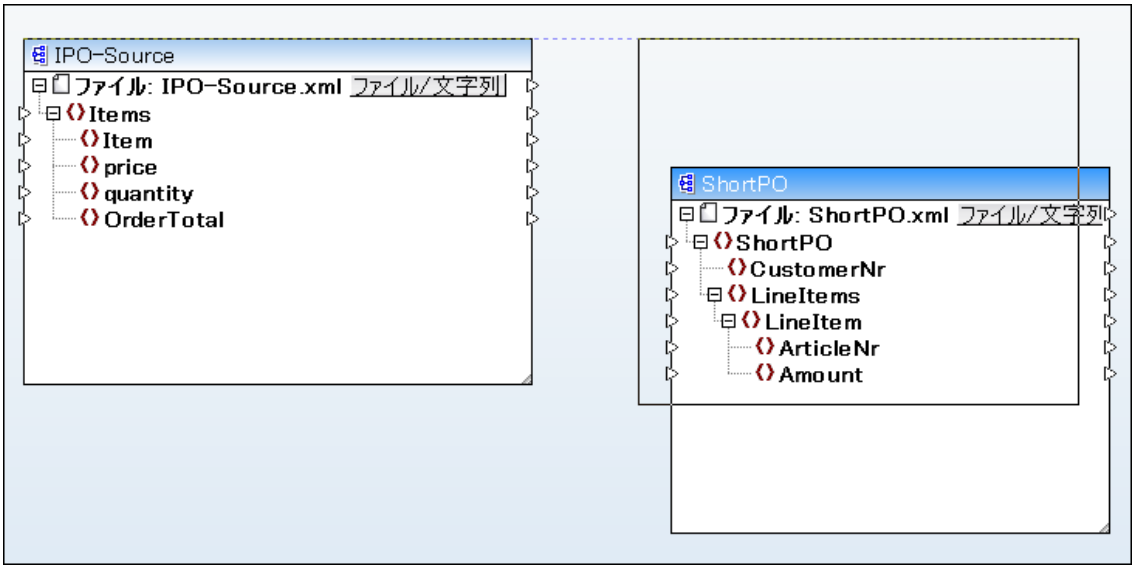


高度なオプションを使用して、どのアイテム (ノード) が検索されるか、また特定の接続をベースにした厳密な検索オプションを定義します。

4.2.2 コンポーネントの整列

コンポーネントをマッピングペイン内でドラッグすると、MapForce は、自動配置 ガイドラインを表示します。このガイドラインは、マッピングウィンドウ内で他のコンポーネントと位置を合わせる際とても役に立ちます。

下のサンプルマッピングでは、下のコンポーネントを移動します。ガイドラインは、マッピングの左のコンポーネントに位置を合わせることができます。



コンポーネント自動配置 ガイドライン

このオプションを有効化または無効化する：

1. **「ツール」** メニューから **「オプション」** をクリックします。
2. **「編集」** グループから、**「マウスのドラッグにてコンポーネントを整列」** のチェックボックスをチェックします。

4.2.3 コンポーネント設定を変更する

コンポーネントをマッピングエリアに追加した後、から適用することのできる設定をコンポーネント設定ダイアログボックスから構成します。コンポーネント設定 ダイアログボックスを以下の方法で開くことができます：

- コンポーネントを選択して、 **「コンポーネント」** メニューから **「プロパティ」** をクリックします。
- コンポーネントヘッダーをダブルクリックします。
- コンポーネントヘッダーを右クリックして、 **「プロパティ」** をクリックします。

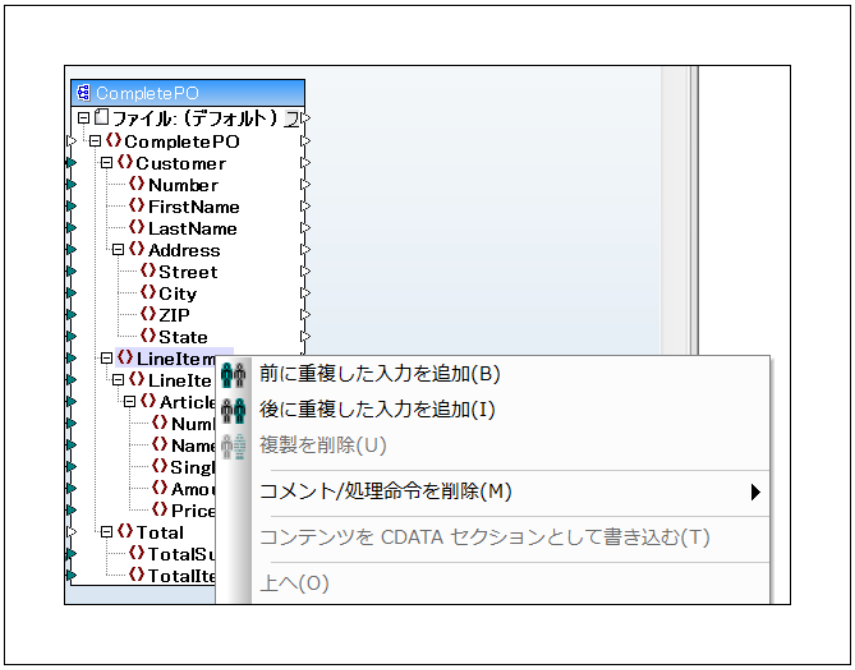
コンポーネント設定ダイアログボックスで使用することのできる設定に関しては XML コンポーネント設定を参照してください。

XML などの、ファイルベースのコンポーネント、 **「ファイル」**（ File ）ボタンがルートノードの横に表示されます。このボタは、単一のマッピング内で複数のファイルを処理または生成する際に適用することのできる高度なオプションを指定します（複数の入力または出力ファイルを動的に処理を参照）。

4.2.4 入力の複製

1つ以上のソースからデータを受け入れるために コンポーネントを構成する必要がある場合があります。例えば、2つの異なるスキーマからのデータを単一のスキーマに変換する必要があるとします。目的のスキーマが2つのソーススキーマからのデータを受け入れるようにするには、コンポーネント内の入力アイテムを複製して行います。入力の複製は、ターゲットコンポーネントであるコンポーネントにとってのみ意味があります。与えられたターゲット コンポーネントは、必要に応じて、必要な数複製することができます。

特定の入力アイテムを複製するには、アイテムを右クリックして、コンテキストメニューから **「後 前に入力を複製する」** を選択します。



上のイメージでは、アイテム `LineItem` が 2 番目のソースからマップを可能にするために複製されています。

入力を複製すると、元の入力と複製された入力の間をマップすることができます。例：これによりデータをソース A からもとの入力に、またデータをソース B から複製された入力にコピーすることができます。

メモ： XML インスタンスを無効にするため、XML 属性の複製は許可されていません。スキーマ内の要素の `maxOccurs` 属性の値に関わらず、XML 要素の場合、入力の複製は許可されています。この振る舞いは、スキーマが後に変更することが可能なため、また、ソースデータが任意のため、意図的です。 例えば、マッピングは、マッピング上で入力が複製されても、単一の XML 要素を生成することができます。

順序を追った例は、複数のソースから1つのターゲットにマップを参照してください。

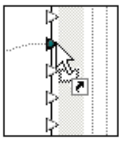
4.3 接続との作業

マッピングは、データを1つのフォーマットから他のフォーマットに変換することを意味します。基本的なマッピングシナリオでは、マッピングエリアにソースとターゲットデータを表すコンポーネントを追加します（例：ソースXML スキーマと目的のスキーマ）、そして視覚的なマッピング接続を2つの構造間に描きます。ですから、接続はソースから目的地へどのようにデータがマップされたかを表す視覚的な表示といえるでしょう。

コンポーネントには、マッピングで小さい三角形として表示される入力と出力があり、これらはコネクタと呼ばれます。出力コネクタは接続を描くアイテムの右横に配置されます。

2つのアイテム間に接続線を引く:

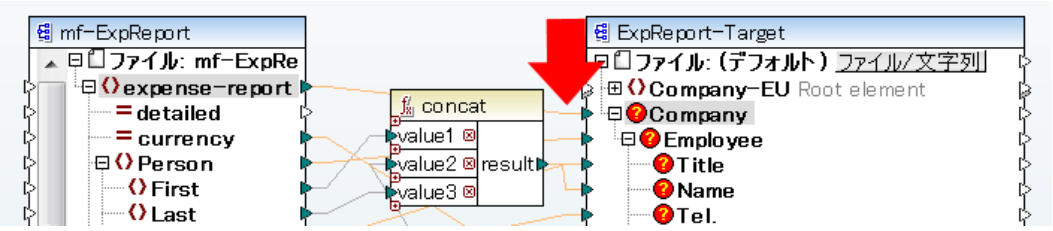
- ソースアイテムの出力コネクタをクリックし、目的アイテムまでドラッグします。ドロップアクションが許可されると、ヒントのリングがカーソルの横に表示されます。



入力コネクタは入力される接続のみを受け入れます。同じ入力に第2番目の接続を追加使用とすると、接続を新規の接続と置き換えるか、または入力アイテムを複製するかを問うメッセージボックスが表示されます。出力コネクタは複数の接続を持つことができ、それぞれが異なった入力を持ちます。

異なったアイテムに接続を移動する:

- 接続のスタブ（ターゲットに近い直線のセクション）をクリックして、移動先までドラッグします。

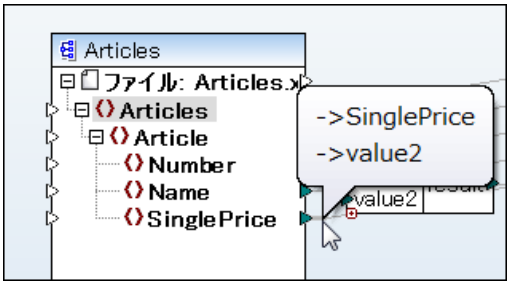


異なったアイテムへの接続をコピーする:

- 接続のスタブ（ターゲットに近い直線のセクション）をクリックして「Ctrl」キーを押しながら移動先までドラッグします。

接続の対極にあるアイテムを表示する:

- マウスポインターを（入力/出力アイコンの近くにある）コネクタの直線部へマウスオーバーすることで、コネクタがハイライトされ、ポップアップが表示されます。ポップアップには、コネクタの対極にあるアイテムの名前が表示されます。同一の出力アイコンから複数の接続が定義されている場合には、最大10の名前が表示されます。スクリーンショットでは、**SinglePrice** と**value2** という複数のターゲットに対して接続が行われていることが理解できます



接続設定 を変更するには以下を行います :

- **「接続」** メニューから **「プロパティ」** をクリックします。 (このメニューアイテムは、接続を選択すると有効化されます)。
- 接続をダブルクリックします。
- 接続を右クリックして、 **「プロパティ」** をクリックします。

接続設定を参照してください。

接続を削除するには、以下を行います :

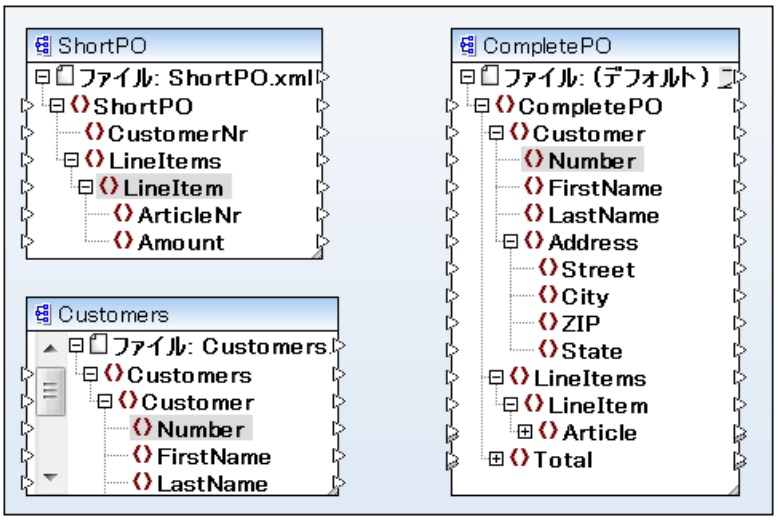
- 接続をクリックして、 **「削除」** キーを押します。
- 接続を右クリックして、 **「削除」** をクリックします。

4.3.1 必須の入力について

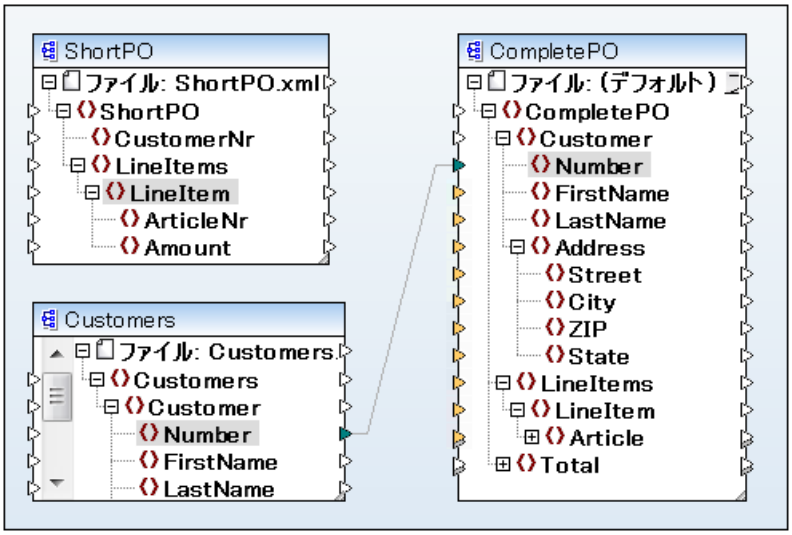
マッピング処理内での手助けをするために、MapForce は、ターゲットコンポーネント内の必須の入力をオレンジ色でハイライトします :

- XML とEDI コンポーネントは、minOccurs パラメーターが 1 と同じまたは 1 より大きいアイテムです。
- In データベース内では、これらは『ゼロではない』として定義されるフィールドです。
- WSDL 呼び出しとWSDL レスポンス (全てのノード)
- 必須と定義されたXBRL ノード
- 関数内では、これらは、パラメーターがマップされると、接続の必要な他の必須のパラメーターがハイライトされる特定の必須のパラメーター です。例 : フィルター入力パラメーターの1つがマップされると、他のパラメータは自動的にハイライトされます。
- MS Excel シート内のワークシート名

例 :
...\MapForceExamples フォルダー内にある、CompletePO.mfd などのマッピング作成の際、挿入されたXML スキーマファイルは下に表示されるとおりです。



Customers コンポーネントのNumber 要素はCompletePO コンポーネントのNumber 要素に接続されています。接続が作成されると、CompletePO コンポーネントの必要なアイテム/ノードは、ハイライトされます。折りたたまれた「Article」ノード/アイコンもハイライトされます。



4.3.2 優先する接続線の表示の変更

MapForce ではマッピングウィンドウに表示されるコネクタの表示方法を選択することができます

選択コンポーネントの接続線の表示 により以下のように表示が切り替わります：

- 全てのマッピングコンポーネントが黒色で表示されるか、
- 現在選択されているコンポーネントに関係したコネクタが黒色で表示されるとともに、その他の接続がグレーで表示されます。

ソースからターゲットへの接続線の表示により以下の表示が切り替わります：

- 現在選択されているコンポーネントに**直接**接続されているコネクタが黒色で表示されるか、
- 現在選択されているコンポーネントの入力または出力アイコンへリンクされているコネクタが黒色で表示されます。

4.3.3 接続の注釈

個別の接続をマッピングに詳細なコメントを与えラベル付けすることができます。 このオプションは **「全ての接続の型」**で使用することができます。

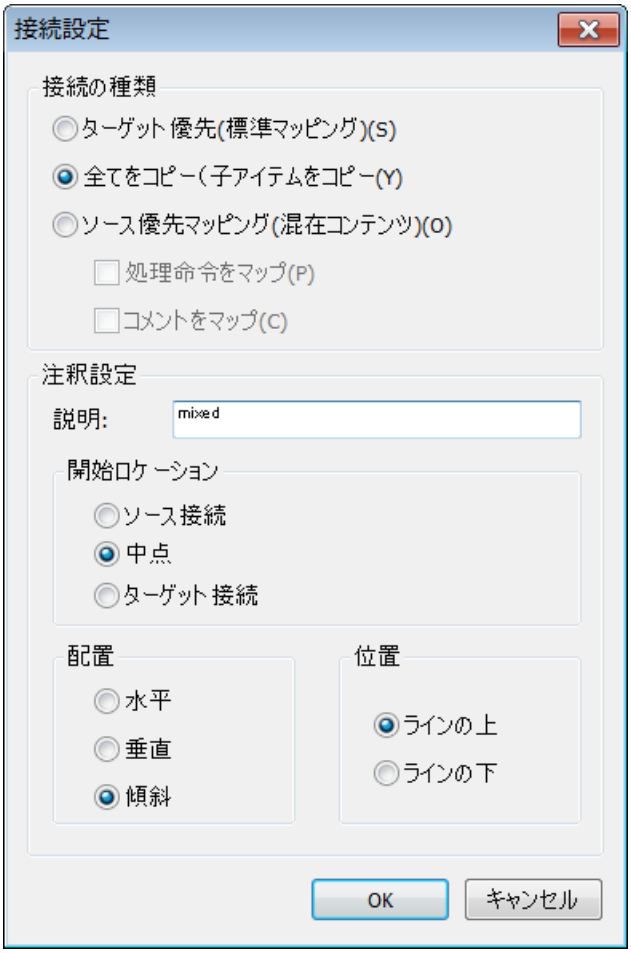
接続に注釈をつける：

1. 接続を右クリックして、コンテキストメニューから**「プロパティ」**を選択します、
2. 現在選択されている接続の名前 を **「詳細」** フィールドに入力します。これにより注釈設定グループ内の全てのオプションが有効化されます。
3. 残りのグループを使用して、ラベルの **「開始の箇所」、「配置」、「位置」**を定義します。
4. 注釈テキストを確認するには、表示オプションツールバー内の **「注釈の表示」** アイコンを有効化します。

メモ： **「注釈の表示」**アイコンが無効化されている場合、接続にカーソルをポイントすると、注釈を表示することができます。 **「ヒントの表示」** ツールバーボタンが表示オプションツールバー内で有効化されている場合、注釈のテキストは吹き出し内に表示されます。

4.3.4 接続設定

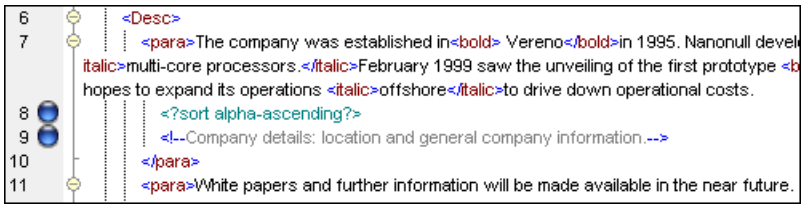
コネクタを右クリックしてコンテキストメニューから**プロパティ**を選択するか、コネクタをダブルクリックして接続設定ダイアログボックスを開き、現在のコネクタに対して特別な（混合コンテンツ）設定を定義することができます。使用することのできないオプションは無効化されています。



接続設定ダイアログボックス

complexType のアイテムのために、以下のマッピングのための接続型を選択することができます（これらの設定は、テキストノードを持たない**complexType** アイテムにも適用することができます）：

ターゲット優先（標準）	接続の型をターゲット優先に変更します（ターゲット優先（標準）マッピング を参照）。
全てコピー（子アイテムのコピー）	接続の種類を「全てコピー」に変更し、ソースとターゲットコンポーネント内の全ての等しいアイテムと自動的に接続します（全てコピー接続 を参照）。
ソース優先（複合コンテンツ）	接続の種類を"ソース優先"に変更し、またマップされる追加要素の選択を有効化します。マッピングに適するために、追加要素はXML ソースファイル内のマップされたアイテムの子アイテムである必要があります。 **「処理命令をマップ」**または**「コメントをマップ」**チェックボックスを有効化することにより、これらのデータグループを出力ファイルに含むことができます。



メモ:CDATA セクションはテキストとして扱われます。

注釈設定グループにより接続へ注釈を与えることができます (接続の注釈 を参照)。

4.3.5 マッチする子要素の接続

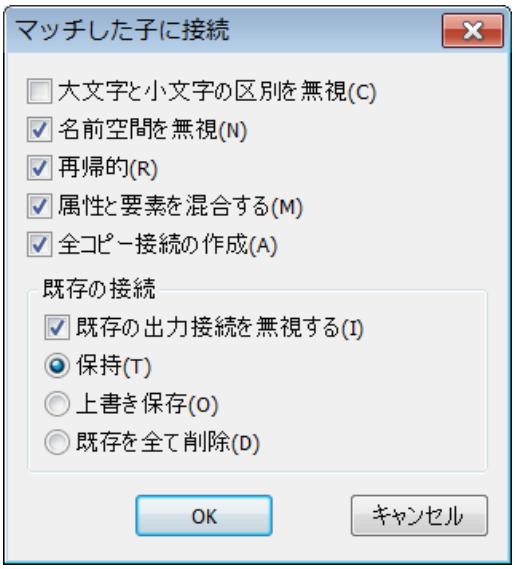
ソースとターゲットコンポーネントの両方内で**同じ名前**のアイテム間で複数の接続を作成することができます。 全てコピー」接続は、デフォルトで作成されます (全てコピー接続を参照)。

「マッチする子を自動的に接続する」オプションをオンまたはオフに切り替えるには、以下を行います:

- **「マッチする子要素の自動接続」**( ) ツールバーボタンをクリックします。
- **接続」**メニューから**「マッチする子要素の自動接続」**をクリックします。

"マッチする子を接続する"のための設定を変更する方法:

1. 両方のコンポーネント内のと **「子アイテム」**いう名前を共有する 2つの 親)アイテムを接続します。
2. 接続を右クリックして、 **「一致する子要素の接続」**オプションを選択します。



3. 必要なオプションを選択して (下のテーブルを参照してください)、 **「OK」**をクリックします。ダイアログボックス内の設定定義に当てはまる、接続が同じ名前の全ての子アイテムのために作成されます。

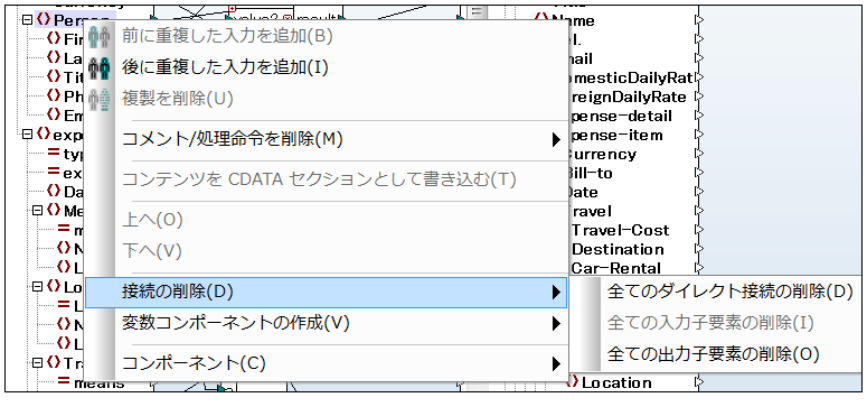
メモ: **「子の自動接続の切り替え」**( ) ツールバーボタンがアクティブな場合、ここで定義する設定が2つのアイテム

を接続する際に適用されます。

大文字と小文字の区別を無視	子アイテム名の大文字と小文字の区別を無視します。
名前空間を無視	子アイテムの名前空間を無視します。
再帰的	一致するアイテム間の接続を再帰的に作成します。すなわち、階層構造内でどれほど深くアイテムがネストされていても、アイテムが同じ名前を持つ限り、アイテム間の接続が作成されます。
属性と要素を混合	有効化されている場合、同じ名前を持つ属性と要素の間の接続を許可します。例：2つの「Name」アイテムが存在する場合、1つが属性、もう1つが要素である場合でも、接続を作成します。
全コピー接続の作成	この設定は、デフォルトの設定では有効化されています。この設定は、「全てコピー」とソースとターゲットアイテム間の接続を（可能であれば）作成します。
既存の出力接続を無視	出力接続が既存の場合でも、追加の接続を一致するアイテムのために作成します。
保持	既存の接続を保持します。
上書きを保存	定義された設定に従い接続を再作成します。既存の接続は破棄されます。
既存を全て削除	新規接続を作成する前に、既存のすべての接続を削除します。

接続の削除

接続、マッチする子を接続するダイアログを使用して作成された接続、または、マッピング処理中の接続をグループ化して削除することができます。



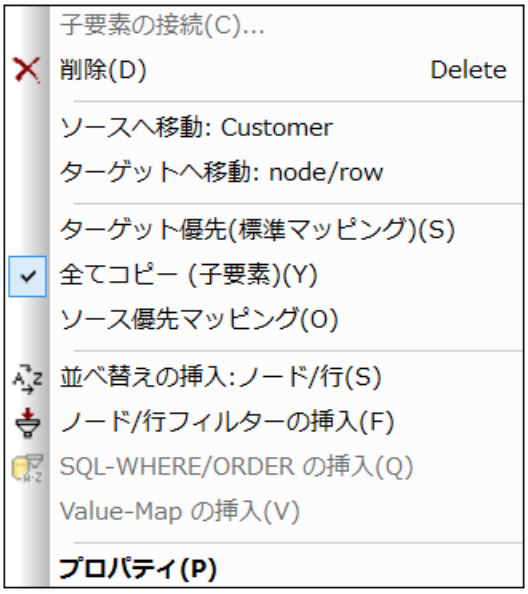
接続の削除：

1. コンポーネント内のアイテムの名前を右クリックします（このサンプルでは「Person」）。
2. **接続の削除 | 全ての..接続の削除」**を選択します。

全てのダイレクト接続を削除	直接マップされた、または現在のコンポーネントから他のソースまたはターゲットコンポーネントにマップされたすべての接続を削除します。
全ての入力子接続を削除	ターゲットコンポーネント内でアイテムを右クリックした場合のみアクティブになります。全ての入力子接続を削除します。
全ての出力子接続を削除	ソースコンポーネント内でアイテムを右クリックした場合のみアクティブになります。全ての出力子接続を削除します。

4.3.6 コンテキストメニューの接続

接続を右クリックすると、以下のコンテキストコマンドを使用することができます。



マッチする子を接続	「マッチする子を接続する」ダイアログボックスを開きます。(マッチする子要素の接続 を参照)。接続がマッチする子アイテムを持つことができる場合、このコマンドを有効化することができます。
削除	選択された接続を削除します。
ソースへ移動 : <item name>	現在の接続のソースコネクタを選択します。
ターゲットに移動 : <item name>	現在の接続のターゲットコネクタを選択します。
ターゲット優先 (標準)	接続の型をターゲット優先に変更します (ターゲット優先接続 を参照)。
全てコピー (子アイテムのコピー)	接続の種類を 全てコピー」に変更し、ソースとターゲットコンポーネント内の全ての等しいアイテムと自動的に接続します (全てコピー接続 を参照)。 ソースアイテムとターゲットアイテムの両方が子アイテムを持つ場合、このコマンド

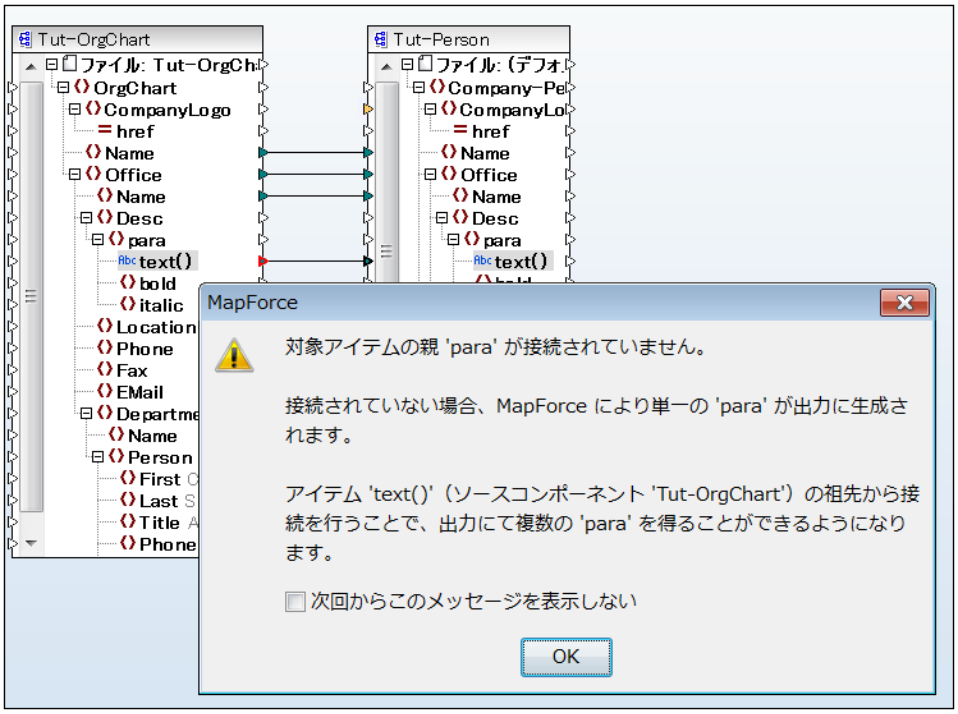
	は有効化されます。
ソース優先（複合型コンテンツ）	接続の種類を「ソース優先」に変更します（ソース優先マッピングを参照）。 ソースアイテムとターゲットアイテムの両方が子アイテムを持つ場合、このコマンドは有効化されます。
並べ替えの挿入：Nodes/Rows	ソースとターゲットアイテム間のソートコンポーネントを追加します。（データの置換えを参照）
フィルターの挿入：Nodes/Rows	ソースとターゲットアイテム間のフィルターコンポーネントを追加します。（フィルターと条件を参照）
Value-Map の挿入	ソースとターゲットアイテム間の Value-Map コンポーネントを追加します。（Value-Maps の使用 を参照）
プロパティ	「接続設定」ダイアログボックスを開きます。（接続設定 を参照）

4.3.7 見つからない親接続の通知

ソースならびにターゲットアイテム間の接続を手動で作成する場合、起こりえるマッピング結果が MapForce により分析されます。子アイテム同士をマッピングした場合、ソースアイテムの親とターゲットアイテムの親も接続するように促すプロンプトが表示されます。

これはマッピングのプレビューを出力ウィンドウにて行う際に、子アイテムの表示を 1 つに限定しないためのものです。ソースノードから得られるものが単一の値ではなく、シーケンス形式の場合、通常親アイテムの接続を行う必要があります。

<Documents>\Altova\MapForce2017\MapForceExamples\Tutorial フォルダーに収められている **Tut-OrgChart.mfd** マッピングを以下に示します。ソースコンポーネントにある text() アイテムがターゲットコンポーネントにある text() アイテムへ接続されると、メッセージボックスが表示され、親アイテムの para が接続されておらず、出力では一度だけ生成される旨が通知されます。複数の para アイテムをターゲットにて表示するには、ソースならびにターゲットアイテムの para を接続する必要があります。



Tut-OrgChart.mfd (MapForce Basic Edition)

ターゲット内で複数の `para` アイテムを生成するには、ソースとターゲット `para` アイテムをそれぞれ接続します。

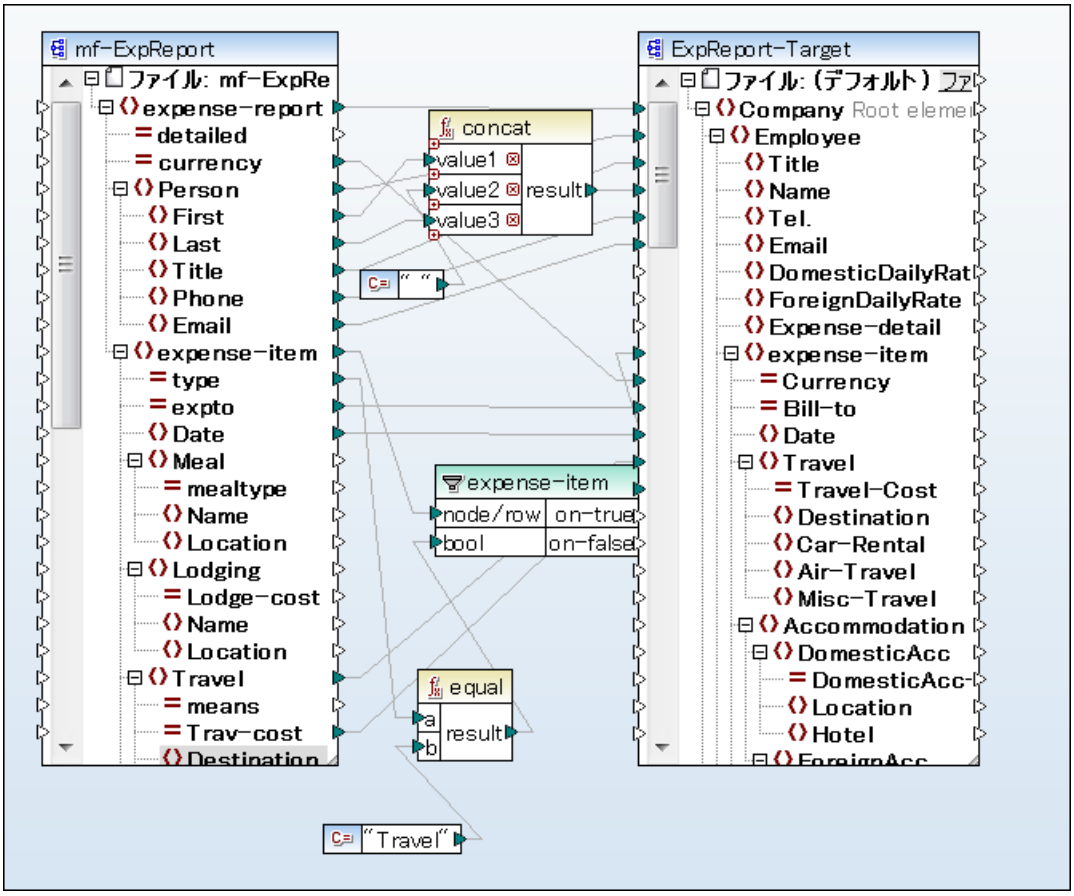
上に表示されるようなメッセージを非表示にするには以下を行います：

1. **「ツール」** メニューから **「オプション」** をクリックします。
2. **「メッセージ」** グループをクリックします。
3. **接続を作成した際、祖先アイテムへの接続を提案アイテム** チェックボックスをクリックします。

4.3.8 接続と子接続の移動

異なったコンポーネントに接続を移動する場合、MapForce は、自動的に同じ子接続をマッチし、この接続が新しい場所に移動されるべきかプロンプトします。この機能の通常の使用方法は、既存のマッピングがある場合、ターゲットスキーマのルート要素の変更です。通常、このような状況が発生すると、すべての降順 接続 を手動でマッピングを再度行う必要がありますが、この機能はそのような状況を回避するために役に立ちます。

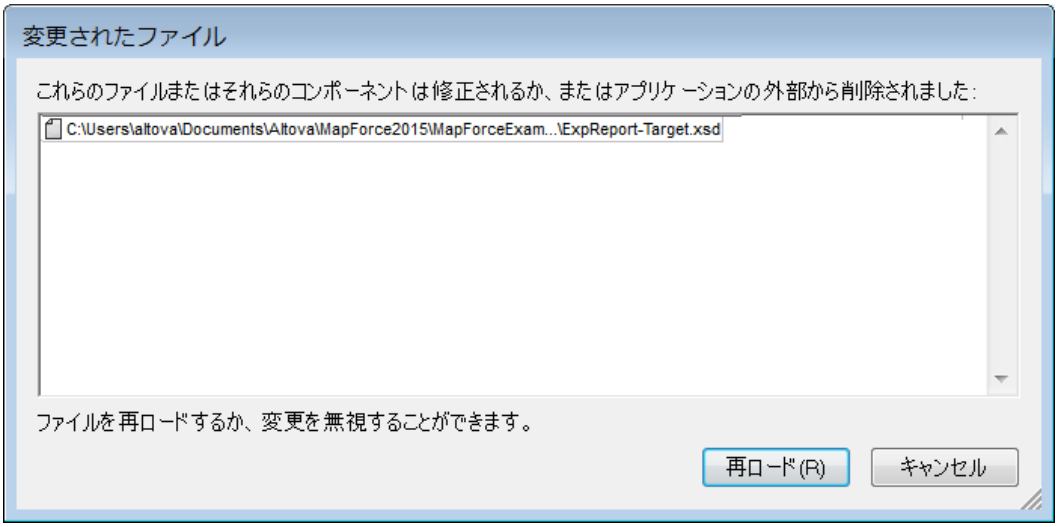
<Documents>\Altova\MapForce2017\MapForceExamples\Tutorial フォルダーで使用することのできる**Tut-ExpReport.mfd** ファイルは、この例で修正され、別名で保存されます。



Tut-ExpReport.mfd (MapForce Basic Edition)

動作を理解するためには、以下を行います：

1. **Tut-ExpReport.mfd** サンプルマッピングを開きます。
2. ターゲットスキーマの`Company` ルート要素を`Company-EU` に変更するために、MapForce の外部で **ExpReport-Target.xsd** スキーマを編集します。MapForce を閉じる必要はありません。
3. ターゲットスキーマの`Company` ルート要素を`Company-EU` に変更すると、MapForce に変更されたファイル"プロンプトが内に表示されます。

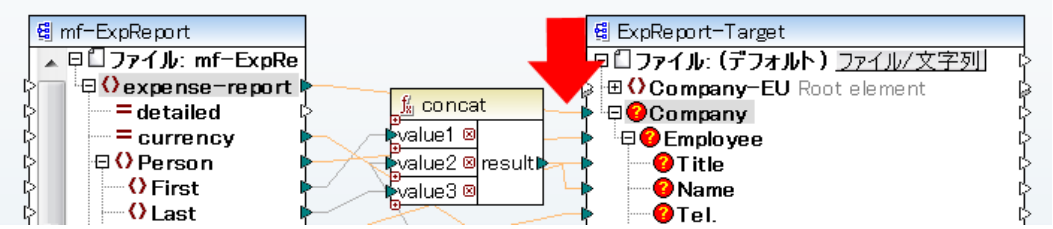


4. 更新されたスキーマを再ロードするために、**「再ロード」**ボタンをクリックします。ルート要素が削除されたため、コンポーネントは複数の見つからないノードを表示します。

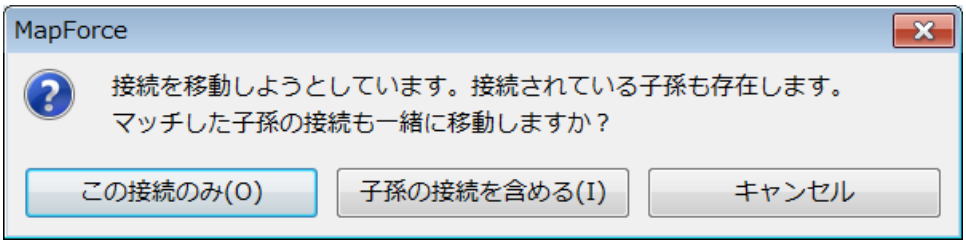
5. コンポーネントの上の **「新しいルート要素の選択」**をクリックします。(ルート要素をコンポーネントヘッダーを右クリックしてコンテキストメニューから **「ルート要素の変更」**を選択することで、ルート要素を変更することができます。)

6. `Company-EU` を新しいルート要素として選択し、**「OK」**をクリックして確認します。`Company-EU` ルート要素がコンポーネントの上に表示されます。

7. ソースコンポーネントの`expense-report` アイテムとターゲットコンポーネントの`Company` アイテム間にある接続のターゲットスタブをクリックして、`Company-EU` ターゲットコンポーネントのルート要素の上にドラッグアンドドロップします。



通知ダイアログボックスが現れます。



8. **「子孫の接続を含む」**をクリックする。これにより、新しいルート要素の下で再度マップするようにMapForce に命令し、マッピングは再度有効となります。

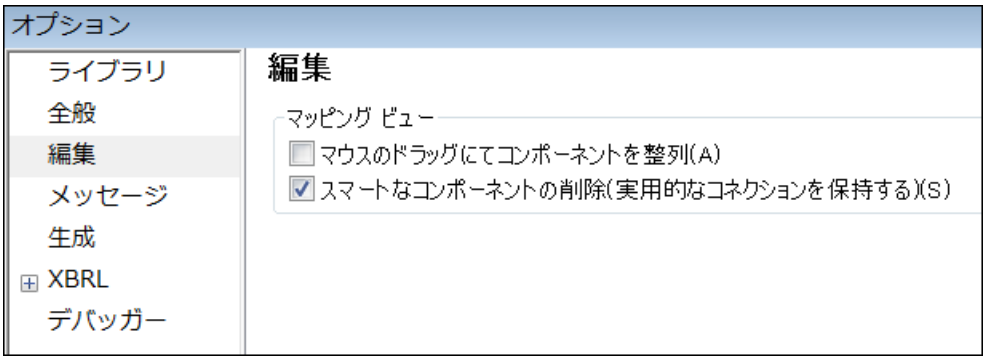
メモ: マッピングするノードが、ソースノードと同じ名前を持ち、異なる名前空間にある場合、通知ダイアログボックスに、以下のを表示する追加ボタンが表示されます: 「子孫とマップ名前空間を含む」。このボタンをクリックすると、同じ名前空間の子接続が同じ子ノードのソース親ノードとして移動されます。このボタンをクリックすると、異なる名前空間ノードの下の同じ子ノードのソース親ノードと同じように同じ名前空間の子接続を移動します。

4.3.9 コンポーネント削除後の接続の保持

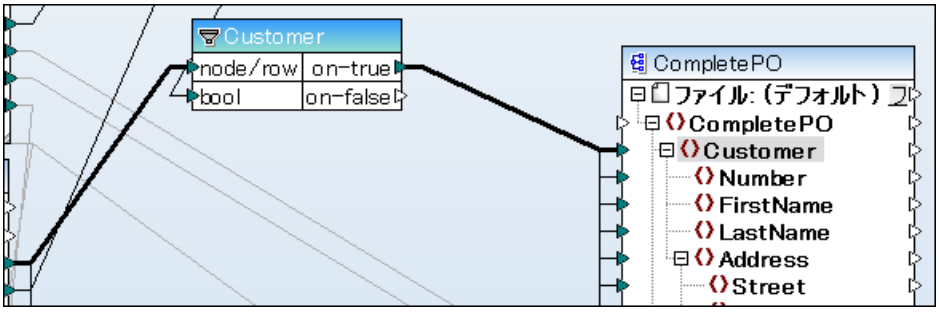
他のコンポーネントに複数の (子) 接続を持つ、コンポーネントを削除すると、何が起こるかを決定することができます。例：フィルター または ソートコンポーネント。全ての子接続を保持し、個別に復元する必要がないためこの機能は役に立ちます。

コンポーネントが削除された後に、子接続を保持または復元すること、または、全ての子接続をすぐに接続することが選択できます。

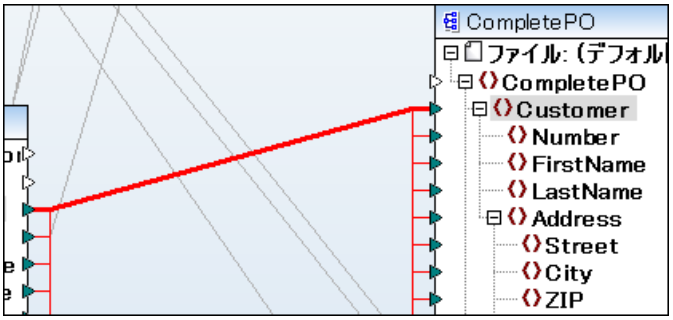
現在の設定を確認するために (タブ) を選択します。チェックボックスのデフォルト設定は、**非アクティブ**です。すなわち「スマートなコンポーネント削除 (役に立つ接続を保持)」が無効化されます。



例：..\MapForceExamples フォルダー内でCompletePO.mfd マッピングを使用するには、チェックボックスがアクティブ化されていると、Customer フィルターは、以下に表示される子アイテムに接続された**全てコピー**接続になります。



Customer フィルターを削除すると、削除の確認を問うプロンプトが表示されます。「はい」を選択すると、フィルターは削除されますが、子接続は保持されます。



その他の接続はまだ選択されています (すなわち、赤で表示される箇所)。削除する場合は、Del キーを押します。

マッピングエリアをクリックすると、コネクタの選択が解除されます。

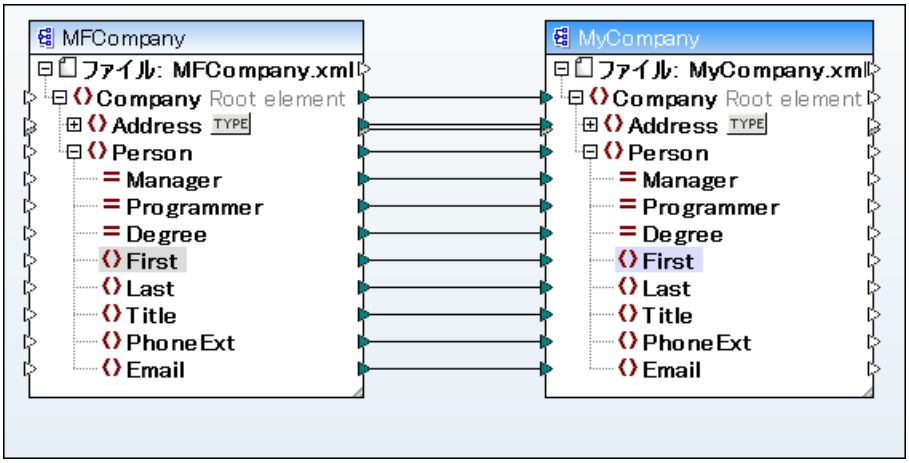
「スマートなコンポーネント削除 ..」チェックボックスが**無効化されている**場合、フィルターを削除すると全ての子接続がすぐに削除されます。

メモ: フィルターコンポーネントが「on-true」と「on-false」出力に接続されている場合、両方の出力のための接続は保持されます。

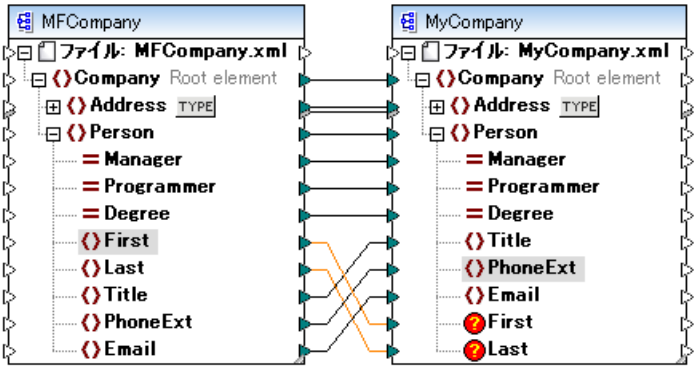
4.3.10 見つからないアイテムの扱い方

時が経つに従って、XML スキーマにて定義された要素や属性が追加/削除され、マッピング内にあるコンポーネントの構造が変更されるということがあります。MapForce ではプレースホルダーアイテムを使用することで、全ての接続を保持し、アイテムが削除された際にコンポーネント間で関連する接続データの保持を行います。

例 :
MFCompany.xsd スキーマファイルをサンプルファイルとして使用します。スキーマファイルを MyCompany.xsd という名前でコピーし、両スキーマ間にて Company アイテムに対する接続を作成します。これで子要素の自動接続が有効になっている場合、コンポーネント間にある全ての子アイテムに対する接続が作成されます。



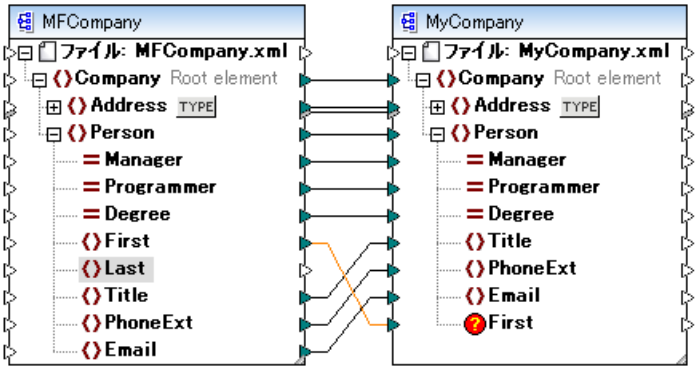
XMLSpy にてMyCompany.xsd を編集し、スキーマ内にあるFirst ならびにLast アイテムを削除します。MapForce へ再度切り替えると、スキーマを**再ロード**するよう促されます。再ロードすることで、MapForce 内にあるコンポーネントが更新されます。



削除された**アイテム**ならびに**コネクタ**が、アイコンとともにMyCompany コンポーネントに表示されます。必要な場合は他のアイテムへコネクタを再接続することができるほか、コネクタを削除することもできます。

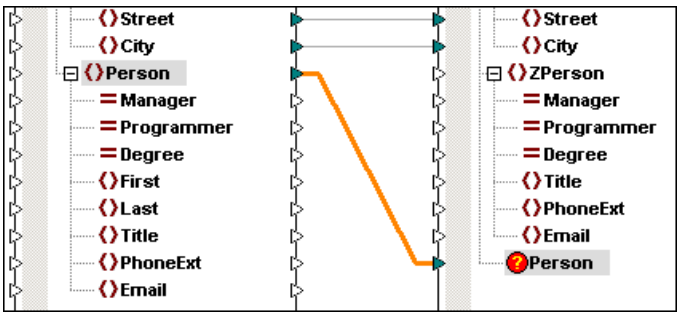
この状態でマッピング（または生成されたコード）のプレビューを行うこともできますが、警告がメッセージウィンドウに表示されます。プレビューならびにコード生成において、上のアイコンで示されるような、見つからないアイテムは無視されます。

ハイライトされているコネクタをクリックし、削除することで、見つからないアイテム（この場合はMyCompany 内部のLast）がコンポーネントから削除されます。



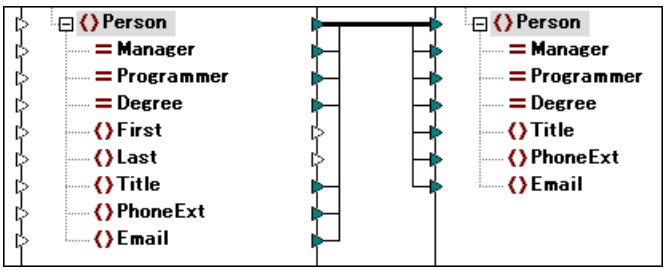
名前変更されたアイテム

例えばPerson という親アイテムがZPerson という名前に変更された場合、オリジナル（Person）の親アイテムコネクタが保持され、子アイテムのコネクタが削除されます。



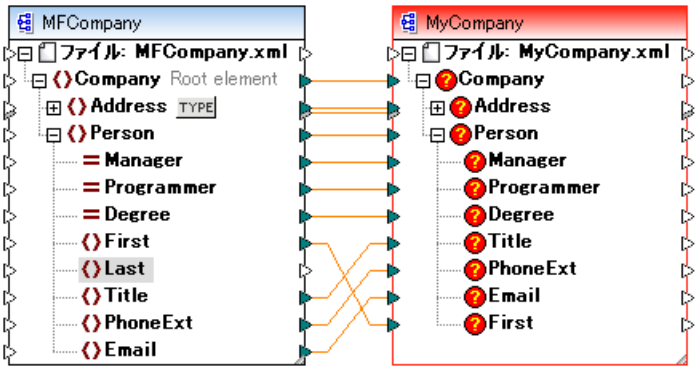
全てコピー」接続と見つからないアイテム

全てコピー接続は通常の接続と同じように扱われますが、見つからないアイテムへの接続は保持も表示もされません。

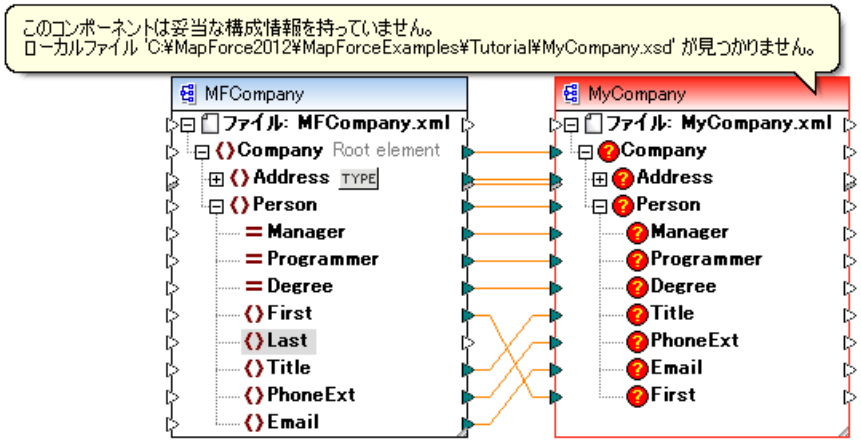


名前変更または削除されたコンポーネントソース

スキーマやコンポーネントの**データソース**が名前変更または削除された場合、コンポーネントに含まれる全てのアイテムがハイライトされます。コンポーネントの外周であるフレームにより、スキーマに対して正常な接続がなされていないことが示されます。この状態でプレビューやコード生成を行うことはできません。



マウスカーソルをハイライトされているコンポーネントへマウスオーバーすることで、ポップアップが表示されます。



ハイライトされたコンポーネントのタイトルバーをダブルクリックすることで、コンポーネント設定ダイアログボックスが表示されます。**スキーマファイル**のグループにある参照ボタンをクリックすることで別の（またはバックアップとなる）スキーマを選択することができます。詳細については*リファレンス*セクションの "コンポーネント" セクションを参照ください。

同様の構造を持ったスキーマを選択した場合全ての妥当な（正確な）接続が保持されます。

Chapter 5

マッピングのデザイン

5 マッピングのデザイン

このセクションは、データマッピングのデザイン方法とマッピングエリア上のデータの変換方法について説明します。また、マッピングデザインに関連した、注意点に関しても説明します。以下のロードマップを利用して、特定のタスクまたはコンセプトに素早くアクセスすることができます:

...を実行するには	以下を参照してください...
その他のスキーマのパス参照とマッピングで使用されるインスタンスと他のファイルの作成と編集を行います。	相対と絶対パスの使用
必要に応じてデータマッピングを微調整します。(例:ターゲットコンポーネント内のアイテムのシーケンスの調整)。	マッピング接続の種類
異なるスキーマを持つ複数のソースからのデータを単一のスキーマにマップします。	複数のスキーマからデータをマージする
コンポーネントの出力を他のコンポーネントの入力として使用します。	チェーンマッピング/パススルーコンポーネント
複数のファイルを同じマッピングにソースまたはターゲットとして処理します。(例:ディレクトリ内の全てのファイル)。	複数の入力または出力ファイルを動的に処理
(文字列パラメーターなど)外部の値をマッピングにパスします。	マッピングにパラメーターを与える
ファイルの代わりに、マッピングから文字列の値を取得します。	マッピングから文字列の値を返す
マッピングデータの一部を後の処理のために一時的に保管します(プログラミング言語内の変数に類似)。	変数の使用
昇順または降順の順序でデータを並べ替える。	データの置換え
特定の条件に従いnodes/rowsをフィルター、または値を処理します。	フィルターと条件
キーと値のペアの処理を行います。例:数値の表示を月数に変換します。(01、02、などを)テキストの表示(一月、二月、などに変換します。	Value-Maps の使用
複雑なマッピングのデザインで予期しない結果の発生を回避します。	マッピングのルールと戦略

重要な点は、MapForce は、多種の処理タスクを助ける、広範囲なビルドイン関数ライブラリを搭載している点です(関数ライブラリリファレンスを参照)。ビルドインライブラリでも、十分ではない場合、自身のカスタム関数を MapForce で作成し、外部 XSLT ファイルで再利用することができます。更に詳しい情報に関しては、関数の使用を参照してください。

5.1 相対と絶対パスの使用

マッピングデザインファイル (*.mfd) は、複数のスキーマとインスタンスファイルを参照する場合があります。MapForce により使用されるスキーマファイルは、マップされるデータの構造を決定し検証します。インスタンスファイルは他方、スキーマに対してソースデータを読み込み、プレビューし、検証することが要求されます。

マッピングにコンポーネントを追加する場合、マッピングデザインで使用されるファイルへの参照はすべて MapForce により作成されます。ですが、上記のパス参照を常に手動で設定または変更することができます。

このセクションは設定のための命令、またはマッピングにより参照されるその他のファイル型にパスの変更方法、相対と絶対パスの使用が指する意味について説明します。

5.1.1 コンポーネント上で相対パスを使用する

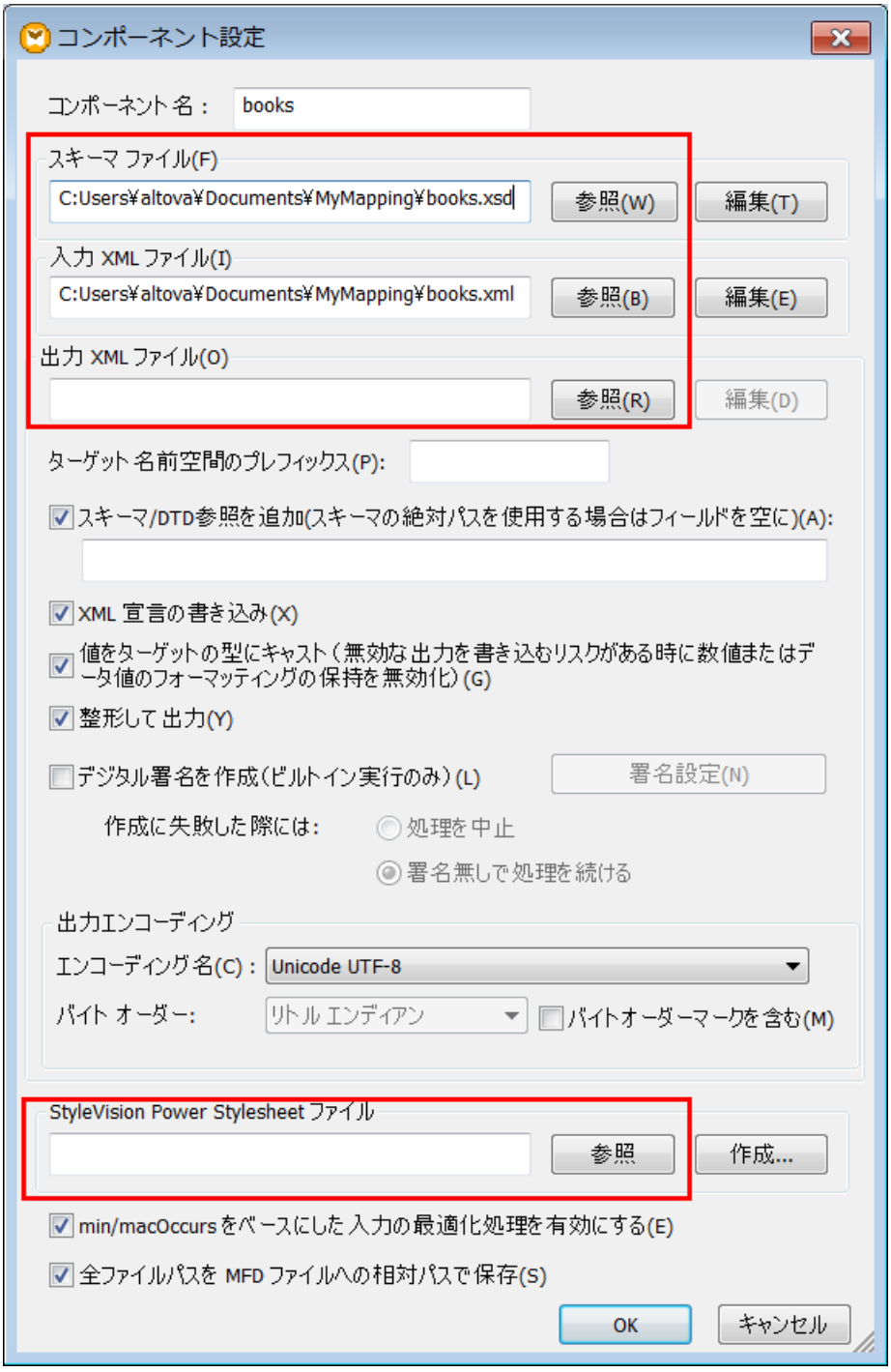
コンポーネント設定ダイアログボックス (XML コンポーネントのために下に表示されています) は、コンポーネントにより参照される可能性のある複数のファイルのための絶対または相対パスを指定するオプションを提供します:

- 入力ファイル (すなわち、MapForce がデータを読み込むファイル)
- 出力ファイル (すなわち、MapForce がデータを書き込むファイル)
- スキーマファイル (スキーマを持つコンポーネントに適用可能)
- 構造ファイル (ユーザー定義関数の入力または出力パラメーター, または変数などの複雑な構造を持つコンポーネントに適用可能)
- PDF、HTML とWord などの出力のためのデータをフォーマットするために使用される、StyleVision Power Stylesheet (*.sps) ファイル

対応するテキストボックス内で相対パスを直接入力することができます (下のイメージで赤線の枠内に表示されています)。

> 相対ファイルパスを入力する前に、マッピングファイル (.mfd) を最初に保存してください。それ以外の場合、全ての相対パスはのWindows のパーソナルアプリケーションフォルダーに対して解決されます (ドキュメント\Altova\MapForce2017) これは、予期しない振る舞いの場合があります。

.mfd ファイルマッピングに対して相対的な上記のファイルパスすべてを保存するようにMapForce に命令することができます。下のサンプルイメージでは、**「MFD ファイルに相対的なすべてのファイルパスを保存する」**オプションに注意してください。(デフォルトから奨励される オプションである) チェックボックスが有効化されていると、コンポーネントにより参照されるファイルのパスは、マッピングデザインファイル (.mfd) のパスに相対的に保存されます。これはコンポーネントに参照される全てのファイルに影響を及ぼします (イメージで赤線の枠内に表示されています)。



コンポーネント設定ダイアログボックス

上で説明されているコンポーネントはXML コンポーネンですが、 設定 **「MFD ファイルに相対的なすべてのファイルパスを保存する」** は、下のファイルでも同じように動作します：

- 複雑なユーザー定義関数の入力または出力パラメーターと複雑な型の変数により使用される構造ファイル
- 入力または出力 フラットファイル *

- XML フィールドをサポートするデータベースコンポーネントにより参照されるスキーマファイル *
- 入力または出力 XBRL、FlexText、EDI、Excel 2007+、JSON ファイル **

* MapForce Professional とEnterprise Edition
** MapForce Enterprise Edition のみ

上のコンポーネントを例として、 mfd ファイルが**books.xsd** と**books.xml** ファイルと同じフォルダーに存在する場合、パスは、以下のとおり変更されます :

C:\Users\altova\Documents\MyMapping\books.xsd は、**books.xsd** に変更されます。
C:\Users\altova\Documents\MyMapping\books.xml は、**books.xml** に変更されます。

ローカルではないドライブまたはURL を使用するパスは相対的にはなりません。

「上書き保存」メニューコマンドを使用して、マッピングを新規のフォルダーに保存する場合、チェックボックスが有効化されている場合、MapForce は、コンポーネントにより参照されるファイルを記録します。また、全てのファイルがマッピングと同じディレクトリに存在する場合、ディスク上の新しい場所にディレクトリ全体を移動してもパスの参照が壊されることはありません。

相対パスを使用することと(**「MFD ファイルに相対的なすべてのファイルパスを保存する」**チェックボックスを有効化)は多くの場合とても重要です。例 :

- オペレーティングシステム上のマッピングのロケーションが将来変更される可能性のある場合。
- (TortoiseSVN, などのバージョンコントロールシステムを使用して)マッピングがソースコントロール化にあるディレクトリに保管されている場合。
- 異なったマシンに、または異なったオペレーティングシステムにマッピングを実行のために デプロイする場合。

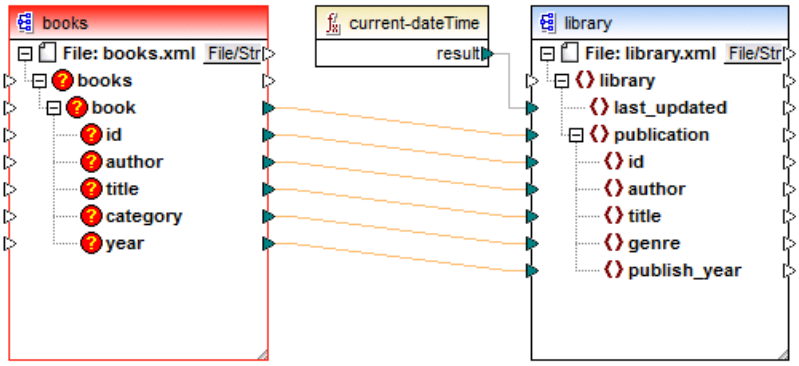
「MFD ファイルに相対的なすべてのファイルパスを保存する」チェックボックスが無効化されている場合、マッピングの保存は、ファイルパスを変更しません (すなわち、コンポーネント設定ダイアログボックス内で表示されるままになります)。

5.1.2 破損したパスの参照を修正する

マッピング内でファイルレファレンスを追加または変更する場合にパスが解決されない場合、MapForce は、警告メッセージを表示します。こうすることにより、MapForce は、破損したパスへの参照を削減します。しかしながら、以下の場合破損したパスへの参照が起こる可能性があります :

- 相対パスを使用し、マッピングファイルをスキーマとインスタンスファイルを移動することなく、新しいディレクトリに移動する場合。
- マッピングファイルと同じディレクトリ内のファイル絶対パスを使用し、ディレクトリを他の場所に移動する場合。

このような状況が発生すると、MapForce は、このコンポーネントを赤でハイライトします。例 :



破損したパスの参照

この場合の解決方法は、コンポーネントヘッダーをダブルクリックして、**「コンポーネント設定」**ダイアログボックス内で破損したパスを更新します（コンポーネント設定を変更するも参照）。

5.1.3 生成されたコード内のパスについて

コードをマッピングから生成する場合、生成されたファイルは、MapForce で実行することはできず、選択されたターゲット環境で実行できるようになります。（例：RaptorXML Server）。インスタンスファイルに対する相対ファイル名は、実行環境に対して相対となるよう解決されます。

従い、インスタンスまたはスキーマファイルに対してマッピングが相対パスを使用する場合、各ターゲット言語のベースパスに関しては以下のテーブルを考慮してください：

ターゲット言語	ベースパス
XSLT/XSLT2	XSLT ファイルのパス
XQuery*	XQuery ファイルのパス
C++、C#、Java*	生成されたアプリケーションの作業ディレクトリ
BUILT-IN* (MapForce でマッピングをプレビューする場合)	マッピング (mfd) ファイルのパス
BUILT-IN* (MapForce Server でマッピングを実行する場合)	現在の作業ディレクトリ
BUILT-IN* (FlowForce Server の管理下で、MapForce Server でマッピングを実行する場合)	ジョブの作業ディレクトリまたは FlowForce Server の作業ディレクトリ

** MapForce Professional とEnterprise エディションで使用可能な言語*

必要な場合、マッピングのためにコードを生成する際、MapForce に全てのパスを相対から絶対パスに変換するように命令することができます。このオプションは、マッピングコードを、同じオペレーティングシステム内で、または、マッピングにより使用される絶対パスへの参照がある他のオペレーティングシステムで、実行する際に役に立ちます。

生成されたコード内で全てのパスを絶対パスに変換するには、マッピング設定ダイアログボックスの**生成されたコード内でパスを絶対パスにする**チェックボックスを選択します（マッピング設定の変更を参照）。

コードを生成する際、チェックボックスが選択されていると、MapForce は、マッピングファイル (mfd) のディレクトリをベースにした相対パスを解決し、生成されたコード内でこれらを絶対パスにします。この設定は、以下のファイルのパスに影響を及ぼ

します:

- すべてのファイルベースの コンポーネント型のための入力と出力インスタンスファイル

チェックボックスが選択されていない場合、ファイルパスは、コンポーネント設定内で定義されたとおりの生成されたコードで書き込まれます。

5.1.4 コピーと貼り付けと相対パス

マッピングからコンポーネントをコピーし、貼り付ける場合は、スキーマファイルの相対パスを目的のマッピングのフォルダーに対して解決することができます。もし、パスが解決されない場合、相対パスを絶対パスにソースマッピングのフォルダーを使用して変更するようにプロンプトされます。最初に目的のマッピングに保存することが奨励されます。それ以外の場合、相対パスは、個人のアプリケーションフォルダーに対して解決されます。

5.2 マッピング接続の種類

マッピング接続を作成すると、オプションで接続 (子アイテムを持つソースとターゲットアイテム) の種類を以下から選択することができます。

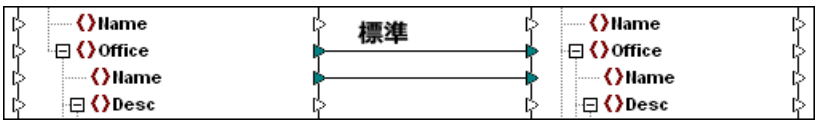
- ターゲット優先 (標準)
- ソース優先 (混合コンテンツ)
- 全てコピー (子アイテムのコピー)

接続の種類はマッピングにより生成された出力内の子アイテムのシーケンスを決定します。このセクションは各接続の種類と役に立つシナリオについての情報を提供します。

5.2.1 ターゲット優先マッピング

ターゲット優先 (標準マッピング) は、MapForce におけるデフォルトのマッピング方法で、ターゲットスキーマ内のノードのシーケンスにより出力が決定されます。このマッピングの種類は大部分のシナリオに適しており、MapForce のデフォルトのマッピングとして設定されています。

標準マッピングは実線で表示されます。



ターゲット優先マッピングは (文字データや子要素などの) 複合コンテキストを含む XML ノードをマップする場合適していない場合があります。例:

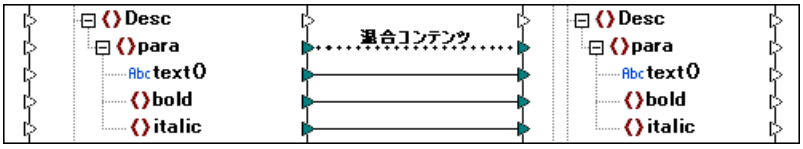
```
<p>This is our <i>best-selling</i> product.</p>
```

複合コンテンツが含まれるマッピングの場合、ソースファイル内で現れるアイテムのシーケンスを保持ため、ソース優先マッピングを使用することが奨励されます (ソース優先マッピング を参照)。

5.2.2 ソース優先マッピング

ソース優先 (混合コンテンツ) マッピングを使うことで、テキストならびに子ノードを、XML **ソース**ファイルにある順序通りにマッピングすることができます。

- 混合コンテンツテキストノードコンテンツがサポートされます。
- 子ノードの順序はソース XML インスタンスファイルに依存します。



混合コンテンツのマッピングは点線により表示されます。

勿論、ソース優先 /混合コンテンツマッピングを XML スキーマの**複合型**に適用することもできます。XML ソースファイルの順序に従って子ノードのマッピングが行われます。

ソース優先 /混合コンテンツマッピングでは以下がサポートされます：

マッピング

- **ソース**コンポーネントとして：
  - XML スキーマ複合型（mixed=true となっている混合コンテンツを含む）
- **ターゲット**コンポーネントとして：
  - XML スキーマ複合型（混合コンテンツを含む）メモ：CDATA セクションはテキストとして扱われます。

5.2.2.1 混合コンテンツのマッピング

以下のサンプルでは**...\MapForceExamples\Tutorial** フォルダー以下にある**Tut-OrgChart.mfd、Tut-OrgChart.mfd.xml、Tut-OrgChart.mfd.xsd, Tut-Person.xsd** を使用します。

ソース XML インスタンス
このセクションで使用されている**Tut-OrgChart.XML** ファイルの一部を以下に示します。ここで注目したいのは "bold" ならびに "italic" 子ノードも含まれている "para" の混合コンテンツです。

para 要素には処理命令（`<?sort alpha-ascending?>`）だけではなく、コメントテキスト（`<!--Company details... -->`）も含まれていることに注目してください。これらのデータもマッピングすることができます。

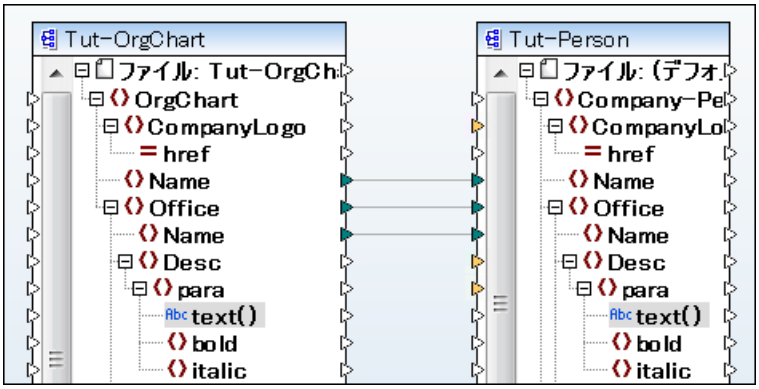
```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XMLSpy v2005 sp2 U (http://www.altova.com) by Mr. Nobody (Altova GmbH) -->
<OrgChart xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="Tut-OrgChart.xsd">
  <CompanyLogo href="nanonull.gif"/>
  <Name>Organization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

XML インスタンスファイルにあるテキストの**シーケンス**ならびにbold/italic ノードは以下のような構成（順序）となっています：

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

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

初期のマッピング
Tut-OrgChart.mfd を開いたときのマッピングを以下に示します。



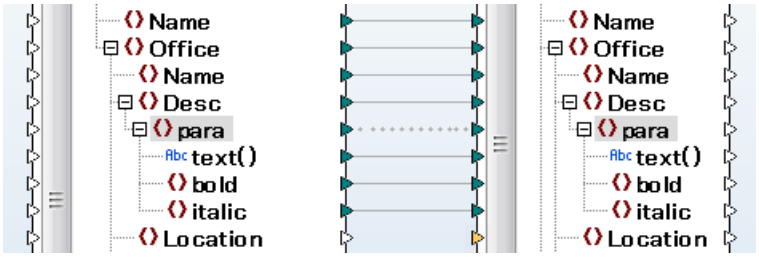
上のマッピングを出力

上に示されるマッピング結果を以下に示します。組織図 や個々のオフィス名が表示されます。

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

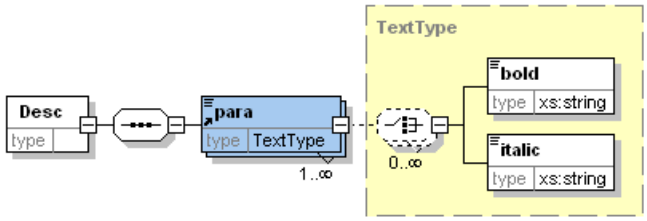
para 要素のマッピング

以下のスクリーンショットは混合コンテンツマッピングの例です。混合コンテンツの para 要素にコネクタが接続されており、**点線**によりそれが混合コンテンツであることを表しています。**text()** ノードにはテキストデータが含まれており、ターゲットコンポーネントへテキストを表示するには、このノードをマッピングする必要があります。



コネクタを右クリックしてプロパティを選択することで、コネクタに注釈を追加したり、ラベルを付与することができます。詳細については 接続の注釈 を参照ください。

以下のイメージは **Tut-OrgChart.xsd** スキーマファイルにある Desc 要素のコンテンツモデルを表しています。この例の場合、この定義はソースならびにターゲットスキーマの両方にて共通したものとなっています。



コンテンツモデルにある **para** 要素について、以下の事柄を理解できます：

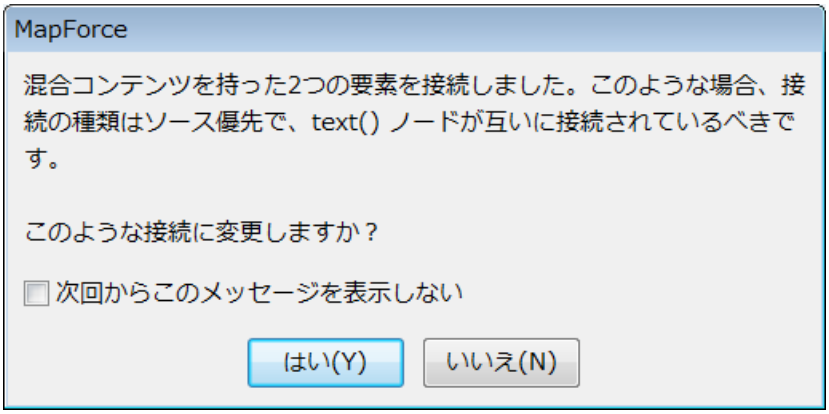
- **para** は "TextType" という型で、mixed="ture" の複合型です。

- **bold** ならびに**italic** 要素は両方とも "xs:string" 型となっており、この例では再帰的な型として定義されていません（つまり**bold** も**italic** も "textType" 型ではありません）。
- **bold** ならびに**italic** 要素は、**para** 内部においてどのような順序でも無制限の数だけ出現することができます。
- **para** 要素内では**bold** ならびに**italic** 要素とともに、任意のテキストを入力することができます。

アイテム間で混合コンテンツの接続を作成：

1. メニューオプションの「接続 | 子要素の自動接続」を選択して、有効にします（既に有効になっていない場合）。
2. ソーススキーマにて**para** アイテムから、ターゲットスキーマの**para** アイテムへの接続を行います。

   ソース優先のマッピングを行うかを尋ねるメッセージが表示されます。



3. 「はい」をクリックして混合コンテンツ接続を作成します。

メモ：
para は混合コンテンツであるため、この時点でメッセージが表示されます。自動接続オプションを無効にした状態で para アイテムを直接接続した際にも混合コンテンツのメッセージが表示されます。

para アイテムにある全ての子アイテムが接続されました。para アイテムへ接続しているコネクタが点線で表示され、それが混合コンテンツであることを表しています。

4. 出力タブをクリックして、マッピングの結果を確認します。

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

5. 出力タブのアイコンバーにある**ワードラップ**アイコン  をクリックして、全てのテキストを出力ウィンドウにて表示します。

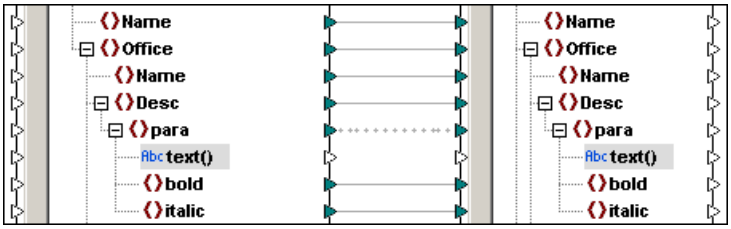
```
3     <Name>Organization Chart</Name>
4     <Office>
5       <Name>Nanonull, Inc.</Name>
6       <Desc>
7         <para>The company was established in<bold> Vereno</bold>in 1995. Nanonull develops
      nanoelectronic technologies for<italic>multi-core processors.</italic>February 1999 saw the
 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

各オフィスに関する記述である混合コンテンツテキストが正しくマッピングされました。テキストだけでなくbold ならびにitalic タグコンテンツが、XML **ソース**ファイルにある通りにマッピングされました。

6. マッピングビューに切り替えます。

混合コンテンツからテキストノードを削除する：

1. **text()** ノードのコネクタをクリックして、Del キーを押下することで接続を削除します。



2. 出力タブをクリックしてマッピングの結果を確認します。

```
5       <Name>Nanonull, Inc.</Name>
6       <Desc>
7         <para>
8           <bold> Vereno</bold>
9           <italic>multi-core processors.</italic>
10          <bold>Nano-grid.</bold>
11          <italic>offshore</italic>
12        </para>
13        <para/>
14      </Desc>
15    </Office>
16    <Office>
17      <Name>Nanonull Europe, AG</Name>
18      <Desc>
19        <para>
20          <italic>Europe</italic>
21          <bold> five research scientists </bold>
22        </para>
23      </Desc>
```

結果：

- para 要素内にある全ての**テキスト**ノードが削除されました
- それまでマッピングされていたbold ならびにitalic 要素のテキストはまだ残っています。
- bold ならびにitalic アイテムのシーケンスもソースXML ファイルと同様のものとなっています。

処理情報とコメントのマッピング:

1. コンテンツ接続を右クリックして、**プロパティ**を選択します。
2. **ソースドライブ (混合コンテンツ)**から、**処理命令をマップ**と**コメントをマップ**のチェックボックスを選択します。

5.2.2.2 混合コンテンツのサンプル

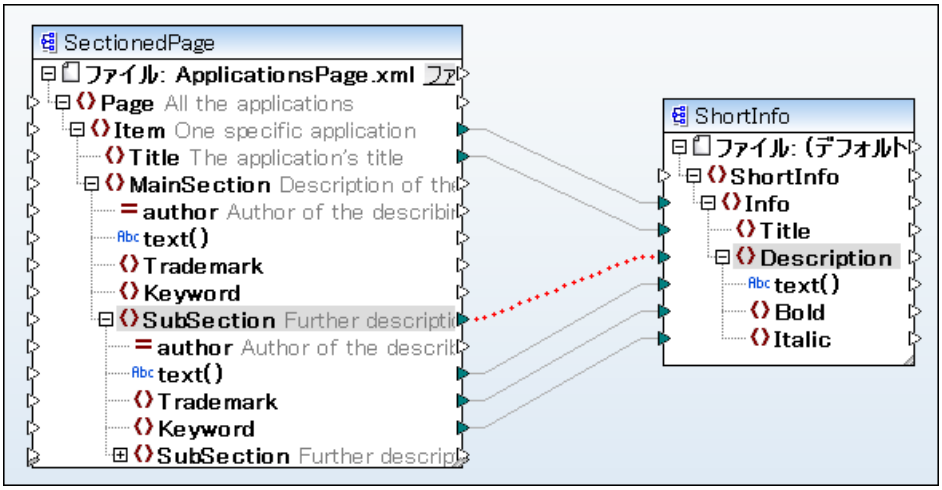
以下のサンプルでは **...\ MapForceExamples** フォルダー以下にある**ShortApplicationInfo.mfd** を使用します。

この例で使用するXML ソースファイルの一部を示します：

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

以下にマッピングを示します。以下の点に注意してください：

- "SubSection" アイテムのコネクタは混合コンテンツで、ターゲットXML/スキーマの詳細 アイテムへマッピングされています。
- text() ノード同士がマッピングされています。
- Trademark テキストがターゲットのBold アイテムへマッピングされています。
- Keyword テキストがターゲットのItalic アイテムへマッピングされています。



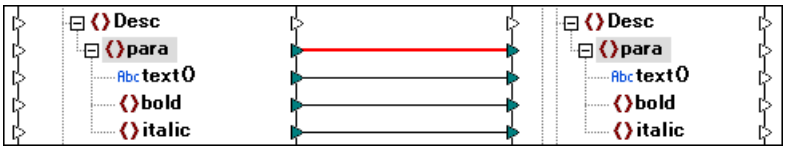
マッピング結果
各説明文の混合コンテンツテキストが正しくマッピングされ、テキストだけではなく、bold とitalic タグコンテンツがXML ソースファイルにある順序の通り、正しくマッピングされます。

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

5.2.2.3 混合コンテンツアイテムに対して標準接続を使用する

上記のように、ソース優先の（標準ではない）接続は、通常、複合コンテンツノードからデータをマッピングする際に使用されます。それ以外の場合、結果する出力は、希望する結果ではない可能性があります。通常の標準（ターゲット優先）接続を使用して、複合コンテンツノードからデータをマッピングする場合の結果を確認するには、以下の手順を踏んでください：

1. **<Documents>\Altova\MapForce2017\MapForceExamples\Tutorial** フォルダーからマッピング **Tut-OrgChart.mfd** を開きます。
2. ソース内の`para`ノードとターゲット内の`para`ノード間に接続を作成します。MapForce が接続をソース優先と定義するかを問うメッセージが表示されます。**【いいえ】**をクリックします（これにより、MapForce による提案を破棄し、通常の接続を作成します）。



メモ： 接続が上で表示されているように標準（ターゲット優先）であることを確認してください。すべてコピー接続が自動的に作成されると、接続を右クリックして、コンテキストメニューから **【ターゲット優先】（標準）**を選択します。

3. **【出力】**タブをクリックして、マッピングの結果を表示します。

```
<Office>
  <Name>Nanonull, Inc.</Name>
  <Desc>
    <para>The company was established inin 1995. Nanonull develops nanoelectronic techn
unveiling of the first prototype The company hopes to expand its operations to drive down opera
      <bold> Vereno</bold>
      <bold>Nano-grid.</bold>
      <italic>multi-core processors.</italic>
      <italic>offshore</italic>
    </para>
    <para>White papers and further information will be made available in the near future.
    </para>
  </Desc>
</Office>
<Office>
```

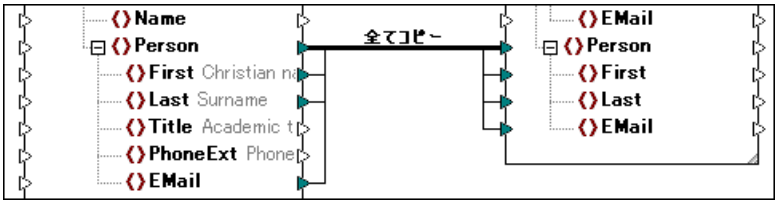
上に示されるように、混合コンテンツのアイテムに対して標準マッピングを適用すると、以下のような結果となります：

- `text()`ソースアイテムのコンテンツがターゲットにコピーされます。しかしながら、出力内の子ノード（この場合、`bold`と`italic`）のシーケンスは、ターゲット XML スキーマ内のシーケンスに対応します。すなわち、子ノード（この場合、`bold`と`italic`）は、複合コンテンツノードテキストの後に表示されます。
- それぞれの`para`要素に対して、MapForce は、`text()`ノードを最初にマップし、すべての`bold`アイテムと、最後に全ての`italic`アイテムをマップします。この結果、複数の`bold`と`italic`アイテムは、互いに積み上げられて表示されます。各アイテムのコンテンツは、接続がソースから存在する場合に表示されます。

5.2.3　全てコピー接続

全てコピーという種類の接続方法では、作業領域にあるマッピングをより単純なものとして、ソースならびにターゲットコンポーネントにある**全ての**同一アイテムを自動的に接続します。つまり、ソースならびにターゲットコンポーネントにおける型に従うかたちで、ソースならびにターゲットアイテムの**型**が**同一**の場合か、またはターゲットの型が xs:anyType の場合、該当する全てのソースアイテムがターゲットコンポーネントへ**コピー**されます。

*ソースならびにターゲットの**型**が**同一で無い**場合、かつターゲットの型がxs:anyType で無い場合、ソースデータは同一の階層にある同一名のアイテムにマッピングされます。ターゲットアイテムの名前が異なる場合、そのターゲットアイテムに対するマッピングは作成されません。*



全てコピー接続

マッピング上の2つの構造間にマッピングを描く場合シースとターゲット構造に互換性があると検出した場合、MapForce は、「全てコピー」接続を自動的に作成します（これは、両方の構造が同じ型、または、ターゲットがソース型のサブ型であることを意味します）。マッピングのランタイムでは、子を含む全てのインスタンスデータはソースからターゲットに再帰的にコピーされます。

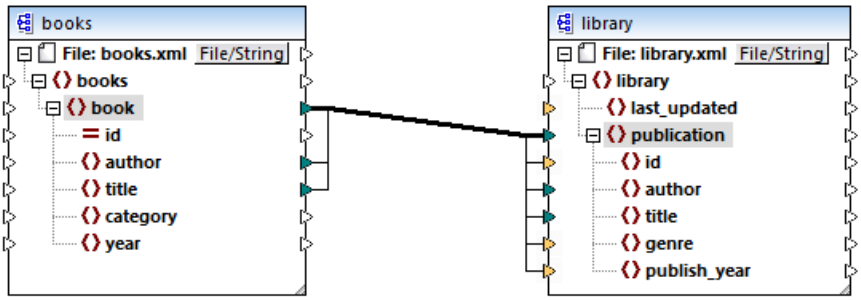
「全てコピー」接続を手動で作成するには、子アイテムを持つ同類の2つのノード間の既存の接続を右クリックし、コンテキストメニューから**全てコピー（子アイテムのコピー）**を選択します。

次に注意してください：

- 「全てコピー」接続が意味を成さない、または、サポートされていないコンテキストでは、この種類の接続を手動で作成することはできません。
- 「全てコピー」接続をXML/スキーマ コンポーネントの`root` 要素に対して作成することはできません。
- ユーザー定義関数のスキーマとパラメーター間に「全てコピー」接続を作成する場合、2つのコンポーネントは同じスキーマをベースにしている必要があります。しかしながら、同じルート要素を持つ必要はありません。

「全てコピー」接続の例は次のステップで作成することができます：

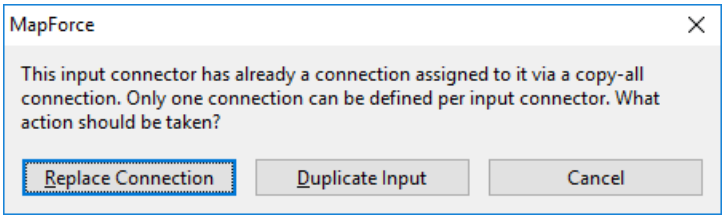
1. 新規マッピングを作成します。
2. **「挿入」**メニューから**XML スキーマ/ファイル**をクリックし、フォルダー**<Documents>\Altova \MapForce2017\MapForceExamples\Tutorial** 内にある**books.xml** ファイルを参照してください。
3. **「挿入」**メニューから**XML スキーマ/ファイル**をクリックし、フォルダー**<Documents>\Altova \MapForce2017\MapForceExamples\Tutorial** 内にある**library.xsd** ファイルを参照してください。
4. 「books」コンポーネントの`book` ノードと「library」コンポーネントの`publication` ノード間にマッピング接続を描きます。
5. 新規の接続を右クリックして、コンテキストメニューから**全てコピー（子アイテムのコピー）**を選択します。



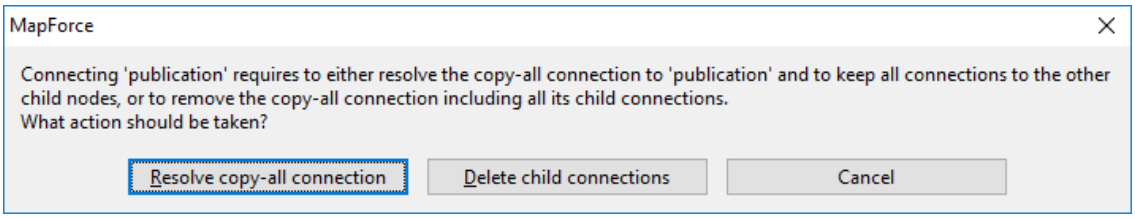
ソースとターゲット構造間に差異が存在する場合、「全てコピー」接続が列挙します。マッピングのランタイムでは、(要素と属性などの)ソースアイテムは、ターゲット型内に存在するものだけがコピーされます。これは再帰的に繰り返されます。

例えば、上のマッピングでは、2つの構造(`author` と`title`)の2つの子アイテムが同一です、ですから、ターゲットにマップすることができます。アイテム`id`は、ソース属性内の属性とターゲット内の要素であるため、自動的には含まれません。マップする必要がある場合、例えば、`category` から`genre`, へマップする場合、異なるアイテムのため、「全てコピー」接続を使用することはできません。

入力コネクタが(コンポーネントの横の小さい三角で表示されている)「全てコピー」接続を受け取る場合、他の接続を受け入れることはできません。上のサンプルでは、`category` と`genre` 間の接続を作成する場合、MapForce 置換え、または、入力の複製を行うようにプロンプトします。



ここでは必要ありませんが、入力の複製は、ターゲットが1つ以上の入力からデータを受け入れる場合に役に立ちます(次を参照:入力の複製)。「全てコピー」接続を置き換える場合、メッセージボックスは、「全てコピー」接続を解決、または、削除するようにプロンプトします。

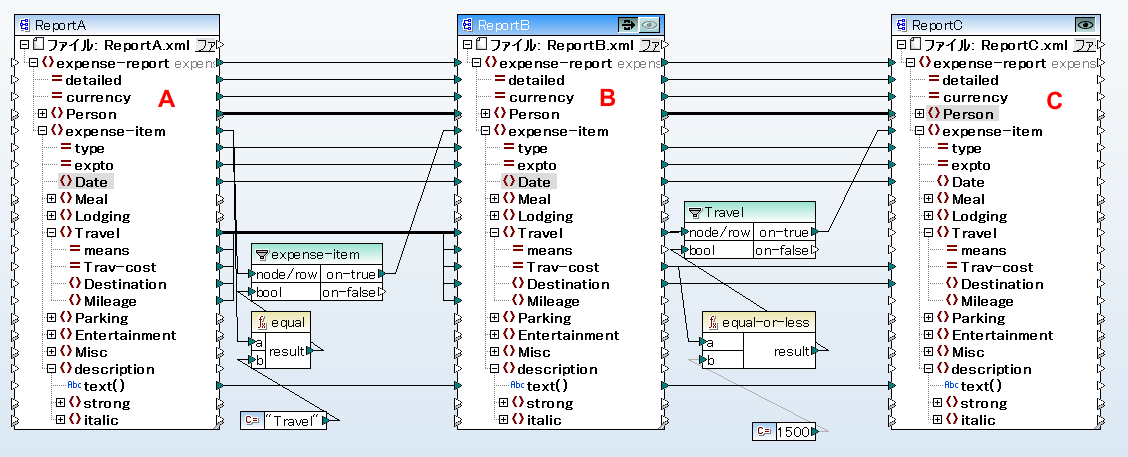


「全てコピー」接続を対応する子アイテムの標準のターゲット優先 接続と置換えるには、**全てコピー接続を解決する**をクリックします。「全てコピー」接続を削除するには、**「子接続の削除」**をクリックします。

5.3 チェーンマッピング

MapForce ではチェーン構造により複数のコンポーネントを含むようなマッピングがサポートされます。チェーン構造のマッピングとは、少なくとも 1 つのコンポーネントがソースかつターゲットコンポーネントとして機能するマッピングのことです。このようなコンポーネントの出力は、それ以降のマッピングで入力として使用されます。このようなマッピングのことを「中間」コンポーネントと呼びます。

例えば、下に示されるマッピングは、(XML 書式の)2つの段階で処理される経費報告書を表示しています。 A からB へのマッピングの部分は、 "Travel" とマークされる経費のみをフィルターします。 B からC へのマッピングの部分は、1500以下の "Travel" 経費のみをフィルターします。コンポーネントB は、入力と出力接続の両方が存在するため、"中間 "コンポーネントです。このマッピングは、次のパスで見つけることができます :**<Documents>\Altova\MapForce2017 \MapForceExamples\Tutorial\ChainedReports.mfd**。



ChainedReports.mfd

"パススルー"と呼ばれる機能がチェーンマッピングに搭載されています。"パススルー"は、出力ウィンドウ内でチェーンマッピングの各段階による出力を確認することのできるプレビュー機能です。例えば、上のマッピングでは、A からB への、および、B からC への、結果するXML 出力をプレビューして、保存することができます。

メモ: XML、CSV、TXT ファイルといったファイルをベースにした「中間」コンポーネントだけが「パススルー」機能を搭載します。データベースコンポーネントを中間コンポーネントとして使用することはできますが、パススルーコンポーネントは表示されません。プレビューやコードの生成を行うたびに、中間コンポーネントは再度生成されますが、生成を行う前に一度削除しなければならないため、データベースにてこのような操作を行うことはできません。

MapForce Server または、生成されたコードによりマッピングが実行される場合、マッピングのチェーン全体が実行されます。チェイン内の各ステップで必要な出力ファイルをマッピングは生成し、マッピングのチェーンのステップの出力が次のマッピングステップへ入力として転送されます。

中間コンポーネントのために、動的なファイル名を生成することもできます。コンポーネントが対応して構成されていることを条件に、マッピングから "File:" アイテムへの接続を受け入れることができます。更に詳しい情報に関しては、次を参照してください:複数の入力、および、出力ファイルを動的に処理する。

プレビューボタン

コンポーネントB ならびにC にはプレビューボタンが表示されます。この機能を使用することによりMapForce 内でB の中間マッピング結果のプレビューだけではなく、チェーン構造の最終結果をプレビューすることができます。対応するコンポーネントのプレビューボタンをクリックし、出力ボタンをクリックすることでマッピング結果を確認することができます。

パススルーボタンが有効化されている中間 "コンポーネントをプレビューすることはできません。同時にデータをプレビューし、パススルーすることは意味を成さないため、プレビューボタンは、自動的に無効化されます。このようなコンポーネントの出力を確認するためには、パススルーボタンをクリックして、無効化してから、プレビューボタンをクリックします。

パススルーボタン
中間コンポーネントB には、更に「パススルー」という名前のボタンがコンポーネントのタイトルバーに表示されます。

パススルーボタンが有効 になっている場合、コンポーネントA からコンポーネントB へ、そしてコンポーネントC へというかたちで全てのデータをプレビューウィンドウへマッピングします。2つの結果が生成されます。

- マッピングコンポーネントA の結果が中間コンポーネントB へマッピングされます。
- 中間コンポーネントB の結果がターゲットコンポーネントC へマッピングされます。

パススルーボタンが**無効** になっている場合、マッピングチェーンの一部だけが実行されます。生成されるマッピング結果は、(コンポーネントB とC のうち) どちらのプレビューボタンがアクティブになっているかに依存します。

- コンポーネントB のプレビューボタンがアクティブになっている場合、コンポーネントA からB へのマッピング結果が生成されます。マッピングチェーンはコンポーネントB で停止し、プレビューを行うのにコンポーネントC は全く使用されません。
- コンポーネントC のプレビューボタンがアクティブになっている場合、中間コンポーネントB からC へのマッピング結果が生成されます。パススルー機能が無効になっているため、コンポーネントB に対する自動チェーンが中断されます。マッピング右側にあるチェーンだけが実行され、コンポーネントA は使用されません。

-

"パススルー"ボタンが無効化されている場合、中間コンポーネントは、"入力 XML ファイル"と "出力 XML ファイル" フィールド内で同一のファイル名を持つ必要があります。これにより、B とC の間のマッピングの部分をプレビューする際に、出力として生成されるファイルがA とB の間のマッピングを入力と使用することができます。また、生成されたコード内で、または、MapForce Server 実行内で、マッピングのチェーンが壊れないことを保証することができます。

前記のように、マッピング MapForce Server、または、生成されたコードにより実行された場合、すべてのコンポーネントの出力が生成されます。この場合、コンポーネントB のパススルーボタン、および現在選択されているプレビューコンポーネントは破棄されます。上のマッピングを例にして、2つの結果ファイルは以下のように生成されます:

1. A からB へのマッピングコンポーネントから結果する出力ファイル
2. B からC へのマッピングコンポーネントから結果する出力ファイル

次のセクションであるサンプル:パススルーが有効な場合 とサンプル:パススルーが無効な場合 は、どのようにソースデータがパススルーボタンが有効化、または、無効化されている場合に、転送されるかを説明しています。

5.3.1 サンプル :パススルーが有効な場合

このサンプルで使用されているマッピング (**ChainedReports.mfd**) は、**<Documents>\Altova \MapForce2017\MapForceExamples\Tutorial** フォルダー内にあります。このマッピングは、旅費の経費を含む **ReportA.xml** という名前のXML ファイルを処理します。簡素化のために、名前空間宣言と`expense-item` 要素の一部は省略されています:

```
<?xml version="1.0" encoding="UTF-8"?>
<expense-report currency="USD" detailed="true">
   <Person>
      <First>Fred</First>
      <Last>Landis</Last>
      <Title>Project Manager</Title>
```

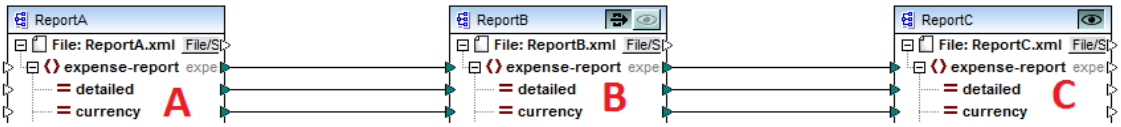
```
      <Phone>123-456-78</Phone>
      <Email>f.landis@nanonull.com</Email>
   </Person>
   <expense-item type="Travel" expto="Development">
      <Date>2003-01-02</Date>
      <Travel Trav-cost="337.88">
         <Destination/>
      </Travel>
      <description>Biz jet</description>
   </expense-item>
   <expense-item type="Lodging" expto="Sales">
      <Date>2003-01-01</Date>
      <Lodging Lodge-cost="121.2">
         <Location/>
      </Lodging>
      <description>Motel mania</description>
   </expense-item>
   <expense-item type="Travel" expto="Marketing">
      <Date>2003-02-02</Date>
      <Travel Trav-cost="2000">
         <Destination/>
      </Travel>
      <description>Hong Kong</description>
   </expense-item>
</expense-report>
```

ReportA.xml

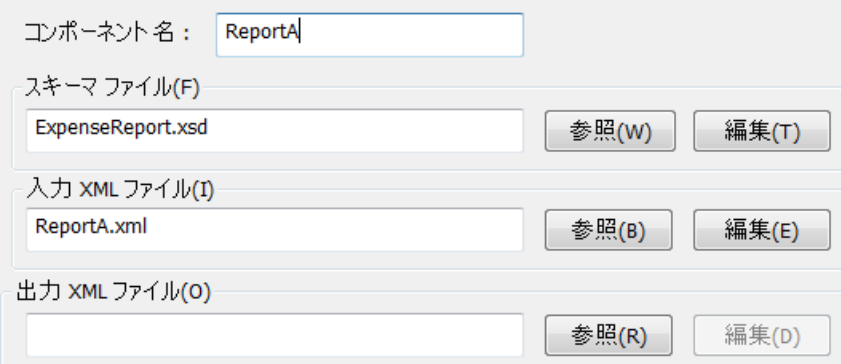
マッピングの目的は、上のファイルを基にして、2つの更なるレポートを生成することです：

- **ReportB.xml** - このレポートは、"Travel" 型の旅費のみを含みます。
- **ReportC.xml** - このレポートは、1500 を超えない "Travel" 型の旅費のみを含みます。

この目的を達成するためには、マッピング (コンポーネントB) の中間コンポーネントでは、表示されているとおりパススルーボタン が有効化されています。これによりマッピングは段階的に実行されます：A からB は、そして、B からC へ実行されます。中間コンポーネントにより作成される出力は、B とC の間のマッピングのために使用されます。



マッピングのチェーン内の各段階で生成される出力ファイルは、各コンポーネントの設定により決定されます。 (コンポーネント設定を開くには、右クリックし、コンテキストメニューから**プロパティ**を選択します)。具体的には、最初のコンポーネントは、**ReportA.xml** という名前のXML ファイルからデータを読み取るように構成されています。ソースコンポーネントであるからであり、**出力 XML ファイル** フィールドは関連性がないため、空白のままです。



ソースコンポーネントの設定

下に示されるように、2番目のコンポーネント (**ReportB**) は、**ReportB.xml** という名前の出力ファイルを作成するように構成されています。**入力 XML ファイル** フィールドは、灰色で表示されています。(このサンプル内で示されているように) パススルーが有効化されている場合、中間コンポーネントの **入力 XML ファイル** フィールドは、自動的に無効化されます。マッピングを実行するために入力ファイル名が存在する必要はなく、マッピング内のこの段階で生成される出力は、一時的なファイル内に保管され、マッピングの後の段階で再利用することができます。また、(下で示されるように) **出力 XML ファイル** が定義されている場合、中間出力ファイルのファイル名のために使用されます。**出力 XML ファイル** が定義されていない場合、デフォルトのファイル名が自動的に使用されます。

中間コンポーネントの設定

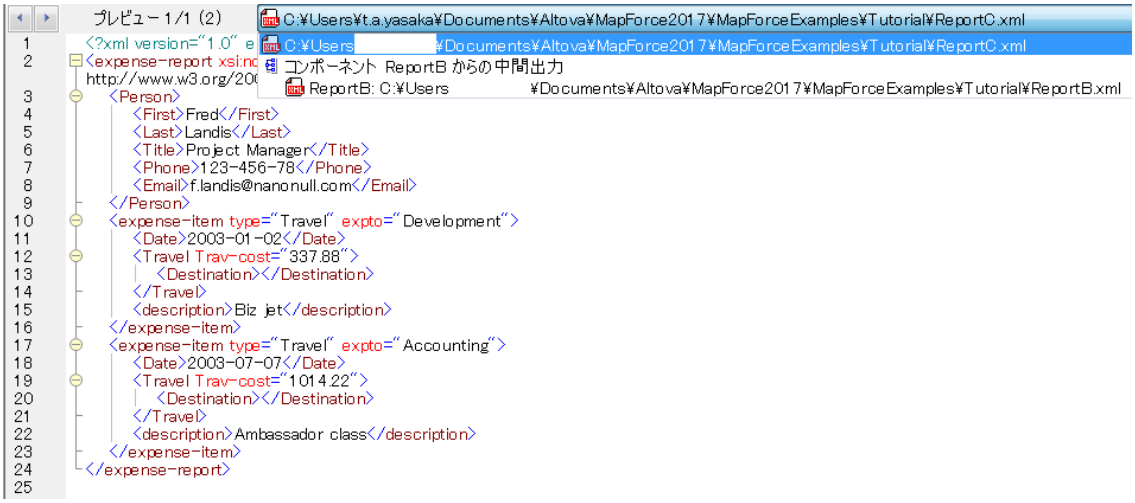
最後に、3番目のコンポーネントは、**ReportC.xml** という名前の出力ファイルを生成するように構成されています。これはターゲットコンポーネントであるため、**入力 XML ファイルフィールド** には関連性はありません。

ターゲット コンポーネントの設定

マッピングウィンドウ内の**出力**タブをクリックして、マッピングをプレビューする場合は、2つのファイルが出力に表示されます：

1. **ReportB.xml** は、A からB へのマッピングの結果を表します。
2. **ReportC.xml** は、B からC へのマッピングの結果を表します。

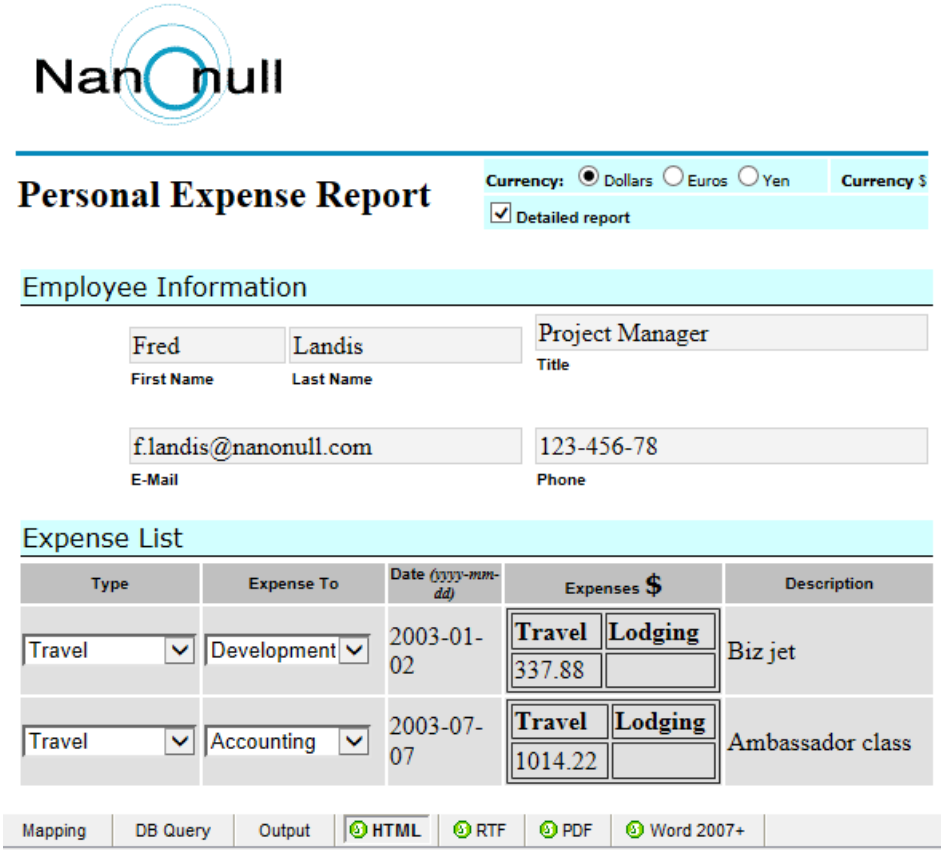
2件の生成された出力ファイルから選択するには、出力ウィンドウ内に表示されているファイルを矢印ボタンをクリックして、または、ドロップダウンリストからエントリを選択します。



生成された出力ファイル

MapForce によりマッピングが実行される場合、(**ツール| オプション| 一般**から構成することのできる) 設定「出力ファイルに直接書き込む」により、中間ファイルが一時的なファイル、または、物理的ファイルとして保存されるかを決定します。MapForce 内でマッピングが直接プレビューされる場合のみ、これは妥当であることに注意してください。このマッピングが MapForce Server により、または、生成されたコードにより実行された場合、実際のファイルは、マッピングのチェーンの各段階で生成されます。

StyleVision がインストールされている場合、そして、StyleVision Power Stylesheet (SPS) ファイルがターゲットコンポーネントに (このサンプル内で示されているように) 割り当てられている場合、最後のマッピング出力をHTML、RTF ファイルとして確認 (保存) することができます。MapForce 内でこの出力を表示するには、対応する名前を持つタブをクリックしてください。

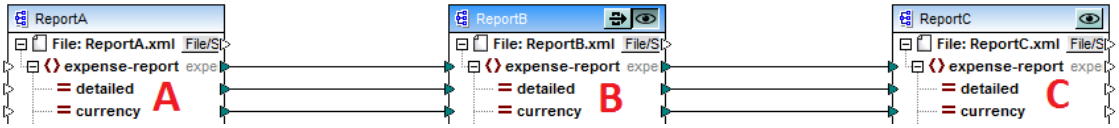


生成されたHTML 出力

マッピングのチェーンの最後のターゲットコンポーネントの出力のみが表示されることに注意してください。中間コンポーネントのStyleVision 出力を表示するには、パススルー ボタンを無効化して中間コンポーネントを (サンプル：パススルーが無効な場合 内で表示されているとおり)プレビューする必要があります。

5.3.2 サンプル :パススルーが無効な場合

このサンプル (**ChainedReports.mfd**) で使用されているマッピングは、**<Documents>\Altova \MapForce2017\MapForceExamples\Tutorial** フォルダー内で見つけることができます。このサンプルは、中間コンポーネントのパススルーボタン が無効化されている場合、出力がどのように生成されるかを説明しています。



サンプル：パススルーが有効な場合 で説明されているように、マッピングの目的は、2件の個別のレポートを生成することです。前のサンプルで、パススルー ボタン は有効化されており、両方のレポートは期待されるとおり生成され、**出力** タブ内で確認することができました。しかしながら、レポートの1件 (**ReportB.xml** または**ReportC.xml**) のみをプレビューする場合、パススルー ボタン ( ) は無効化されている必要があります。更に具体的には、パススルー ボタンを無効化することは、次の目的を達成するためには有効な場合があります：

- B からC へのマッピングを無視して、A からB により生成された部分をプレビューする。

- A からB へのマッピングを無視して、B からCにより生成された部分をプレビューする。

上に示されるようにパススルーボタンを無効化すると、**ReportB** または**ReportC** を選択することができます (両方に 👁 ボタンが表示されます)。

パススルーボタンの無効化により、中間コンポーネントがどの入力ファイルを読み取るかを選択することができます。多くの場合、これは**出力 XML ファイル** フィールド (このサンプル内で示されているように)内で定義されたファイルと同じです。

中間コンポーネントの設定

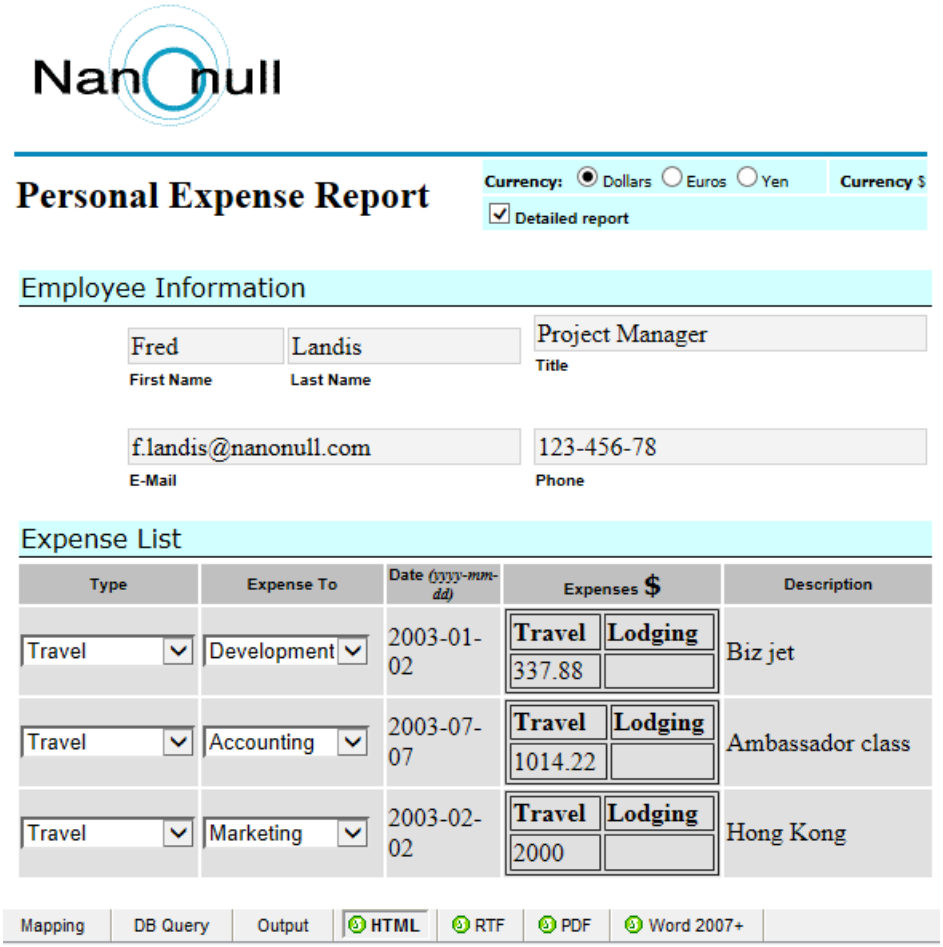
中間コンポーネント上に同じ入力と出力 ファイルが存在することは、マッピングからコードを生成する場合、または、マッピングをMapForce Server 上で実行する場合、特に重要です。前記のように、この環境は、マッピングチェーン内の各コンポーネントにより作成される出力により生成されます。ですから、中間コンポーネントが処理するファイル (この場合は **ReportB.xml**) を受け取り、異なるファイルを検索するよりも、同じファイルを次のマッピングに送ることは意味があります。(パススルーボタンが無効化されており)中間コンポーネント上で同じ入力と出力ファイル名が存在しないと、生成されたコード内で、または、MapForce Server 実行中に「システムが指定されたファイルを見つけることができません」などのエラーが発生する場合があります。

3番目のコンポーネント (**ReportC**) のプレビューボタン 👁 をクリックし、MapForce 内でマッピングをプレビューしようと試みると、エラーが発生します。上記の設定に従い、**ReportB.xml** という名前のファイルが入力として期待されるため、これは予想されている振る舞いです。しかしながら、マッピングは、(パススルーボタンが有効化されておらず、B からC へのマッピングの部分のみが実行されているため)まだこの名前のファイルは生成されていません。この問題は簡単に解決することができます：

1. 中間コンポーネントのプレビューボタンをクリックします。
2. マッピングをプレビューするために、**出力** タブをクリックします。
3. マッピング (<**Documents**>**\Altova\MapForce2017\MapForceExamples\Tutorial**) と同じフォルダー内に結果出力ファイルを**ReportB.xml** として保存します。

3番目のコンポーネント (**ReportC**) のプレビューボタンをクリックすると、エラーは表示されなくなります。

パススルーボタンが無効化されている場合、関連したStyleVision Power StyleSheet (SPS) ファイルを持つ各コンポーネントのためのStyleVision-により生成された出力をプレビューすることができます。特に、(最後のレポートに加え)中間 レポートのHTML バージョンを確認することもできます ：



NanoNull

Personal Expense Report

Currency: (•) Dollars ( ) Euros ( ) Yen Currency $

[x] Detailed report

Employee Information

First Name	Last Name	Title
Fred	Landis	Project Manager

E-Mail	Phone
f.landis@nanonull.com	123-456-78

Expense List

Type	Expense To	Date (yyyy-mm-dd)	Expenses $		Description
			Travel	Lodging	
Travel	Development	2003-01-02	337.88		Biz jet
Travel	Accounting	2003-07-07	1014.22		Ambassador class
Travel	Marketing	2003-02-02	2000		Hong Kong

Mapping | DB Query | Output | HTML | RTF | PDF | Word 2007+

中間コンポーネントのHTML 出力

5.4 複数の入力または出力ファイルを動的に処理

マッピングが実行される際、MapForce に複数のファイルを処理するように構成することができます（例：ディレクトリ内の全てのファイル）。この機能を使用して、以下のようなタスクを解決することができます：

- マッピングに処理する入力ファイルのリストを与えます。
- 単一の出力ファイルの代わりにファイルのリストのマッピング出力を生成します。
- 入力と出力ファイル名がランタイムで定義されている箇所のマッピングアプリケーションを生成します。
- ファイルのセットを他のフォーマットに変換します。
- 大きなファイルを小さなパーツに分割します。
- 複数のファイルを1つの大きなファイルにマージする

MapForce コンポーネントに複数のファイルを処理するように以下の方法で構成することができます：

- コンポーネント設定で必要とされる入力または出力ファイルに固定されたファイル名の代わりにワイルドカード文字を使用して、パスを与えます。（コンポーネント設定を変更するを参照）。すなわち、ワールドカード *と？をコンポーネント設定ダイアログボックスに入力し、MapForce は、マッピング実行される際に対応するパスを解決します。
- パスを動的に提供する、コンポーネント シーケンスルートノードに接続します。（例：`replace-fileext` 関数の結果）。マッピングが実行されると、MapForce は、全ての入力ファイルを動的に読み込み、また全ての出力ファイルを動的に生成します。

目的により異なりますが、同じマッピングで1つまたは両方のアプローチを取ることができます。しかしながら、同じコンポーネント上で同時に両方のアプローチを使用することは、あまり意味をなしません。特定のコンポーネントのために、MapForce がどのアプローチを使用するかを命令するには、コンポーネントのルートノードの横の **「ファイル」**（File）または **「ファイル文字列」**（File/String）ボタンをクリックします。このボタンにより以下の振る舞いを指定することができます：

コンポーネント設定からファイル名を使用する	コンポーネントが1つまたは複数のインスタンスファイルを処理する場合、このオプションは MapForce にコンポーネント設定ダイアログボックスで定義されたファイルを処理するように命令します。 このオプションを選択すると、ルートノードには入力コネクタが与えられません。  入力または出力ファイルin コンポーネント設定ダイアログボックス内でまだ入力または出力ファイルを指定していない場合、ルートノードの名前は **ファイル: (デフォルト)** です。それ以外の場合、ルートノードは、出力ファイル名、セミコロン（;）が後についた入力ファイル名を表示します。 入力ファイル名が出力ファイルと同じ場合、ルートノードの名前として表示されます。

	 *このオプションまたは、「マッピングから与えられた動的なファイル名を使用する」オプションを選択することができます。*
マッピングから与えられた動的なファイル名を使用する	このオプションは、コンポーネントのルートノードの値に接続することにより、マッピングエリアで定義するファイル名を処理するようにMapForce に命令します。 このオプションを選択すると、ルートノードがマッピングの実行中動的に処理されるファイル名を与える入力コネクタの値に接続します。ファイル名 in コンポーネント設定ダイアログボックス内に定義されたファイル名を持つ場合、これらの値は無視されます。 このオプションが選択された場合、ルートノードの名前は以下として表示されます **File: <dynamic>**。  *コンポーネント設定からファイル名を使用するオプションと共にこのオプションは相互排他的です。*

複数の入力または出力ファイルを以下のコンポーネントのために定義することができます：

- XML ファイル
- テキストファイル (CSV*, FLF* ファイルとFlexText** ファイル)
- EDI ドキュメント**
- Excel スプレッドシート**
- XBRL ドキュメント**

** MapForce Professional Edition 必須*
*** MapForce Enterprise Edition 必須*

以下のテーブルは、 MapForce 言語内での動的な入力と出力ファイルおよびワイルドカードへのサポートについて説明されています。

ターゲット言語	動的入力ファイル名	入力ファイル名のためにサポートされているワイルドカード	動的な出力ファイル名
XSLT 1.0	*	XSLT 1.0 によりサポートされていません	XSLT 1.0 によりサポートされていません

XSLT 2.0	*	*(1)	*
C++	*	*	*
C#	*	*	*
Java	*	*	*
BUILT-IN	*	*	*

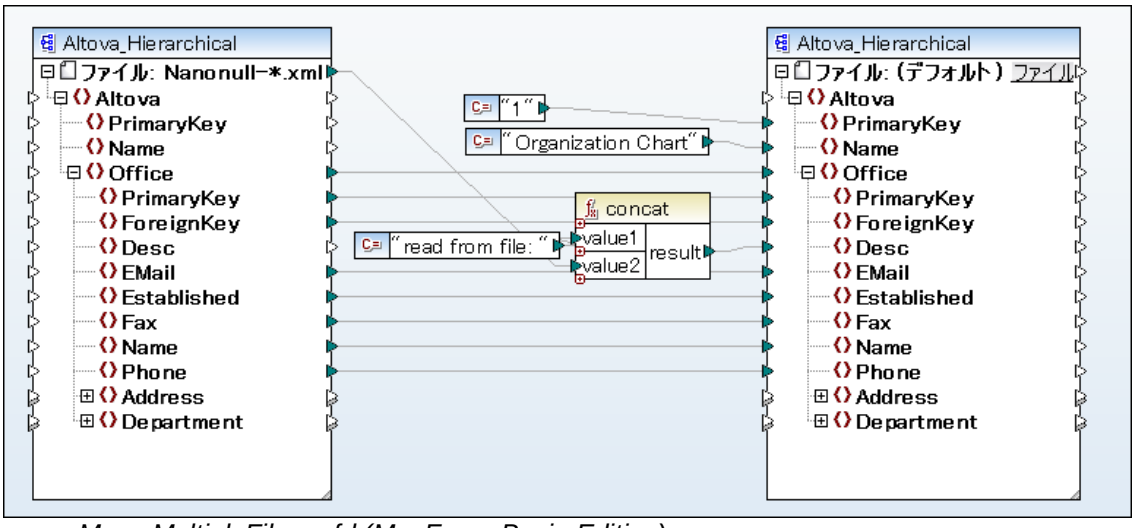
レジェンド:

*	サポートされている
(1)	`fn:collection` 関数を使用します。XSLT 2.0 とXQuery エンジンの実装は、ワイルドカードを解決します。他のエンジンは異なる場合があります。RaptorXML Server エンジンを使用してのXSLT 1.0/2.0 コード変換方法に関しては、 XSLT 1.0 および 2.0 コードの生成 を参照してください。

5.4.1 複数の入力ファイルを単一の出力ファイルにマップする

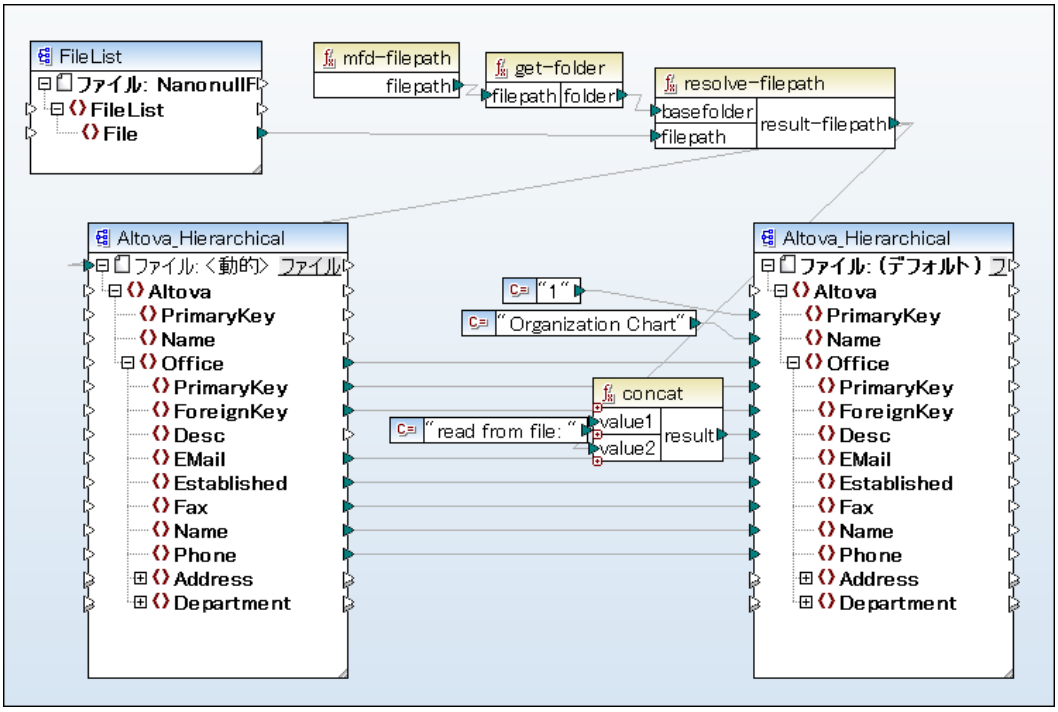
複数の入力ファイルを処理するには、以下を行います:

- コンポーネント設定ダイアログボックスに入力ファイルとしてワイルドカード(*または ?)付きのファイルパスを入力します。一致する全てのファイルが処理されます。下のサンプルは、入力 XML ファイルフィールド内でマッピング入力として名前が "Nanonull-" で開始する全てのファイルを提供する* ワイルドカード文字を使用します。ターゲットコンポーネントに動的コネクタが存在せず、ソースコンポーネントはワイルドカード* を使用して複数のファイルにアクセスしているため、複数の入力ファイルは**単一**の出力ファイルとしてマージされます。ターゲットコンポーネント内のルートノードの名前は**File: <default>** です。これはコンポーネント設定ダイアログボックス内で出力ファイルパスが定義されていないことを示します。複数のソースファイルは、ですから、ターゲットドキュメント内で追加されます。



MergeMultipleFiles.mfd (MapForce Basic Edition)

- ソースコンポーネントのファイルノードに文字列のシーケンスをマップします。シーケンス内の各文字列が1つのファイル名を表します。文字列は自動的に解決されるワイルドカードを含むこともできます。XML ファイルなどのコンポーネントがファイル名のシーケンスを提供します。

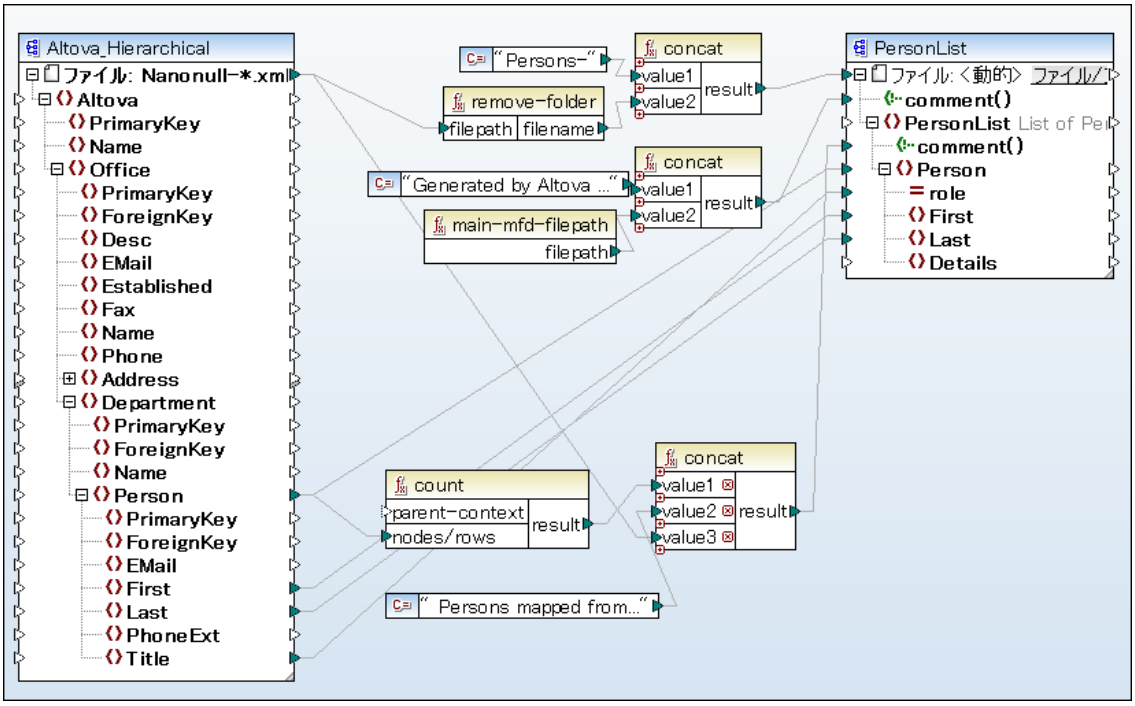


MergeMultipleFiles_List.mfd (MapForce Basic Edition)

5.4.2 複数の入力ファイルを複数の出力ファイルにマッピングする

複数のファイルを複数の ターゲットファイルにマップするには、一意の出力ファイル名を生成する必要があります。出力ファイル名が入力データ内の文字列から派生する場合があり、他の場合、入力ファイル名から出力ファイル名を派生させることが役にたつ場合があります。例：ファイル拡張子の変更により。

以下の マッピングでは、**concat** 関数を使用して プレフィックス "Persons-" を追加し、出力ファイル名を入力ファイル名から派生させます。



MultipleInputToMultiple_出力Files.mfd (MapForce Basic Edition)

メモ: 処理関数を使用することなく、入力と出力ルートノードを直接接続することを回避します。これを行うことによりマッピングが実行される際、入力ファイルが上書きされます。上記の`concat`関数などを使用して、出力ファイル名を変更することができます。

メニューオプション **「ファイル| マッピング設定」** によりマッピングで使用されるファイルパス設定をグローバルに定義することができます（マッピング設定の変更を参照）。

5.4.3 ファイル名をマッピングパラメーターとして提供する

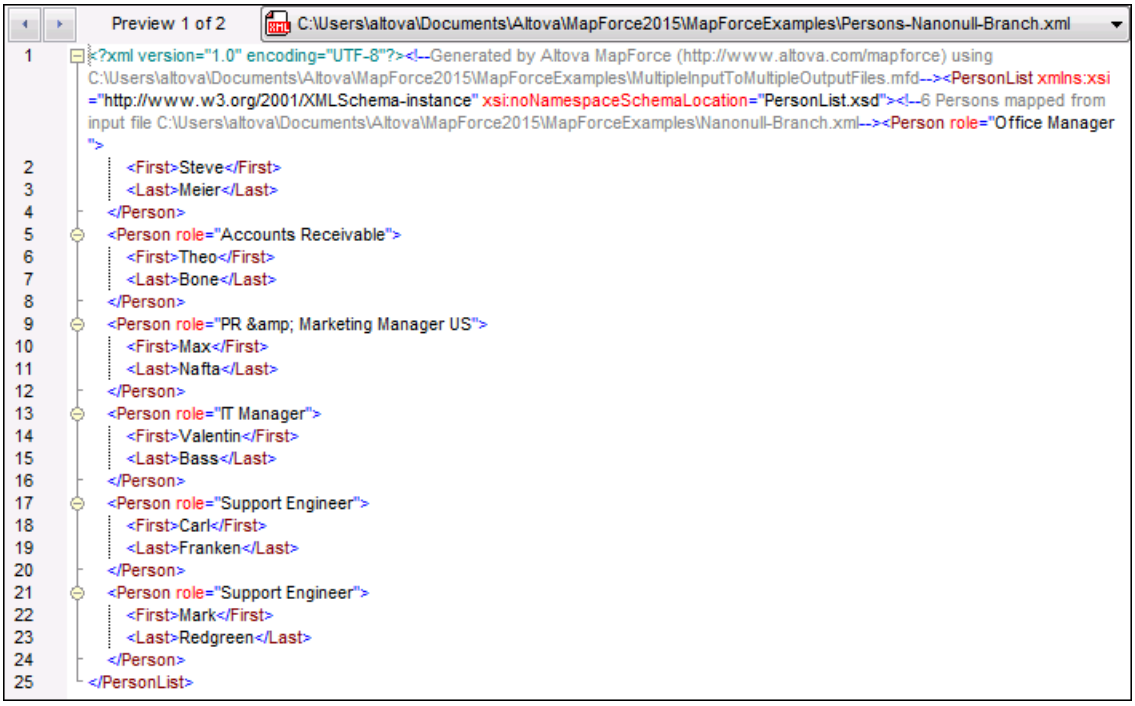
マッピングにカスタムファイル名を入力パラメーターとして提供するには、以下を行います：

1. 入力コンポーネントをマッピングに追加する（**「関数」** メニューから **「入力の挿入」** をクリックします）。コンポーネントについての更に詳しい情報は、単純型入力を参照。
1. ソースコンポーネントの **「ファイル」**（File）または **「ファイル/文字列」**（File/String）ボタンをクリックし、 **「マッピングから与えられた動的なファイル名を使用する」** を選択します
2. マッピングソースとして振舞うコンポーネントのルートノードに入力パラメーターを接続します。

成功例は例：名前をマッピングパラメーターとして使用するを参照してください）。

5.4.4 複数の出力ファイルをプレビューする

プレビューウィンドウにマッピングの結果を表示するために、「出力」タブをクリックします。マッピングが複数の出力ファイルを作成する場合、各ファイルは、「出力」タブ内でペインごとに番号が付けられます。矢印をクリックして、それぞれの出力ファイルを確認してください。



MultipleInputToMultipleOutputFiles.mfd

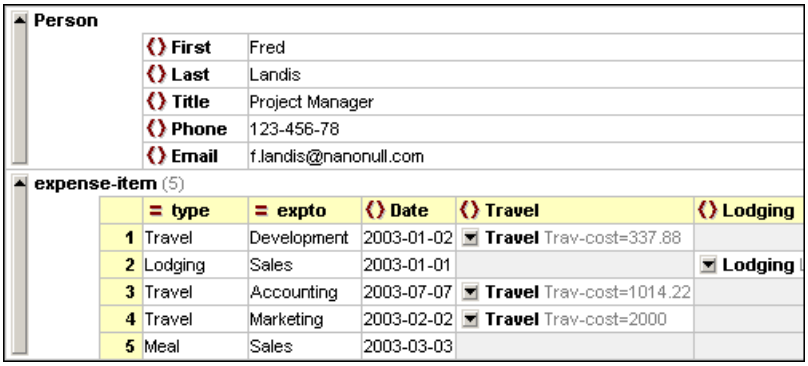
生成された出力ファイルを保存するためには以下を行います：

- **出力**」メニューから**Save 「すべての出力ファイルを保存」**（ ）をクリックします。
- **「すべての生成された出力を保存」**（ ）ツールバーボタンをクリックします。

5.4.5 サンプル：1つの XML ファイルを複数のファイルに分割

このサンプルは、単一のソースXML ファイルから動的に複数のXML ファイルを作成する方法を説明します。このサンプルに使用されるマッピングは以下で検索することができます：**<Documents>\Altova\MapForce2017 \MapForceExamples\Tutorial\Tut-ExpReport-dyn.mfd**。

ソースXML ファイル（マッピングと同じフォルダーで見つけることのできる）は、"Fred Landis" という個人の経費レポートから構成されており、5つの異なる種類の経費アイテムを含んでいます。このサンプルの目的は個々の経費アイテムのために XML ファイルを生成することです。



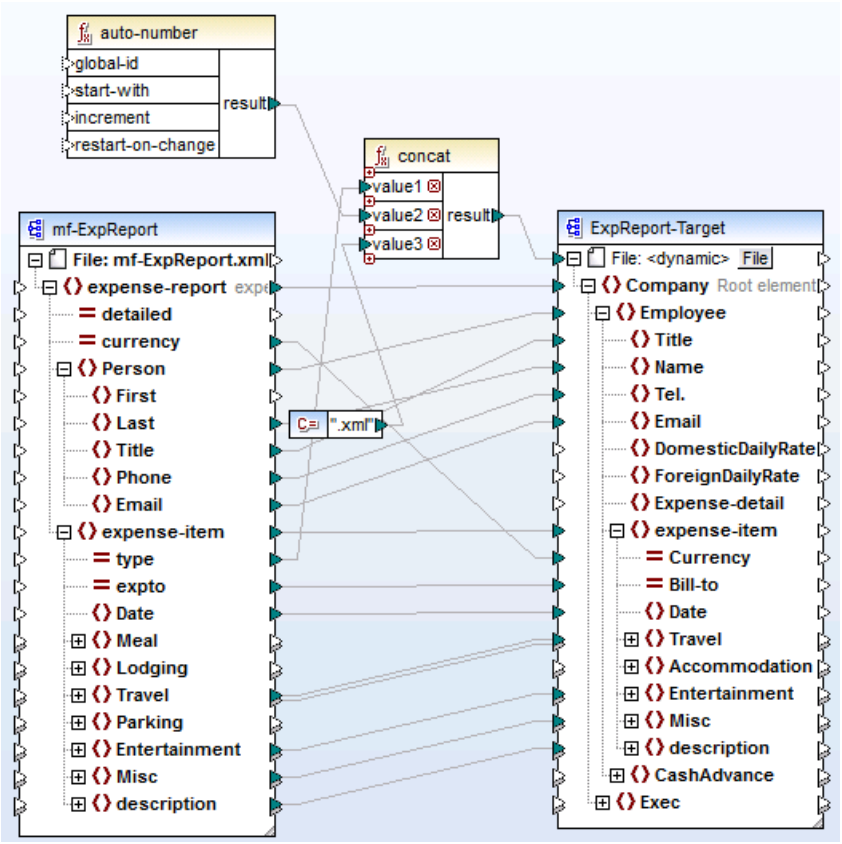
Person		
	First	Fred
	Last	Landis
	Title	Project Manager
	Phone	123-456-78
	Email	f.landis@nanonull.com

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 (XMLSpy グリッドビューにて表示)

`type` 属性は特定の経費アイテムの型を定義し、これはソースファイルを分割するために使用されるアイテムの型です。このエクササイズのゴールを達成するには、以下を行います：

1. (『ライブラリ』ペインの **『core | string 関数』** ライブラリからドラッグして)`concat` 関数を挿入します。
2. 定数を入力するには、(**『挿入』** メニューから **『定数』** をクリックして) "xml" を値として入力します。
3. (『ライブラリ』ペインの **『core | generator 関数』** からドラッグして)`auto-number` 関数を入力します。
1. **『ファイル』** (File) または ターゲットコンポーネントの **『ファイル文字列』** (File/String) ボタンをクリックし、 **『マッピングから与えられた動的なファイル名を使用する』** を選択します
4. 下に表示されるとおり接続を作成し、 **『出力』** タブをクリックしマッピングの結果を確認します。



Tut-ExpReport-dyn.mfd (MapForce Basic Edition)

結果的出力ファイルは、以下のとおり動的に名前が付けられます：

- `type` 属性は、ファイル名の最初の部分を提供します。（例："Travel"）。
- **自動連番** 関数は、ファイルの連番を提供します。（例："Travel1"、"Travel2"、など）。
- 定数は "xml" であるファイル拡張子を提供します。このため "Travel1.xml" は最初のファイル名です。

5.5 マッピングにパラメーターを与える

単純型入力コンポーネントを介して、単純型の値をマッピングにパスすることができます。マッピングエリアでは、単純型入力コンポーネントは、この結果、アイテムとシーケンスの構造の代わりに、単純型のデータ型を持つソースコンポーネントの役割を果たします(例:文字列、整数など)。ファイルベースのソースコンポーネントの代わりにまたは追加して単純型入力コンポーネントを作成することができます。生成されたファイル内では、単純型入力コンポーネントは、スタイルシートのパラメーターに対応します。

それぞれの単純型入力コンポーネント(またはパラメーター)をオプションまたは必須として作成することができます(入力コンポーネント設定を参照)。必要であれば、マッピング入力パラメーターのためにデフォルトの値を作成することができます(デフォルトの入力値を作成するを参照)。これによりマッピングの実行時にパラメーターの値を明示的に提供しない場合でも、マッピングを安全に実行することができます。

マッピングエリアに追加された入力パラメーターをユーザー定義関数内の入力パラメーターと混同しないように注意してください。(ユーザー定義関数を参照)。類似点と相違点は、以下のとおりです。

マッピング上の入力パラメーター	ユーザー定義関数の入力パラメーター
**関数	入力の挿入**」メニューから追加する。	**関数	入力の挿入**」メニューから追加する。
単純型データ型を持つことができます(文字列、整数など)。	単純型と複雑型のデータ型を持つことができます
マッピング全体に適用可能。	定義された方法で関数のコンテキストに適用することができます。

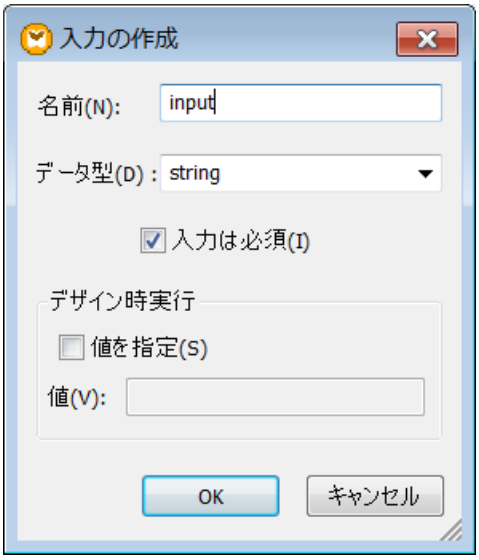
逆のマッピングを作成する場合、(メニューコマンド**「ツール | 逆順マッピングの作成」**を使用すると単純型入力コンポーネントは、単純型出力コンポーネントになります。

サンプル:例:名前をマッピングパラメーターとして使用するを参照。

5.5.1 単純型入力コンポーネントの追加

マッピングに単純型入力コンポーネントを追加する:

1. マッピングウィンドウに(ユーザー定義関数ではない)メインマッピングが表示されていることを確認してください。
2. **「関数」**メニューから**「入力」**をクリックします。
3. 名前を入力して、この入力に必要なデータ型を選択します。入力が必須のマッピングパラメーターとして扱われる必要がある場合、**「入力は必須」**チェックボックスを選択します。設定の完全なリストは、単純型入力コンポーネント設定を参照してください。
4. **「OK」**をクリックします。

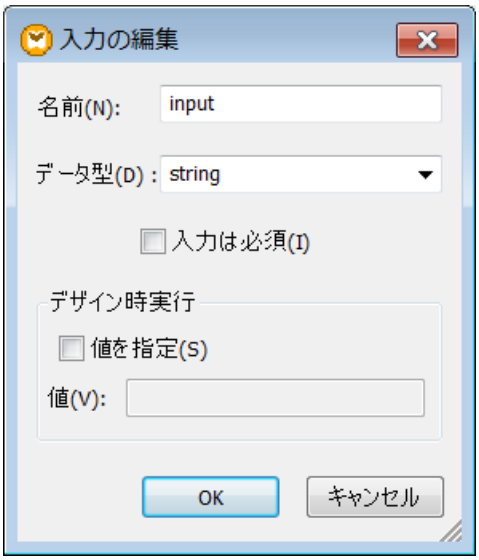


入力の作成 ダイアログボックス

ここで定義されている設定は後に変更することができます（単純型入力コンポーネント設定を参照）。

5.5.2 単純型入力コンポーネント設定

単純型入力コンポーネントに適用することのできる設定をマッピングエリアに追加する際に定義することができます。後に設定を「入力の編集」ダイアログボックスから変更することも可能です。



入力の編集 ダイアログボックス

「入力の編集」ダイアログボックスを開くには、以下を行います：

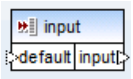
- コンポーネントを選択して、**「コンポーネント」**メニューから**「プロパティ」**をクリックします。
- コンポーネントをダブルクリックします。
- コンポーネントを右クリックして、**「プロパティ」**をクリックします。

使用することのできる設定は、以下のとおりです：

名前	このコンポーネントに対応する入力パラメーターのための詳細名を入力します。マッピングの実行時、このテキストボックスに入力された値がマッピングに与えられるパラメーターの名前になります。ですから、スペースまたは特別な文字は許可されていません。
データ型	デフォルトでは、全ての入力パラメーターは文字列データ型として扱われます。パラメーターが異なる型を持つ場合、リストからそれぞれの値を選択します。マッピングが実行された場合、MapForce は、入力パラメーターをここで選択されたデータ型にキャストします。
入力は必須	有効化されている場合、この設定は、入力パラメーターを必須にします（すなわち、パラメーターの値が提供されない限り、マッピングを実行することはできません）。 入力パラメーターのためのデフォルトの値を指定するには、このチェックボックスを無効化にします（デフォルトの入力値を作成するを参照）。
値の指定	この設定は、デザイン時にマッピングを実行する場合のみ適用することができます。**「プレビュー」**タブをクリックすることにより、コンポーネント内に直接マッピング入力として使用される値を入力することができます。
値	この設定は、デザイン時にマッピングを実行する場合のみ適用することができます。**「プレビュー」**タブをクリックすることにより、マッピング入力として MapForce で使用される値を入力するには、値の指定チェックボックスを選択し、必要な値を入力します。

5.5.3 デフォルトの入力値を作成する

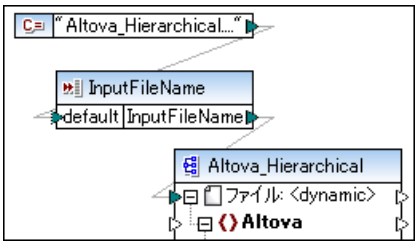
入力コンポーネントをマッピングエリアに追加すると、**default** アイテムがコンポーネントの左側に表示されます。



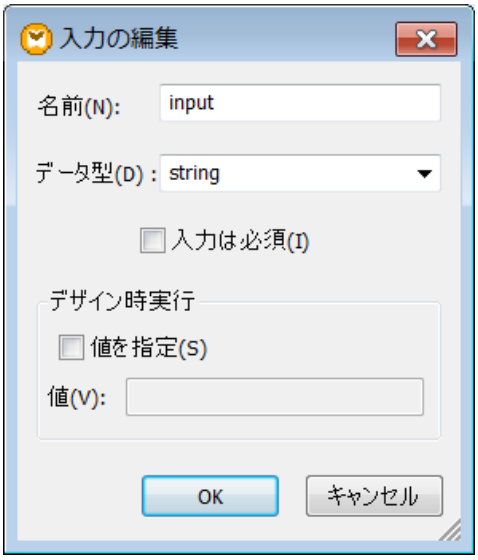
単純型入力コンポーネント

デフォルトアイテムは任意のデフォルトの値のこの入力コンポーネントへの接続を以下のように有効化します：

1. 定数コンポーネントを追加し、（**「挿入」**メニューから**「定数」**をクリックします。）入力コンポーネントの**default** アイテムに接続します。



2. 入力コンポーネントをダブルクリックして、**「入力は必須」**チェックボックスのチェックを外します。デフォルトの入力の値を作成すると、この設定は重要になり、マッピングの検証に関する警告を引き起こす可能性があります。



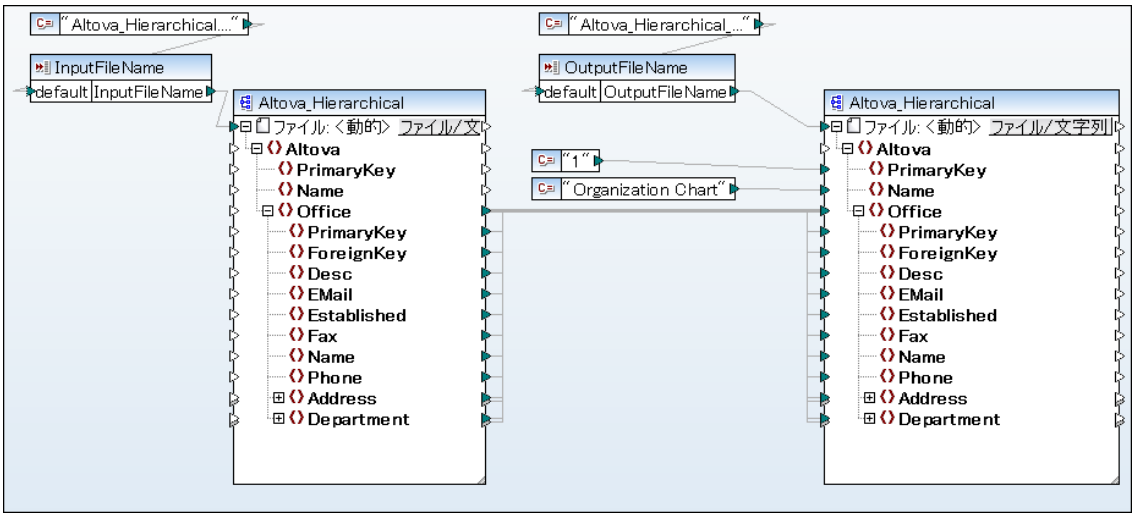
3. 「**OK**」をクリックします。

メモ: 値の指定 チェックボックスをクリックし横のボックスに値を入力した場合、入力された値は、マッピングをプレビューする際に、以前入力された（すなわち、デザイン時の実行時の）デフォルトの値を上書きします。しかしながら、同じ値は生成されたコードでは効果をもたらしません。

5.5.4 例 :名前をマッピングパラメーターとして使用する

このサンプルはランタイムに入力パラメーターを取り込むマッピングの実行に必要なステップを説明します。マッピングデザインファイルは、以下のパスにあるサンプルを使用します :**<Documents>\Altova\MapForce2017\MapForceExamples\FileNamesAsParameters.mfd**.

マッピングは2つの入力コンポーネントを使用します :**InputFileName** と**OutputFileName**。これらのコンポーネントはソースとターゲットXML ファイルの入力ファイル名（および出力ファイル名をそれぞれ）与えます。この理由のため、これらのコンポーネントは、**File: <dynamic>** アイテムに接続されています。



FileNamesAsParameters.mfd (MapForce Basic Edition)

InputFileName と**OutputFileName** コンポーネントは、マッピング内の単純型入力コンポーネントですので、マッピングを実行する際に入力パラメーターとして提供することができます。以下のセクションは以下の変換言語での実行方法を説明しています：

- XSLT 2.0 RaptorXML Server を使用

XSLT 2.0

XSLT 1.0 またはXSLT 2.0 出コードを生成すると、入力パラメーターは、RaptorXML Server での実行のため **DoTransform.bat** バッチファイルに書き込まれます（RaptorXML Server を参照）。**DoTransform.bat** ファイルを呼び出す際、異なった入力（または出力）ファイルを使用するため、コマンドラインで必要なパラメーターをパスするか、または、必要なパラメーターを含むように後者を編集します。

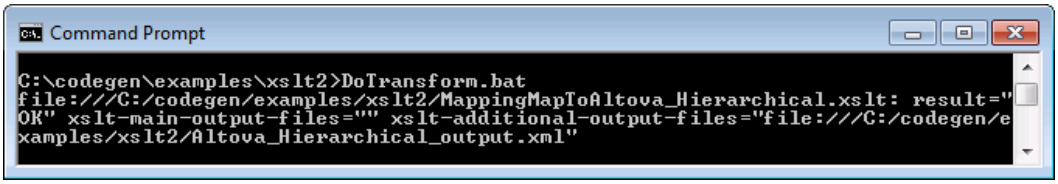
DoTransform.bat ファイル内でカスタム入力パラメーターを提供する：

1. **FileNamesAsParameters.mfd** サンプルからXSLT 2.0 コードを生成（**「ファイル | コード生成 | XSLT 2.0」**）します。
2. **<Documents>\Altova\MapForce2017\MapForceExamples** ディレクトリから **Altova_Hierarchical.xml** ファイルをXSLT 2.0 コードが直接生成されるディレクトリにコピーします（このサンプルでは、**c:\codegen\examples\xslt2**）。このファイルは、カスタムパラメーターとしての役割を果たします。
3. **DoTransform.bat** をカスタム入力パラメーターが（下でハイライトされているように）%* の前または後に含まれるように編集します。パラメーターの値は一重引用符で閉じられていることに注意してください。使用することのできる入力パラメーターは**rem**（リマーク）セクションにリストされています。

```
@echo off

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

DoTransform.bat ファイルを実行すると、RaptorXML Server は **Altova_Hierarchical.xml** を入力パラメーターとして使用して、変換を完了します。



5.6 マッピングから文字列の値を返す

マッピングから文字列を返すことが必要な場合単純型出力コンポーネントを使用します。マッピングエリアでは、単純型出力コンポーネントは、アイテムとシーケンスの構造の代わりに文字列データ型を持つターゲットコンポーネントの役割をします。この結果、単純型出力コンポーネント（または追加して）ファイルベースのターゲットコンポーネントの代わりに作成することができます。. 例：単純型出力コンポーネントを使用して、素早く関数の出力をテストとプレビューすることができます関数の出力（例：関数の出力をテストするを参照）。

単純型出力コンポーネントをユーザー定義関数の出力パラメーターと混同しないように注意してください。（ユーザー定義関数を参照）。類似点と相違点は、以下のとおりです。

出力コンポーネント	ユーザー定義関数出力パラメーター
**関数	出力の挿入」**メニューから追加	**関数	出力の挿入」**メニューから追加
文字列をデータ型として持ちます。	単純型と複雑型のデータ型を持つことができます
マッピング全体に適用することができます。	定義された方法で関数のコンテキストに適用することができます。

必要であれば、複数の単純型出力コンポーネントをマッピングに追加することができます。単純型出力コンポーネントをファイルベースのターゲットコンポーネントと共に使用することができます。マッピングが複数のターゲットコンポーネントを含む場合、特定のコンポーネントにより返されるデータをプレビューすることができます。コンポーネントタイトルバー内の**「プレビュー」**（ ）ボタンをクリックすることにより、マッピングウィンドウの**「出力」**タブをクリックします。

単純型出力コンポーネントをMapForce 変換言語内で以下のように使用することができます：

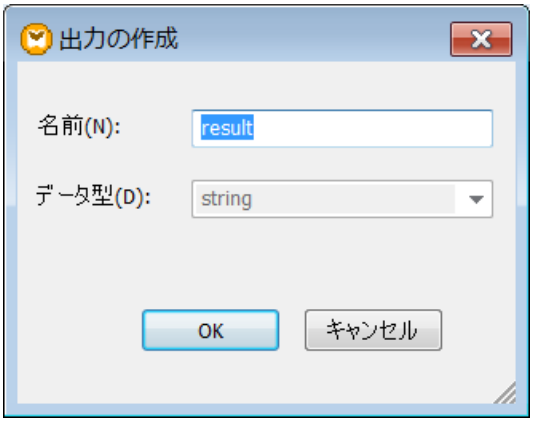
言語	作動のしくみ
XSLT 1.0、XSLT 2.0	マッピング内で定義された単純型出力コンポーネント生成されたXSLT ファイルが変換の出力になった場合。 RaptorXML Server を使用する場合、RaptorXML Server にマッピング出力をファイルに書き込み、`--output` パラメータに値としてパスするように命令することができます。 ファイルに出力を書き込む、**DoTransform.bat** ファイル内の`--output` パラメーターに追加または編集する場合。例：**次のDoTransform.bat** ファイルは**Output.txt** ファイルへのマッピング出力に書き込むために編集されました（ハイライトされたテキスト参照してください）。 `RaptorXML xslt --xslt-version=2 --input="MappingMapToResult1.xslt" --output="Output.txt" %* "MappingMapToResult1.xslt"` `--output` パラメーターが定義されていない場合、マッピングが実行される際、マッピング出力が標準出力ストリーム（stdout）に書き込まれます。

逆のマッピングを作成する場合、（メニューコマンド**「ツール | 逆マッピングの作成」**を使用すると）単純型出力コンポーネントは、単純型入力コンポーネントになります。

5.6.1 単純型出力コンポーネントの追加

マッピングエリアに出力コンポーネントを追加する:

1. ウィンドウがメイン(ユーザー定義関数ではない)のマッピングを表示していることを確認してください。
2. **「関数」**メニューから**「出力」**をクリックします。
3. コンポーネントのための名前を入力します。
4. **「OK」**をクリックします。



出力の作成ダイアログボックス

コンポーネント名を以下の方法で後に変更することができます:

- コンポーネントを選択して、**「コンポーネント」**メニューから**「プロパティ」**をクリックします。
- コンポーネントヘッダーをダブルクリックします。
- コンポーネントヘッダーを右クリックして、**「プロパティ」**をクリックします。

5.6.2 サンプル:関数出力のプレビュー

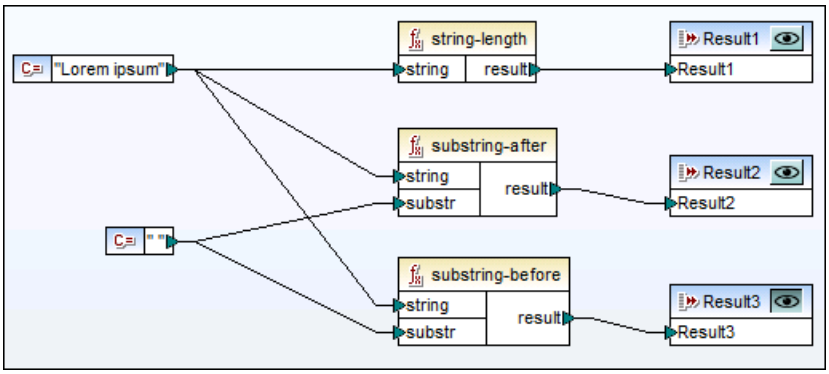
この例は、単純型出力コンポーネントを使用してMapForce関数により返された出力を確認する方法を説明します。関数に関する基本的理解とMapForce関数に関する全般的な知識があると、このサンプルをよく理解することができます。MapForce関数の基礎知識がない場合、例を開始する前に関数の使用を参照してください。

この例の目的は、マッピングエリアに複数の関数を追加することで、また単純型出力コンポーネントを使用して出力をプレビューする方法を理解することです。特に、例はcoreライブラリ内で使用することのできる複数の関数をマッピングエリアに追加する方法を説明しています。次に使用方法の概要が述べられています:

string-length	は、引数として与えられた文字列内の文字の数を返します。例:これを関数値 "Lorem ipsum" にパスすると、テキスト "Lorem ipsum" が持つ文字の数である結果は "11" です。
substring-after	は、引数として与えられたセパレーターの後に発生する文字列の一部を返します。例:これを関数値 "Lorem ipsum" とスペース文字 (" ") にパスすると、結果は "ipsum" です。
substring-before	は、引数として与えられたセパレーターの前に発生する文字列の一部を返します。例:これを関数値 "Lorem ipsum" とスペース文字 (" ") にパスすると、結果は "Lorem" です。

カスタムテキストの値に対して関数をテストするには(このサンプルでは"Lorem ipsum")以下のステップを行います:

1. 値を持つ定数 "Lorem ipsum" マッピングエリアを追加します (メニューコマンド **挿入 | 定数**) を使用して)。定数はテストされる関数のそれぞれの入力パラメーターになります。
2. **string-length**、**substring-after** と**substring-before** 関数をcore ライブラリ **string 関数** セクションからマッピングエリアにドラッグして、マッピングエリアに追加します。
3. 空のスペース (" ") を値として持つ定数を追加します。これは**substring-after** と**substring-before** 関数により必要とされるセパレーターパラメーターとなります。
4. 単純型出力 コンポーネントを (メニューコマンド **関数 | 出力の挿入**) を使用して追加します。他の名前を与えることも可能ですが、このサンプルでは、*Result1*、*Result2*、および *Result3*, と名前が付けられています。
5. コンポーネントを下に表示されるように接続してください。



単純型出力 コンポーネントを使用して関数の出力をテストする

上のサンプルで示されるとおり、"Lorem ipsum" 文字列はそれぞれの、**string-length**、**substring-after**、および**substring-before** 関数入力パラメーターとして振る舞います。更に、**substring-after** と **substring-before** 関数は、スペースの値を第 2入力パラメーター として取ります。*Result1*、*Result2*、および *Result3* コンポーネントは各関数の結果をプレビューするために使用することができます。

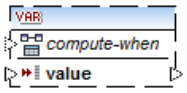
関数の出力をプレビューする

- コンポーネントタイトルバー内の **「プレビュー」** ( ) ボタンをクリックし、マッピングウィンドウの **「出力」** タブをクリックします。

5.7 変数の使用

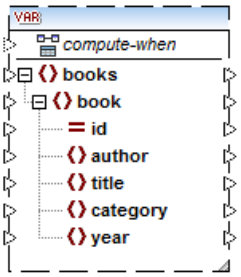
変数とは更なる処理のために中間マッピングの結果を保管するために使用される特別なコンポーネントです。マッピング上のデータを一時的に「記憶」し、処理する必要がある場合があります。例えば、ターゲットコンポーネントにコピーされる前に、フィルター、または、関数を適用するなど。

変数は、単純型（例えば、文字列、整数、ブール値、など）または 複合型（ツリー構造）であることができます。



単純型変数

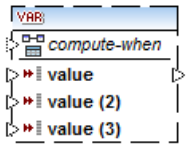
変数の構造を示す XML スキーマを提供して複合型の変数を作成することができます。スキーマが要素をグローバルに定義すると、変数構造のルートノードになるものを選択することができます。変数に関連した インスタンス XML ファイルが存在しない場合、変数のデータはマッピングランタイムに計算されます。



XML スキーマから作成された複合型変数

上のイメージでは、各変数には、compute-when という名前のアイテムが存在することに注意してください。このアイテムに接続することは任意です。これにより、マッピング上でどのように変数の値が計算されるかを管理することができます(次を参照してください :変数のコンテキストとスコープの変更)。

必要な場合、通常のコンポーネントと同様に、変数構造のアイテムを1つ以上のノード接続から受け入れられるように複製することができます（次を参照してください :入力の複製)。これは、しかしながらデータベーステーブルから作成された変数には適用されません。



複製された入力を持つ単純型変数

変数に関して最も重要な点は、変数はシーケンスであり、シーケンスを作成するために使用されるという点です。「シーケンス」という用語は、ここでは、ゼロまたは、それ以上のアイテムのリストを意味します（次も参照してください :マッピングのルールと戦略)。これにより、変数はマッピングの寿命内で複数のアイテムを処理することができるようになります。しかしながら、値を変数に与え、マッピングのその他の部分を同じに保つことも可能です（次を参照してください :変数のコンテキストとスコープの変更)。

ある程度までは、変数をチェーンマッピングの中間コンポーネントと比較することができます（次を参

照してください :チェーンマッピング)。しかしながら、マッピング内の各段階で中間ファイルを作成する必要がない場合、変数は柔軟性があり便利です。次のテーブルの概要は変数とチェーンマッピングの違いを示しています。

チェーンマッピング	変数
チェーンマッピングには、2つの独立したステップが含まれます。例えば、A、B、とC の3つのコンポーネントがマッピングに存在するとします。マッピングの実行には2つの段階があります :A からB へのマッピングの実行、および、B から C へのマッピングの実行。	マッピングの実行中、変数は、コンテキストとスコープに従い評価されます。コンテキストとスコープを影響することはできます 次を参照してください:変数のコンテキストとスコープの変更)。
マッピングが実行されると、中間結果は外部でファイルに保管されます。	マッピングが実行されると、中間の結果は内部で保管されます。変数の結果を含む外部ファイルは作成されません。 .
中間 結果は ボタンを使用してプレビューすることができます。	マッピング ランタイムで計算される変数の結果をプレビューすることはできません

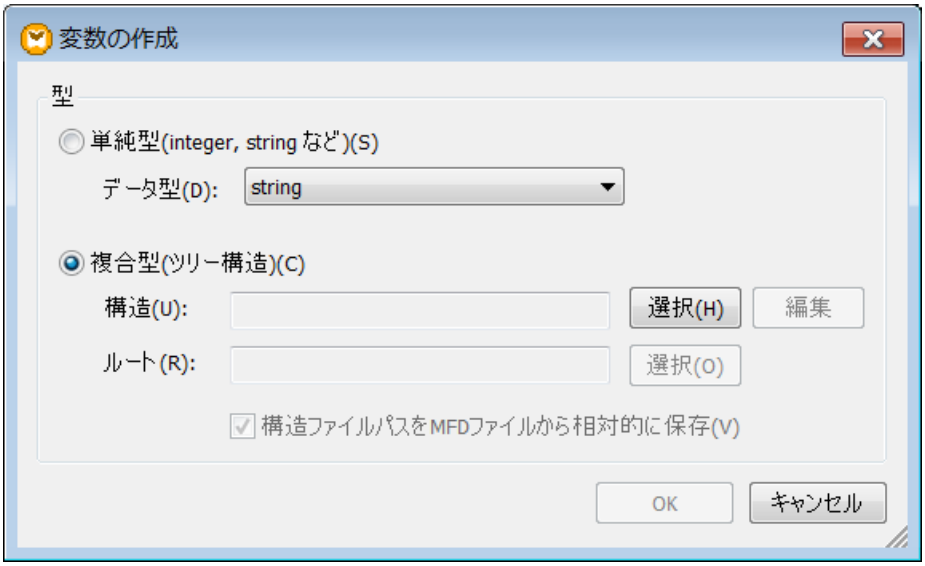
メモ: 変数は、マッピング変換言語がXSLT 1.0 に設定されていない場合サポートされません。

5.7.1 変数の追加

マッピングに変数を追加するには、以下に示されているようにいくつかの方法があります。

メニュー、または、ツールバーコマンドの使用

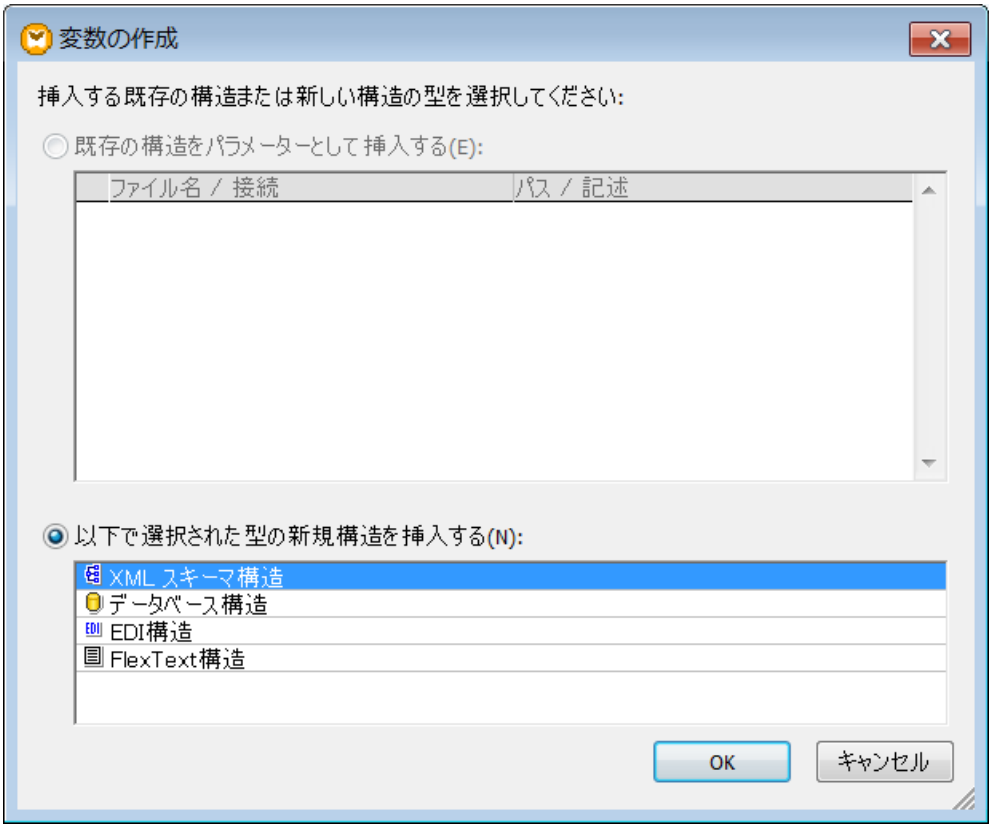
1. **挿入** メニューから、**変数** をクリックします。(**変数** ツールバーボタンを代わりにクリックします)。



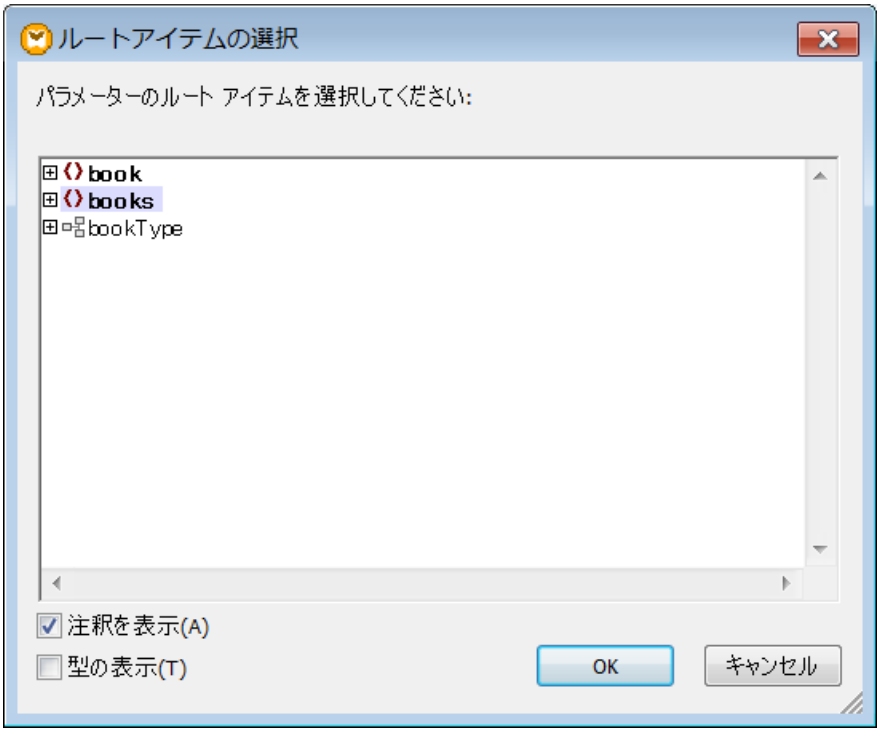
2. 挿入する変数の型を選択します (単純型 または 複合型)。

"複合型 "を選択した場合、追加ステップが存在します :

3. 変数の構造を与えるソースを選択するために、**選択** をクリックします (例えば、XML スキーマ、)。

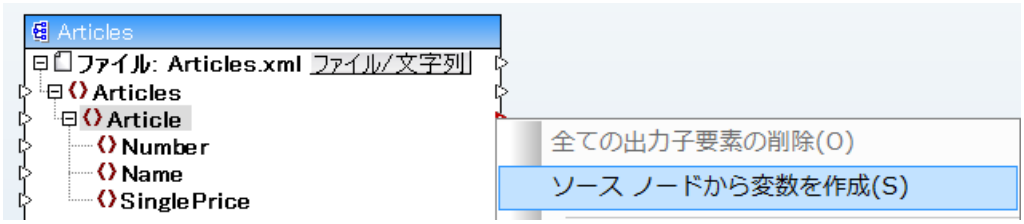


4. プロンプトされると、構造のルートアイテムを指定します。XML スキーマの場合、ルート アイテムは、グローバルに定義されている すべての要素であることができます。データベースの場合は、ルートアイテムはすべてのテーブル であることができます。

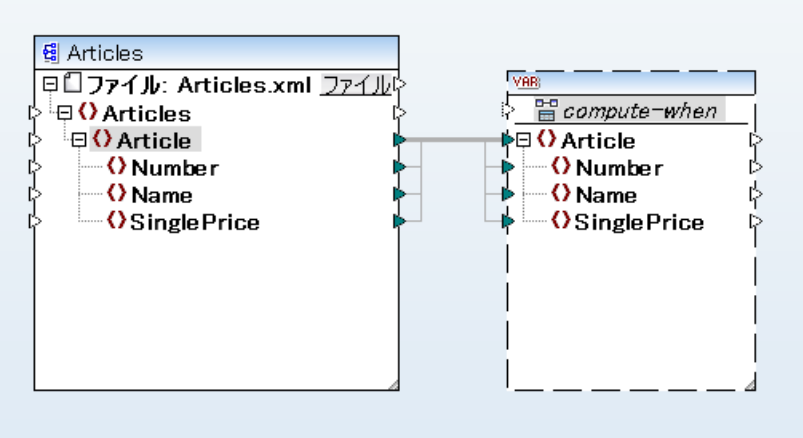


コンテキストメニューの使用

- コンポーネントの出力コネクタを右クリックします (このサンプルでは、 Article」) そして、**「ソースノードから変数を作成する」**を選択します。



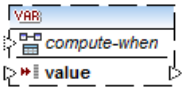
これにより、同じソーススキーマを使用する複合型の変数を作成し、自動的にすべてのアイテムをすべてをコピーする接続により接続します。



- ターゲットコンポーネントの入力コネクタを右クリックして、**「ターゲットノードのために変数を作成する」**を選択します。これにより、同じスキーマをターゲットとして使用し、複合型の変数を作成します。そして、すべてコピー接続を使用してすべてのアイテムを自動的に接続します。
- フィルターコンポーネント (on-true/on-false) の出力コネクタを右クリックして、**「ソースノードから変数を作成する」**を選択します。これにより、ソーススキーマを使用して複合型コンポーネントを作成し、フィルター入力にリンクされているアイテムを中間コンポーネントのルート要素として自動的に使用します。

5.7.2 変数のコンテキストとスコープの変更

各変数には、変数のスコープを管理することを許可する`compute-when` 入力アイテムが存在します。すなわち、マッピングが実行される際に変数の値がいつ、どの頻度で計算されるかを管理することができます。この入力は、多くの場合接続する必要がありませんが、デフォルトのコンテキストを上書きする場合、または、マッピングのパフォーマンスを最適化するには、必要です。



「compute-when」アイテム

次のテキストでの、サブツリーは、ターゲットコンポーネント内のアイテム/ノードセットとその子孫を意味します。例えば、`<FirstName>` と`<LastName>` 子要素を持つ `<Person>` 要素。

変数の値 は、変数コンポーネントの出力サイドでデータが使用できることを意味します。

- 単純型の変数に関しては、コンポーネントプロパティ内で指定されているデータ型を持つ動的な値のシーケンスであることを意味します。
- 複合型の変数に関しては、それぞれが自身の子孫ノードを含む (コンポーネント プロパティ内で指定されている型の) ルートノードのシーケンスであることを意味します。

動的な値 (または、ノード) のシーケンスは、1つの要素、または、要素を全く含まない場合があります。これは、変数の入力サイドに何が接続されているか、および、ソースとターゲットコンポーネント内の親アイテムの存在により異なります。

「compute-when」が接続されていない場合 (デフォルト)

`compute-when` 入力アイテムが (ソースコンポーネントの出力ノードに) 接続されていない場合、ターゲットサブツリー内で最初に使用される際に変数の値は (コネクタにより変数コンポーネントからターゲットコンポーネント内のノードに直接、または、関数を使用して間接的に) 計算されます。同じ変数の値は、サブツリー内のターゲット子ノードすべてのために使用されます。

実際の変数の値は、ソースとターゲットコンポーネントの親アイテム間の接続により異なります。

このデフォルトの振る舞いは、正規ユーザー定義 関数とWeb サービス関数の呼び出しの複合型の出力と同じです。

変数の出力が複数の関連したにターゲットノードに接続されている場合、変数の値は、それぞれのアイテムのために個別に計算されます。これにより、各ケース内で異なる結果を生成することができます。これは異なる親の接続は、変数の値が評価されるコンテキストを影響するからです。

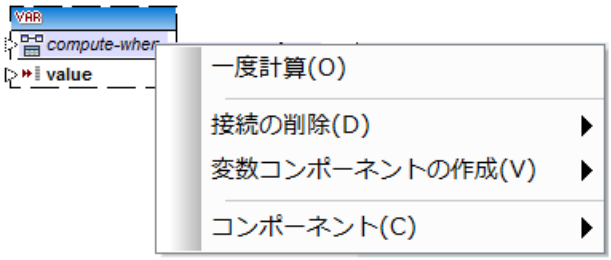
「compute-when」が接続されている場合

ソースコンポーネントの出力コネクタを`compute-when` に接続すると、ソースアイテムが最初にターゲットサブツリー内で使用される都度に変数が計算されます。

変数は実際には、`compute-when` に接続されているアイテムの子アイテムのように振る舞います。これは、新規のアイテムがソースコンポーネント内のシーケンスから読み込まれる都度、ランタイム変数が再評価されます。これは、MapForce 内の接続を管理する一般的なルールに関連しています：各ソースアイテムのために、1つのターゲットアイテムが作成されます。`compute-when` に関しては、各ソースアイテムのために、変数の値が計算されます (次を参照してください：マッピングのルールと戦略)。

「compute-once」

必要であれば、ターゲットコンポーネントの前に1度変数の値を計算し、変数をマッピングの残りでグローバルな定数にすることを選択することができます。これをおこなうには、`compute-when` アイテムを右クリックして、コンテキストメニューから **「1度計算する」** を選択します：



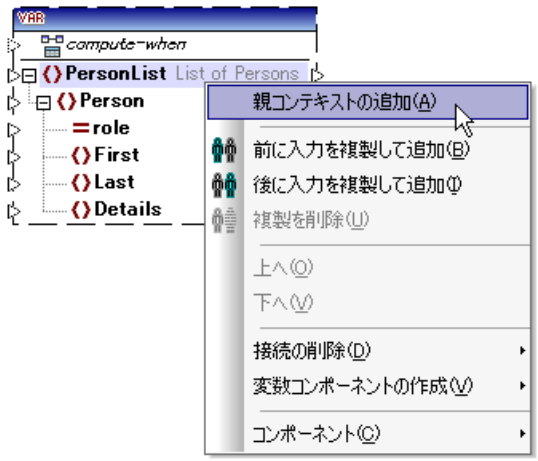
変数のスコープを`compute-when=once` に変更する場合、このような変数は1度のみ評価されるため、入力コネクタは、`compute-when` アイテムから削除されます。

実際の関数の結果が評価される前に、ユーザー定義関数 `compute-when=once` 変数が関数が呼び出される都度評価されます。

親コンテキスト

親コンテキストを追加する必要がある場合があります。例えば、マッピングが複数のフィルターを使用し、反復するために親ノードを追加する必要がある場合など。 マッピングコンテキストのオーバーライド を参照してください。

変数に親コンテキストを追加するには、ルートノード (このサンプルでは 「PersonList」) を右クリックします、そして、コンテキストメニューから、 **親コンテキストを追加する** を選択します。これにより新規ノード、親コンテキストを既存の階層構造に追加します。

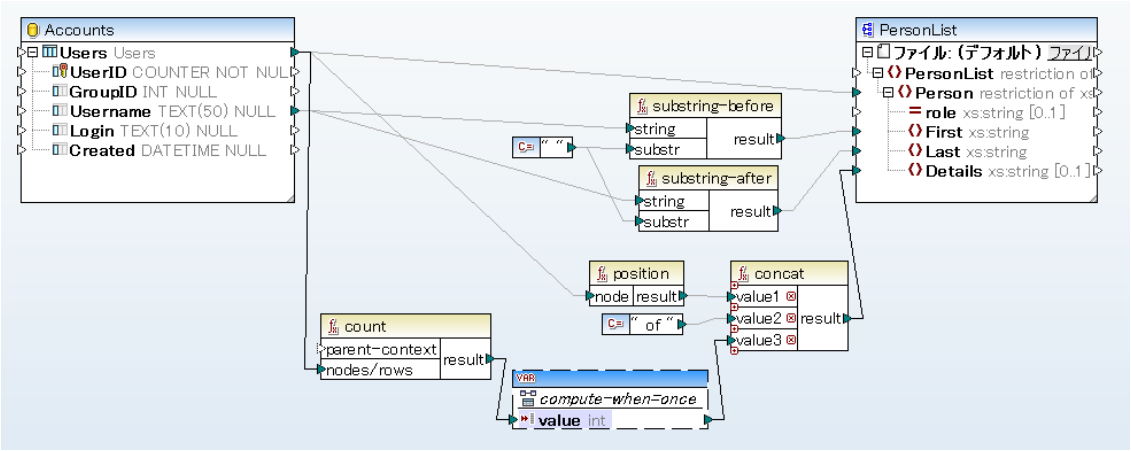


親コンテキストは、仮想の「親」ノードをコンポーネント内の階層構造に追加します。これにより、同じ、または、異なるソースコンポーネント内で追加ノードを反復することができます。

5.7.3 サンプル :データベースのテーブルの行の計算

このサンプルで説明されているマッピングは、**<Documents>\Altova\MapForce2017\MapForceExamples** フォルダー内の**DB_UserList.mfd** です。このマッピングは、ユーザーのレコードを "Users" という名前のデータベーステーブルから抽出し、XML ファイルに書き込みます。データベースの列 "Username" には、個人の姓と名の両方が含まれています (例えば、"Vernon Callaby")。このマッピングの目的は以下のとおりです :

1. "Users" テーブル内の各レコードのために、XML ファイル内に新規の`Person` 要素を作成する。
2. データベースフィールド "Username" から抽出された値をXML ファイル内の個別のフィールド ("First" と "Last") に分割する。
3. 各レコードのために、データベース内に存在するレコードの全体の番号に対して比較される連番を検索し (例えば、"4 件中の 1件目 "など) `Details` 要素にこの情報を書き込みます。



DB_UserList.mfd

上で説明されているとおり、最初の目的を達成するために、ソース "Users" テーブルとターゲットXML ファイルの`Person` 要素 の間に接続が描かれます。これにより、ソーステーブル内の各レコードのために、ターゲット内に新規の`Person` 要素が作成されます。

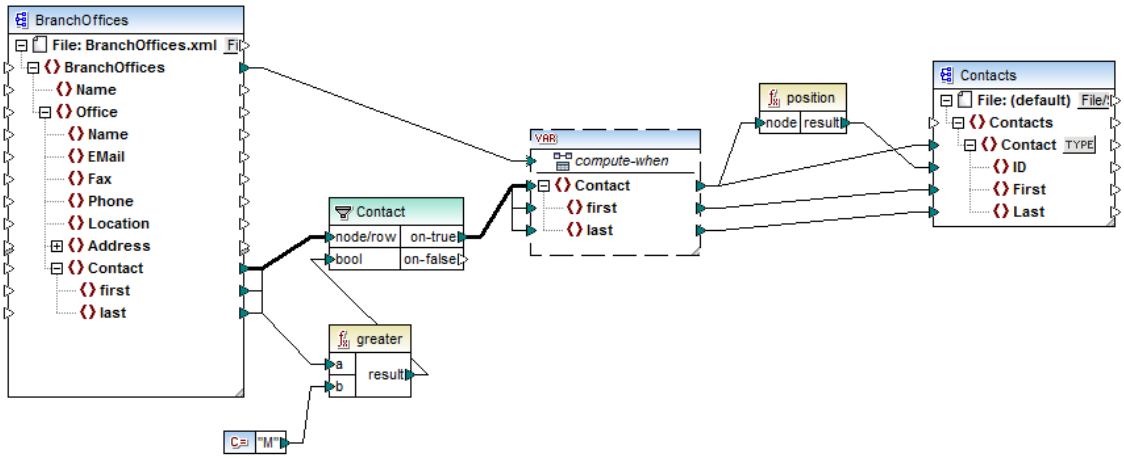
フィールド "Username" の値が**substring-before** と**substring-after** 関数に与えられています。これらの2つの関数は、スペース文字 (" ")の前後のテキストをそれぞれ抽出し、このマッピングの目的を達成します。

3つ目の目的を達成するために、マッピングは**count** 関数を使用します。集計関数の結果は、変数にパスされます。変数は、マッピング上に結果が保管され、各個人の "Details" 要素 をターゲットXML に書き込む際に使用することができます。効率性のために、データベースのレコードは1度のみ数えられるべきであり、変数のスコープは`compute-when=once` に設定されています (次を参照してください:変数のコンテキストとスコープの変更)。

5.7.4 サンプル :データベースのテーブルの行の計算

このサンプル内で説明されているマッピングのシーケンスは、**<Documents>\Altova\MapForce2017 \MapForceExamples** フォルダー内の**PositionInFiltered.mfd** で使用することができます。

このマッピングは、複数の個人の連絡先のデータを含むXML ファイルを読み取り、フィルターし、ターゲットXML ファイルに書き込みます。このマッピングの目的は、ソースXML ファイルから、「M」または、以降のアルファベットから始まる姓を持つ個人をフィルターします。抽出された連絡先には、番号がつけられている必要があります。番号は、ターゲットXML ファイル内で各連絡先の一意の識別子としての役割を果たします。



PositionInFilteredSequence.mfd

上の目的を達成するには、次のコンポーネント型 がマッピングに追加されました:

- フィルター (次を参照してください:フィルターと条件)
- 複合型変数 (次を参照してください:変数の追加)
- 関数 **greater** と**position** (次を参照してください:関数との作業)
- 定数 (定数を追加するには、メニューコマンド **挿入 | 定数** を選択します)。

変数は、ソースコンポーネントと同じスキーマを使用します。変数を右クリックして、コンテキストメニューから**プロパティ**を選択すると、この変数構造のためにルートノードとして ノード**BranchOffices/Office/Contact** が選択されていることを確認してください。

最初に、ソースコンポーネントのデータは、フィルターにパスされます。フィルターは、変数にフィルターの条件を満たすレコードのみをパスします。具体的には、フィルターは、最初の名前が「M」に等しい、または、大きい値を持つ`Contact` ノードのみを取得するように構成されています。この目的を達成するには、関数 **greater** は、各 `last` アイテムを定数値 「M」と比較します。

変数には、ソースコンポーネント (`BranchOffices`) のルートアイテムに接続されている`compute-when` 入力があります

す。ランタイムでは、これは、ソースコンポーネント内のシーケンスから新規のアイテムが読み込まれる都度、変数を再評価するようにします。このマッピングでは、しかしながら、compute-when アイテムの接続、または、未接続に違いはありません。この理由は、変数が(フィルターを使用して間接的に) Contact ソースアイテムに接続されており、フィルター条件を満たす Contact のインスタンスの数だけ計算がおこなわれるからです。

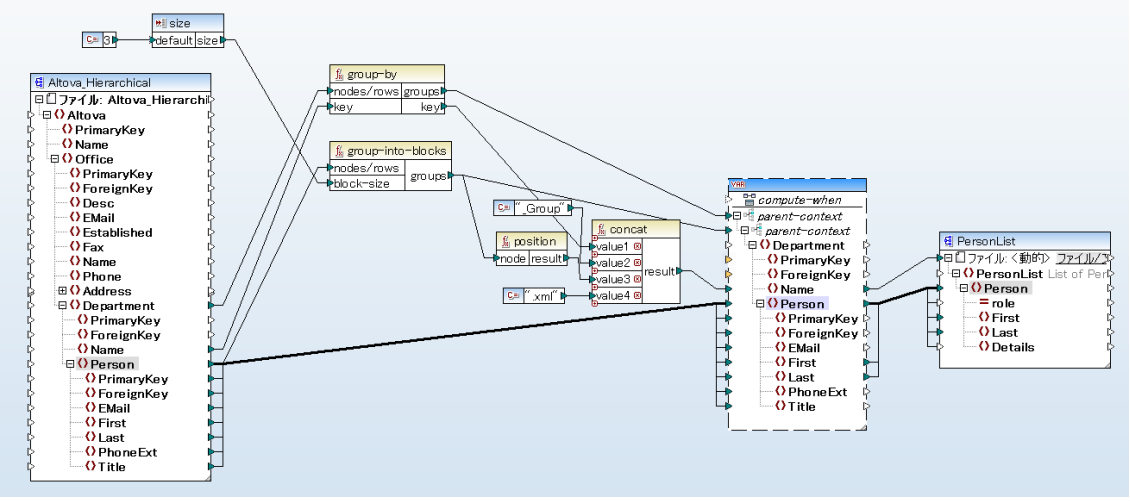
position 関数は、変数の各反復のために現在のシーケンスの番号を返します。8件の連絡先がフィルターの条件を満たしています。ですから、マッピングをプレビューして、出力を確認すると、1から8のID が、ターゲットコンポーネントのID 要素に書き込まれていることが確認することができます。

変数の必要性に関しては、すべてのレコードに番号付けをする必要性に応じるものです。フィルターの結果を直接ターゲットコンポーネントに接続すると、Contact の各発生に番号を付けることができません。このマッピング内の変数の目的は、ですから、Contact の各インスタンスを一時的にマッピングに保管し、ターゲットに書き込まれる前に番号を付けることです。

5.7.5 サンプル 記録のグループ化、および、サブグループ化

このサンプルで説明されているマッピングは、**<Documents>\Altova\MapForce2017\MapForceExamples** フォルダー内の**DividePersonsByDepartmentIntoGroups.mfd** です。

このマッピングは、架空の会社の社員のレコードを含むXML ファイルを処理します。社内には2つのオフィスが存在します：「Nanonull, Inc.」と「Nanonull Partners, Inc」。各オフィスには複数の部署が存在します (例えば、「IT」、「マーケティング」など)。また、各部署には一名以上の社員が存在します。マッピングの目的は、オフィスに関わりなく各部署から3人までで構成されているグループを作成することです。グループのサイズはデフォルトでは3名です。しかしながら、必要に応じて変更することもできます。各グループは、フォーマット "<Department Name>_GroupN" の名前を持つ個別のXML ファイルとして保存される必要があります (例えば、**Marketing_Group1.xml**、**Marketing_Group2.xml**、など)。



DividePersonsByDepartmentIntoGroups.mfd

上で説明されているとおり、マッピングの目的を達成するには、複合型変数と住に関数である)他の コンポーネント型をマッピングに追加します。ソースXML 内のDepartment アイテムと同じ構造が変数にも存在します。プロパティを表示するために変数を右クリックすると、ソースコンポーネントと同じXML スキーマが使用されていること、および、ルート要素として Department が存在していることに気が付きます。重要な点は、各部署のコンテキスト内で変数が最初に計算され、次に各グループのコンテキスト内で計算されるように、変数には、2つのネストされた親コンテキストアイテムが存在することです (次も参照してください：変数のコンテキストとスコープの変更)。

始めに、マッピングは、各部署の名前を取得するためにすべての部署を反復します (これは各グループに対応するファイル名を後に作成するために必要とされます)。これは、**group-by** 関数をDepartment ソースアイテムに接続すること、部署

名をグループ化のキーとして与えることにより達成されます。

次に、各部署のコンテキスト内で、2番目のグループ化が発生します。具体的には必要とされる一のグループを作成するために、マッピングが**group-into-blocks** 関数を呼び出します。デフォルトの値が "3" である単純型のコンポーネントが各グループのサイズを提供します。デフォルトの値は、定数により与えられます。グループのサイズを変更するために、このサンプルでは、必要に応じて定数を簡単に変更することができます。しかしながら、 "サイズ" 入力コンポーネントも変更することができ、マッピングが生成されるコード、または、MapForce Server により実行される場合、グループのサイズは、マッピングへのパラメーターとして便利に提供することができます。更に詳しい情報に関しては、次を参照してください:マッピングにパラメーターを与える

次に、変数の値はターゲット PersonList XML コンポーネントにより与えられます。作成された各グループのためのファイル名は、**concat** 関数を使用して次の部分を連結することにより計算されます:

1. 各部署の名前
2. 文字列 "_Group"
3. 現在のシーケンス内のグループの番号 (例えば、 "1" は、この部署の最初のグループを指します)
4. 文字列 "xml"

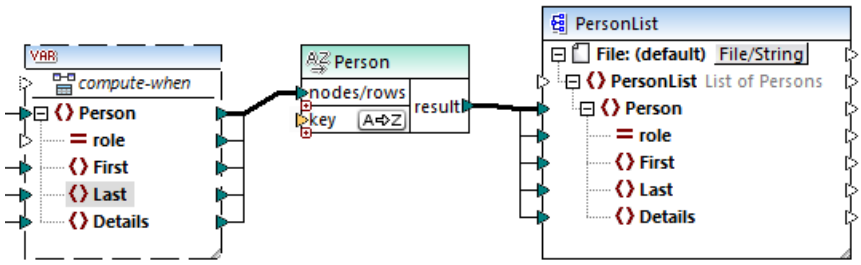
この連結の結果は、変数の Name アイテム内に保管され、ターゲットコンポーネントに動的なファイル名として与えられます。これにより、受信した値のために新規のファイルが作成されます。 このサンプルでは、変数は、8個のグループを計算するため、マッピングが実行されると8件の出力ファイルが作成されます。 この技術に関する更に詳しい情報は、次を参照してください:複数の入力または出力ファイルを動的に処理.

5.8 データの並べ替え

特定のソートキーをベースにした入力データの並べ替えを行う場合、ソートコンポーネントを使用してください。ソートコンポーネントは、XSLT2、XQuery、内蔵の実行エンジンにてサポートされます。

マッピングに並べ替えコンポーネントを追加するには、以下を行います：

- 既存の接続を右クリックして、コンテキストメニューから **「Insert Sort: Nodes/Rows」** を選択します。並べ替えコンポーネントを自動的に挿入し、ソースとターゲットコンポーネントに自動的に接続します。 例えば、下のマッピングでは、並べ替えコンポーネントが変数とXML コンポーネント間に挿入されます。 (並べ替えるフィールドである) 並べ替えキーのみを手動で接続します。

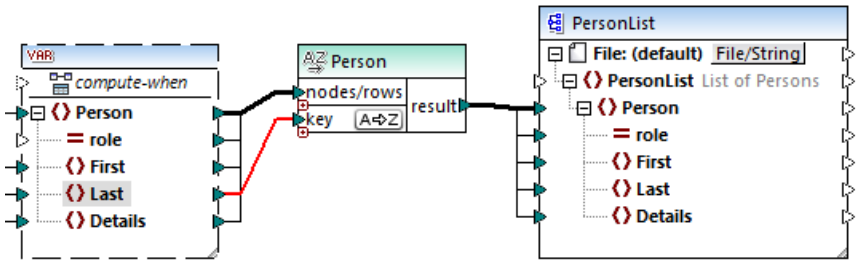


- **挿入」** メニューから **並べ替え」** (または、 **並べ替え」**  ツールバーボタン)をクリックします。 「接続」フォーム内に並べ替えコンポーネントが挿入されます。

ソース コンポーネントに接続が構築されると、,タイトルバー名は、`nodes/rows` アイテムに接続されているアイテムの名前に変更されます。

並べ替えるアイテムを定義する：

- 並べ替えコンポーネントの`key` パラメーターを並べ替えるアイテムに接続します。例えば、下のマッピングでは、`Person` nodes/rows は、フィールド`Last` に従い並べ替えられます。

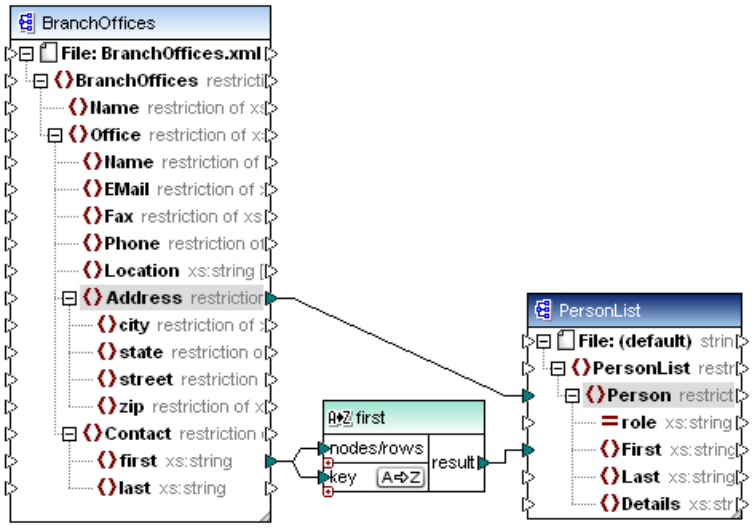


並べ替えの順序の変更：

- 並べ替えコンポーネント内の  アイコンをクリックすると、  に変更され、並べ替えの順序は逆の順序になったことを示します。

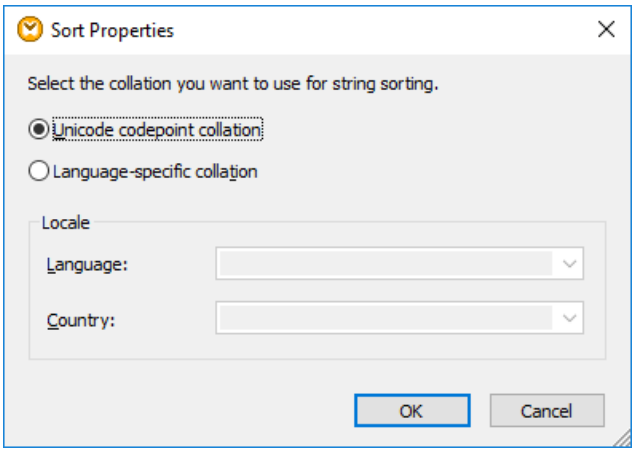
単純型アイテムにより構成される入力データの並べ替え：

- アイテムを並べ替えコンポーネントのnodes/rows とkey パラメータに接続します。下のマッピングでは、単純型 first 要素が並べ替えられます。



言語特有のルールを使用して文字列を並べ替える：

- 並べ替えコンポーネントのヘッダーをダブルクリックして、「並べ替え」プロパティダイアログボックスを開く。

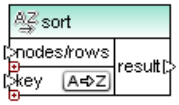


Unicode コードポイント照合：この（デフォルト）オプションにより、コードポイントの値をベースにした比較・並べ替えが行われます。コードポイント値とはUnicode コンソーシアムにより採用されたUniversal Character Set (UCS) における絶対文字に割り当てられた整数のことです。このオプションでは複数の言語やスクリプト間で並べ替えを行うことが可能になります。

言語固有の照合：このオプションにより、並べ替えを行う際にベースとなる言語や国を指定することができます。このオプションは内蔵の実行エンジン（BUILT-IN）が選択されている際にサポートされ、XSLT におけるサポートはコードの実行に使用されるエンジンに左右されます。

5.8.1　複数のキーを使用した並べ替え

マッピングに並べ替えコンポーネントを追加した後、key という名前の並べ替えキーがデフォルトで作成されます。

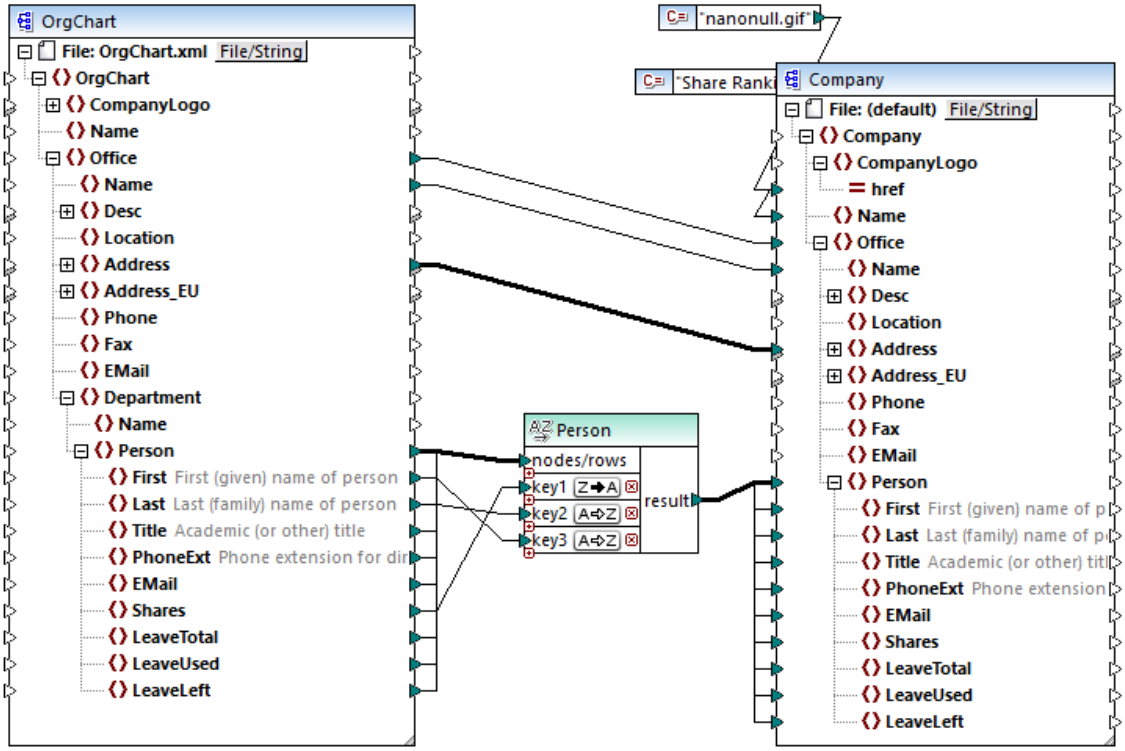


デフォルトの並べ替えコンポーネント

複数のキーを使用して並べ替える場合、並べ替えコンポーネントを以下のように調整します：

- **「キーの追加」**（ ⊡ ）アイコンをクリックして新しいキーを追加します（例えば、下のマッピング内の key2）.
- **「キーの削除」**（ ⊠ ）アイコンをクリックしてキーを削除します。
- ⊡ アイコンに接続をドロップして、キーを追加し、接続します。

複数のキーによる並べ替えを示すマッピングは、次のパスで検索することができます：**<Documents>\Altova\MapForce2017\MapForceExamples\SortByMultipleKeys.mfd**.



SortByMultipleKeys.mfd

上のマッピングでは、Person レコードは3つの並べ替えキーにより並べ替えられています：

1. Shares（個人が所有するシェア数）
2. Last（姓）
3. First（名）

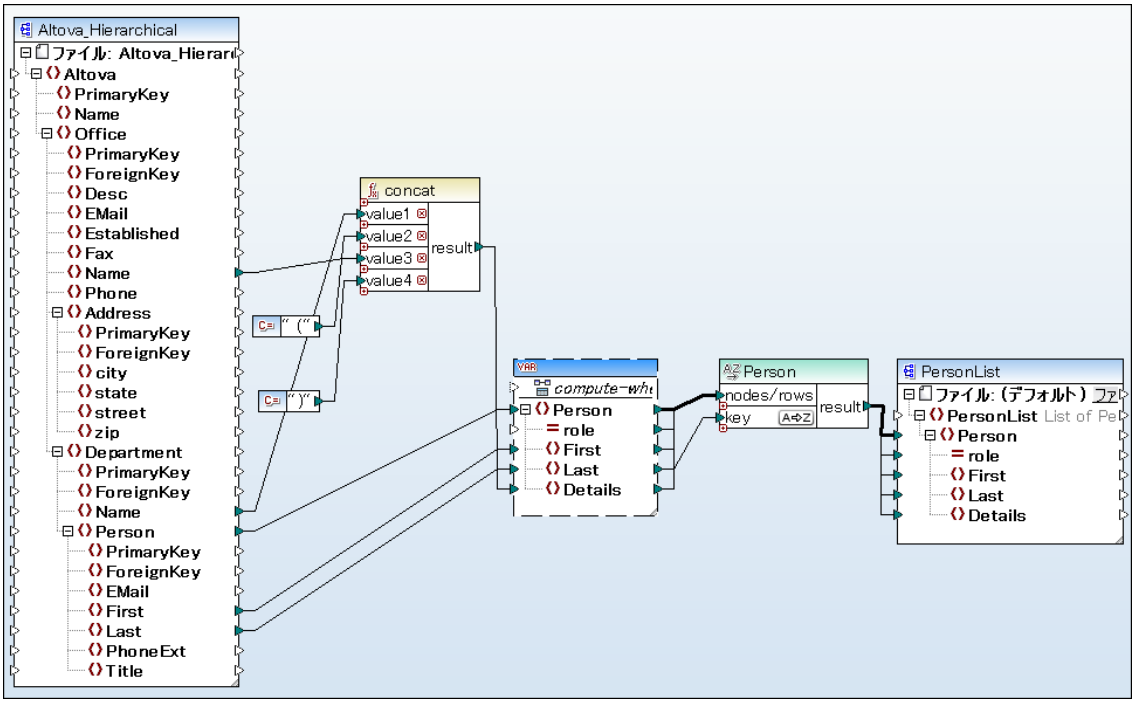
並べ替えコンポーネント内の並べ替えキーの位置は、並べ替えの優先順序を決定します。例えば、上のマッピングでは、所

有するシェア数により並べ替えられています。これは、最も高い優先順位を持つ並べ替えキーです。所有するシェア数が同じの場合も同様で、姓により並べ替えられます。最後に、同数のシェアを持つ同じ性を持つ個人は、名を考慮して並べ替えられます。

各キーの並べ替えは異なる順番であることができます。上のマッピングでは、キー `Shares` は降順の並べ替え順序 (Z-A) が与えられていますが、その他2つのキーには昇順の並べ替え順序 (A-Z) が与えられています。

5.8.2 変数を使用して並べ替える

希望する結果を達成するためにマッピングに中間変数を追加する必要がある場合があります。このサンプルでは、XML ファイルから記録を抽出し、中間変数を使用して並べ替える方法について説明されています。このサンプルには、次のパスで検索することのできるマッピングのサンプルが存在します：**<Documents>\Altova\MapForce2017\MapForceExamples\Altova_Hierarchical_Sort.mfd**。



Altova_Hierarchical_Sort.mfd

このマッピングは、**Altova_Hierarchical.xml** という名前のソースXML ファイルからデータを読み取り、ターゲット XML ファイルに書き込みます。上に示されるとおり、ソースXML には架空の企業の情報が含まれています。企業はオフィスに分岐されています。オフィスは部署に分岐されており、部署は社員に分岐されています。

ターゲットXML コンポーネント、`PersonList`、は`Person` レコードのリストが含まれています。`Details` アイテムは、社員が属するオフィスと部署の情報が保管されています。

ソースXML から全ての社員を抽出し、姓をアルファベット順に並べ替えることが目的です。また、オフィスと部署名は、`Details` アイテムに書き込まれる必要があります。

この目的を達成するために、このサンプルは、次のコンポーネントの型を使用します：

1. **`concat`** 関数：このマッピング内では、この関数は、フォーマット`Office(Department)` で文字列を返します。オフィス名、部署名、かっこ開始し終了する2つの定数を入力として取ります。 次も参照してください：関数と作業。

2. 中間変数：変数の役割は同じマッピングコンテキストに個人に関連する全てのデータを運ぶことです。変数によりマッピングがそれぞれの個人のコンテキストで、個人の部署とオフィスを探します。すなわち、変数は、個人が所属するオフィス名と部署名を「記憶」することを意味します。変数が不在の場合、コンテキストは、(同じXML スキーマを使用して)正確ではなくマッピングは希望しない結果を生成します。(マッピングの実行方法の詳細に関しては、次を参照してください：マッピングルール戦略)。変数はXML ファイルの構造を複製することに注意してください。これにより、全てコピー接続を使用して、ターゲットに並べ替えの結果を接続することができます。次も参照してください：変数の使用 と全てコピー接続.
3. 実際の並べ替えを行う、並べ替えコンポーネント：Sort コンポーネントのキー入力は、姓により個人の記録を並べ替える変数のLast アイテムに接続されていることに注意してください。

5.9 フィルターと条件

データをフィルターする場合、または値を条件に対して取得する場合、以下のうち1つのコンポーネント型を使用することができます：

- フィルター:Nodes/Rows ( )
- If-Else 条件 ( )

挿入 メニューから、または、**コンポーネントの挿入** ツールバーからこれらのコンポーネントをマッピングに追加することができます。上記のそれぞれのコンポーネントには特定の振る舞いや必要条件があることにご注意してください。差異は下のセクションで説明されています。

ノードと行をフィルターする

XML ノードを含むデータをフィルターする場合、**Nodes/Rows をフィルター** コンポーネントを使用します。この**Nodes/Rows をフィルター** コンポーネントにより、true または false 条件をベースにし、大きなデータセットからノードのサブセットを抽出することができます。マッピングエリアでのこの構造は以下のように表されます：

上の構造では、に接続されている条件は、**bool** に接続された**node/row** が、**on-true** または**on-false** 出力に接続されているかを決定します。具体的には、条件がtrue の場合、**node/row** は、**on-true** 出力にリダイレクトされます。一方、条件がfalse の場合、**node/row** は、**on-false** 出力にリダイレクトされます。

マッピングがフィルターの条件を満たすアイテムのみを消費する必要がある場合、**on-false** 出力を未接続のままにできます。フィルターの条件を満たさないアイテムを処理する必要がある場合、このようなアイテムがリダイレクトされる、**on-false** 出力をターゲットに接続します。

手順を追ってのマッピングのサンプルは、サンプル：ノードのフィルターを参照してください。.

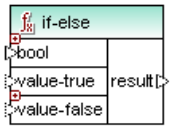
値を条件付きで返す

(ノードまたは行ではない)単一の値を条件付きで取得する必要がある場合、**If-Else 条件** を使用します。If-Else 条件は、ノードまたは行をフィルターするには、適していません。**Nodes/Rows をフィルター** コンポーネントとは異なり、**If-Else 条件** は、文字列または整数などの単純型を返します。ですから、**If-Else 条件** は、単純型を条件付きで処理するシナリオのみに適しています。例えば、各月の平均気温のリストがあると仮定します：

```
<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>
```

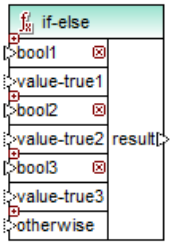
If-Else 条件 により、リスト内の各アイテムのために気温が摂氏20度を超える場合は、値 "高 "を、気温が摂氏5度以下の場合は、値 "低 "を返すように設定することができます。

マッピング上で**If-Else 条件** の構造は、以下のようになります :



bool に接続された条件がtrue の場合、**value-true** に接続された**result** としての出力になり、条件がfalse の場合、**value-false** に接続された**result** としての出力になります。**result** のデータ型を事前に知ることはできず、これは値のデータ型、により異なり、**value-true** または**value-false**. に接続された値により異なります。重要な点は、常に(文字列および整数などの)、単純型であるということです。入力値を(ノードまたは行などの)複合型に接続することは**If-Else 条件**によりサポートされていません。

If-Else 条件は拡張することが可能です。これは、**追加** ( ) ボタンをクリックすることにより、コンポーネントに複数の条件を追加できることを意味します。以前に追加された条件を削除するには、ボタンを**削除** ( ) をクリックします。この機能により、条件がtrue の場合は、複数の条件をチェックして、各条件のために異なる値を返すことができます。



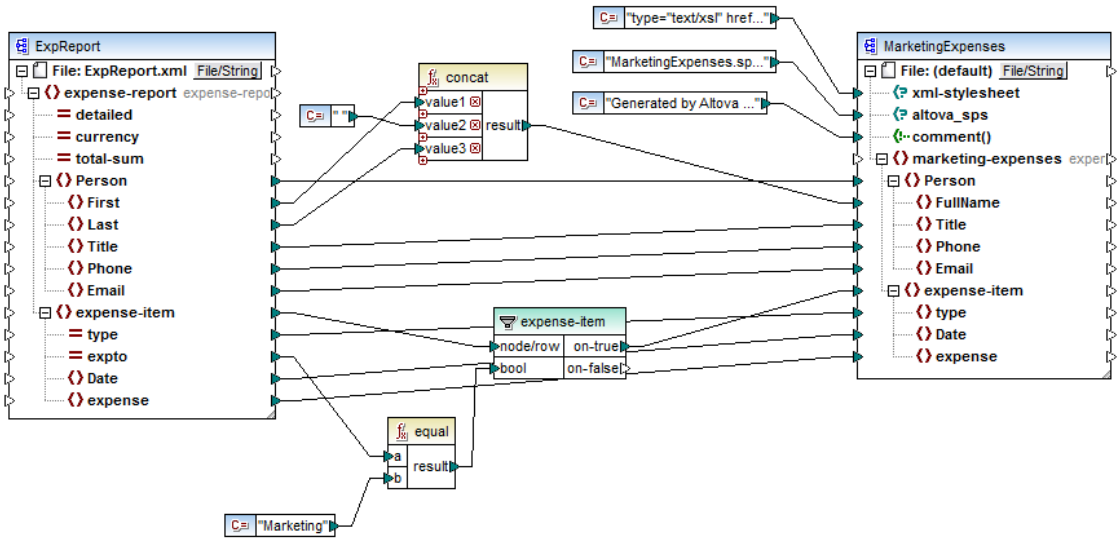
展開された**If-Else 条件** は上から下へと評価されます(最初の条件が最初にチェックされ、2番目が次にチェックされます)。条件の全てが真ではない場合に値を返す場合は、**otherwise** に接続してください。

手順を追ってのマッピングのサンプルは、例 :値を条件付きで返すを参照してください。

5.9.1　サンプル :ノードのフィルター

この例は、true/false 条件を基にしてノードをフィルターする方法について説明しています。**Filter: Nodes/Rows** ( ) コンポーネントがこの目的を達成するために使用されています。

この例で説明されているマッピングは以下のパスで検索することができます :**<Documents>\Altova\MapForce2017\MapForceExamples\MarketingExpenses.mfd**。



上に示されるように、このマッピングは、経費の方向ｋ ("ExpReport") のデータを含むソースXML からデータを読み取り、ターゲットXML ("MarketingExpenses") に書き込みます。ターゲットとソースの間には複数の他のコンポーネントが存在します。最も関連性の高いコンポーネントは、このトピックの主題を表す **expense-item** フィルター（ ）です。

このマッピングのゴールは、マーケティング部署に属する経費アイテムのみをフィルターすることです。この目的を達成するために、フィルター コンポーネントがマッピングに追加されました。（フィルターを追加するには、**「挿入」** メニューをクリックして、**「Filter: Nodes/Rows」** をクリックします）。

各経費がマーケティング部署に属するかを識別するために、このマッピングは、ソース内の "expto" 属性の値を確認します。この属性が経費がマーケティングの経費である場合、値 "Marketing" を有します。例えば、下にリストされるコードでは、最初と3番目のアイテムは、マーケティングに属し、2番目のアイテムは開発部署に、4番目は販売部署に属することが示されています：

```
...
   <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>
...
```

マッピングが実行される前のXML 入力

マッピングエリアで、フィルターの **node/row** 入力が、ソースコンポーネント内の **expense-item** ノードに接続されています。これにより、フィルターコンポーネントが処理されなければならないノードのリストを取得することを保証します。

フィルタリングが発生する条件を追加するには、MapForce core ライブラリから**equal** 関数 を追加しました (詳細に関しては、以下を参照してください:関数と作業する)。**equal** 関数は、"type" 属性を値 "Marketing" を持つ定数と比較します (定数を追加するには、**挿入**」メニュー をクリックして、**定数**をクリックします)。

条件を満たすアイテムのみをフィルターする必要があるため、フィルターの**on-true** 出力のみをターゲットコンポーネントに接続します。

マッピングの結果をプレビューする準備が整うと、**出力**」タブをクリックします。MapForce は、フィルターの**bool** 入力に接続されている条件である、それぞれの経費アイテムノードを評価します。条件がtrue の場合、経費アイテムノードはターゲットにパスされます。それ以外の場合は、無視されます。結果、条件を満たす経費アイテムのみが出力内に表示されます:

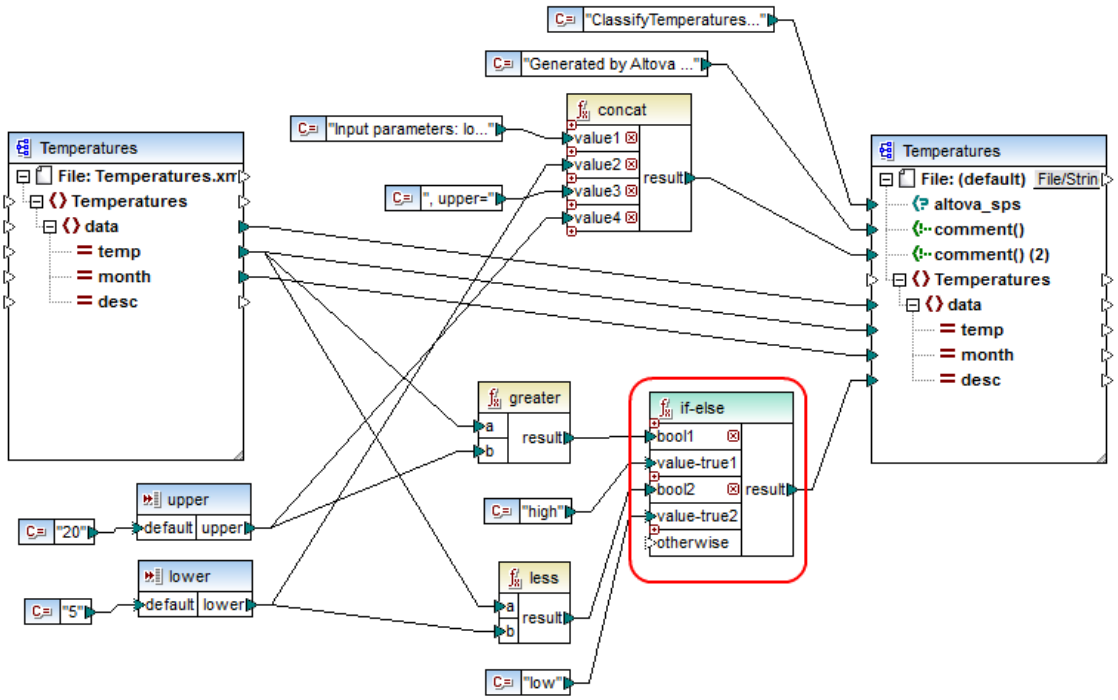
```
...
   <expense-item>
      <type>Meal</type>
      <Date>2003-01-01</Date>
      <expense>122.11</expense>
   </expense-item>
   <expense-item>
      <type>Lodging</type>
      <Date>2003-01-02</Date>
      <expense>299.45</expense>
   </expense-item>
...
```

マッピングの実行後のXML 出力

5.9.2 サンプル :条件付で値を返す

この例では、true/false 条件を基にして、コンポーネントから単純型の値を返す方法を説明しています。**If-Else 条件** ( ) がこの目的を達成するために使用されています。 **If-Else 条件** とフィルターコンポーネントを混同しないようご注意してください。**If-Else 条件** は、(文字列、整数などの) 単純な値を条件付きで処理する場合のみに適しています。ノードなどの複合値をフィルターする場合、代わりにフィルターを使用してください (以下を参照してください:例 : ノードのフィルター)。

この例で説明されるマッピングは以下のパスで検索することができます :**<Documents>\Altova\MapForce2017 \MapForceExamples\ClassifyTemperatures.mfd**。



このマッピングは、気温データを含むソースXML ("Temperatures") からデータを読み取り、データを同じスキーマに準ずるターゲットXML に書き込みます。ターゲットとソースの間には複数の他のコンポーネントが存在します。そのうちの1つが、このトピックの主題である (赤でハイライトされている) **if-else** 条件です。

このマッピングのゴールは、ターゲット内の各気温の記録に短い説明を追加することです。具体的には、摂氏 20度以上の気温の説明は、 "high" と表示され、摂氏 5度以下の気温は、 "low" と表示されるように設定します。上記以外の場合は、説明が表示されません。

この目的を達成するために、条件付き処理が必要とされます。ですから、If-Else 条件 がマッピングに追加されました。(If-Else 条件を追加するには、 **挿入** メニューをクリックして、**If-Else 条件**をクリックします)。このマッピングでは、以下の2つの条件を受け入れるために (▫ ボタンを使用して) If-Else 条件が展開されています :**bool1** と**bool2**。

条件自身は、MapForce core ライブラリから追加された**greater** と**less** 関数により追加されます (詳細に関しては、以下を参照してください:関数と作業する)。 関数は、 "upper" と "lower" と呼ばれる2つの入力コンポーネントより与えられた値を評価します (入力コンポーネントを追加するには、 **挿入** メニューをクリックして、**入力の挿入**をクリックします。入力コンポーネントに関する詳細については、マッピングにパラメーターを提供するを参照してください

greater と**less** 関数は、true または false のどちらかを返します。関数の結果は、ターゲットインスタンスに何か書き込まれるかを決定します。具体的には、ソース内の "temp" 属性の値が、20 よりも大きい場合、定数値 "high" が**if-else** 条件にパスされます。ソース内の "temp" 属性の値が 5 よりも小さい場合、定数値 "low" は、**if-else** 条件にパスされます。**otherwise** 入力が接続されていませんので、上記の条件が満たされない場合は、**result** 出力コネクターにパスされる値はありません。

最後に、**result** 出力コネクタは、(各気温の記録に1度ずつ与えられる) この値をターゲット内の "desc" 属性に与えます。
マッピングの結果をプレビューする準備が整うと、 **出力** タブをクリックします。気温が20度より高い、または、5度より低い場合、結果 XML 出力は、 "desc" 属性を含むことに注意してください。

```
...
   <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"/>
...
```

マッピングの実行後のXML 出力

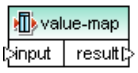
5.10 Value- Map の使用

Value-Map コンポーネントを使用することで、与えられた入力値から、ルックアップテーブルを使った別の値へ出力値を変換することができます。この機能は列挙型の変換を行う際に便利です。コンポーネントには 1 つの入力と 1 つの出力アイテムだけが含まれます。

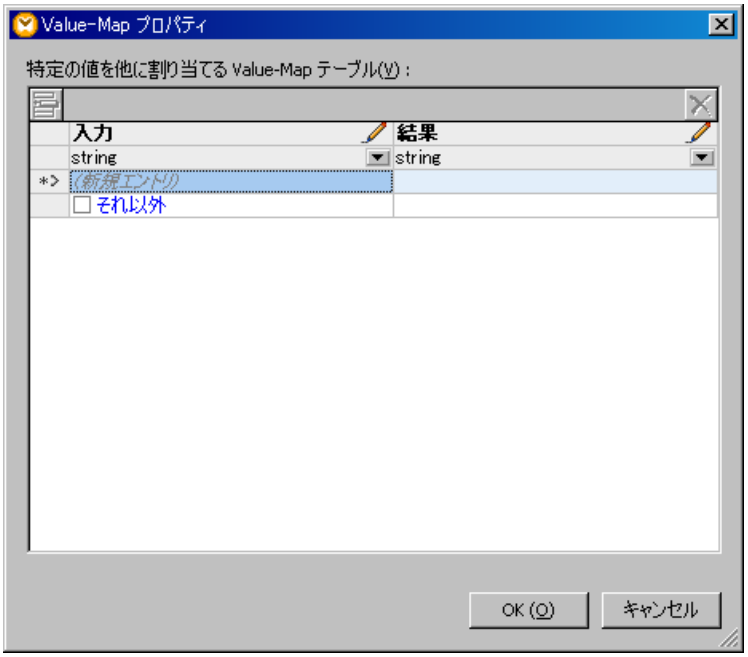
メモ:特定の条件に従ってデータの取得 (フィルタリング)を行いたい場合は、**フィルター**コンポーネントを使用してください。フィルターと条件のセクションを参照してください。

Value- Map コンポーネントを使用する:

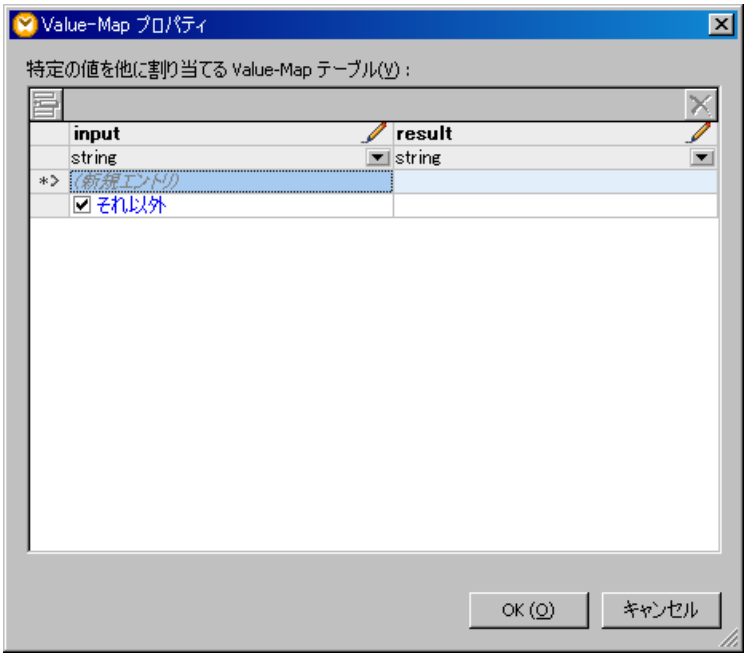
1. メニューオプションの「挿入 | Value-Map」を選択するか、アイコンバーにある Value-Map アイコン  をクリックします。



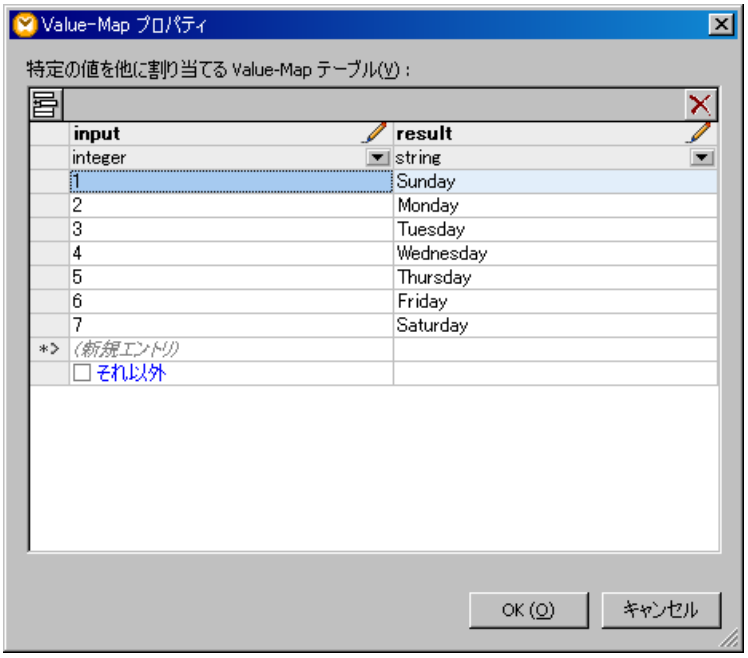
2. Value-Map コンポーネントをダブルクリックして、Value-Map テーブルを開きます。



3. C カラムヘッダーをクリックして、最初のカラムに **Weekday input** と、そして 2 つめのカラムに **Day of the Week** と入力します。



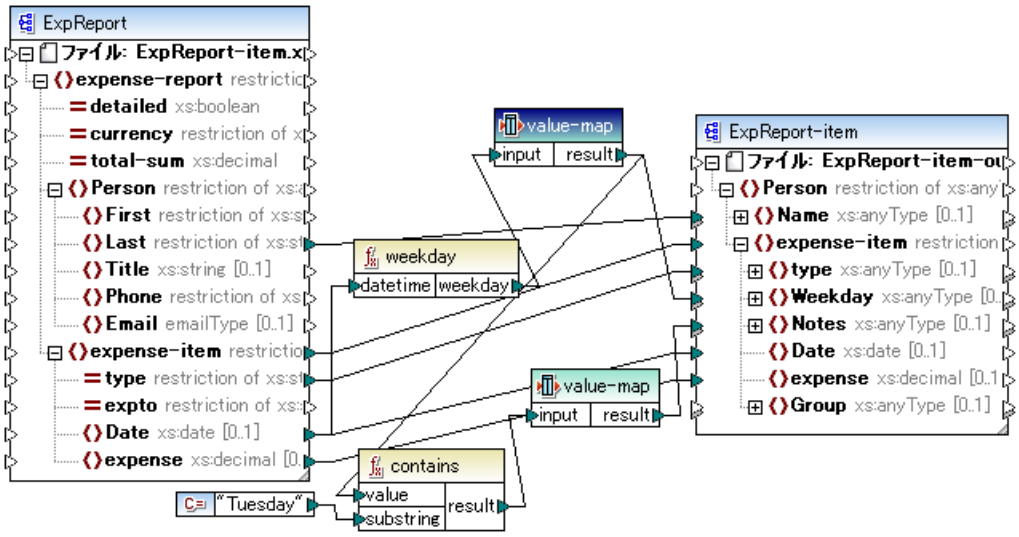
4. 変換の対象となる値を**Weekday input** カラムに入力します。
5. 変換後の値を**Day of the week** カラムに入力します。
6. 新たな値のペアを入力するには **新規エントリー)** 入力フィールドにて値を入力します。
7. カラムヘッダーの下にある**データ型**コンボボックスをクリックして、入力ならびに出力のデータ型を選択します (下の例では integer と string)。



メモ:**それ以外**のチェックボックスを選択することで、与えられた値がテーブルの値にマッチしない場合の代替出力値を定義することができます。ソースデータから与えられた値を変更すること無くパススルーさせる方法については Value-Map にて値を変更することなくパススルーさせるを参照してください。

8. ヘッダー行にある編集アイコンをクリックすることで、列名を変更することができます (列名はマッピングにも表示されます)。この操作により、マッピングにあるコンポーネントの目的を簡単に理解することができるようになります。

...\MapForceExamples\Tutorial\ フォルダーに収められている**Expense-valmap.mfd** ファイルにて、Value-Map を使用方法を示すためのサンプルマッピングを参照することができます。



このマッピングについて：
データソース内に収められているDate アイテムから、週の曜日を取得し、数値データをテキストへ変換した後、その値をターゲットコンポーネントのWeekday アイテムへ配置します。

- weekday 関数により、ExpReport ソースファイルのDate アイテムから週の曜日を取得します。この関数の戻り値は 1から 7までの数値 (nteger) となります。
- Value-Map コンポーネントにより、数値が $unday や Monday といった曜日に変換されます (上にあるスクリーンショットを参照)。
- 出力がTuesday である場合、"Prepare Financial Report" という文字列がターゲットコンポーネントの Notes アイテムへマッピングされます。
- 出力タブをクリックすることで、変換が行われたターゲットXML ファイルを確認することができます。

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

メモ：

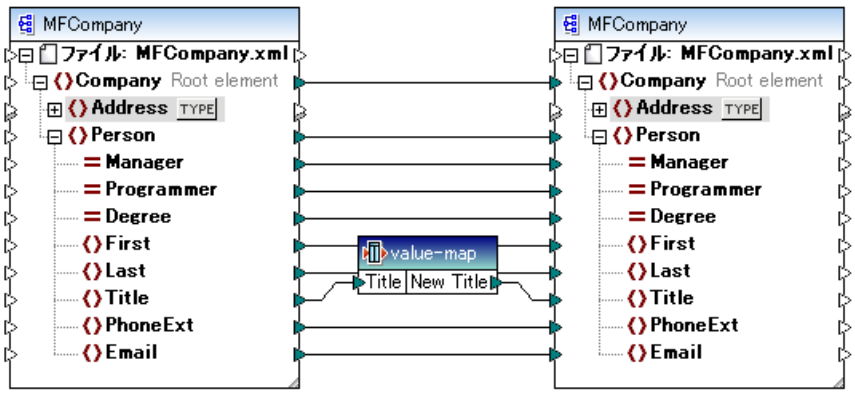
マウスカーソルをValue-Map コンポーネントの上に移動させることで、現在定義されている値を参照することができます。

論理または文字列関数から得られた出力は**"true"** または**"false"** 値となります。Value-Map テーブルの**入力**フィールドに入力する値は、例えば "true" という値になります。

5.10.1 値を変更せずに Value- Map を通過させる

このセクションでは、特定のノードデータを変換し、それ以外のノードデータをそのままのかたちでターゲットへ受け渡すといった状況について記述します。

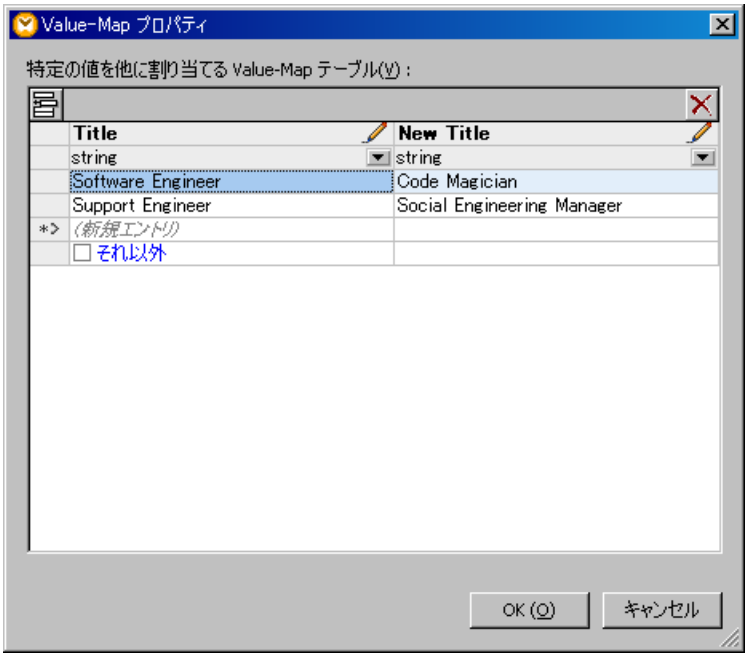
このような状況の例として、役職名を変更する会社内部の例を取り上げます。役職に関する箇所だけが変更され、それ以外はそのまま残すようにします。



マッピングは、を特定のタイトルを変換するためにvalue-map コンポーネントを使用する上のスクリーンショットのようになります。
「出力」タブをクリックすると、マッピングの結果が表示されます：

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

Value-Map コンポーネント内にある値のどれにも当てはまらない場合、Title 要素が出力ファイルにて削除されることになります。

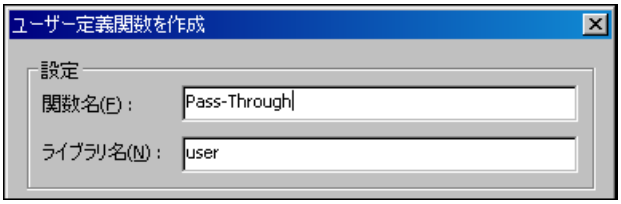


代替案：
それ以外チェックボックスをクリックして、代替となる値を入力することで、その値が出力ファイルに表示されることになります。しかしその場合、テーブルに入力されていない社内の役職全てが、**同じ値**で置き換えられてしまいます。

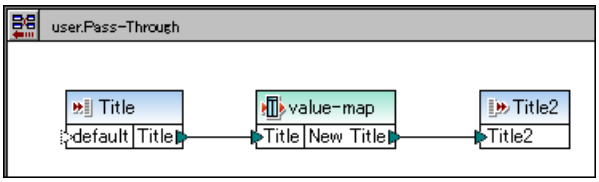
解決方法：

Value-Map コンポーネントを含むユーザー定義関数を作成し、**substitute-missing** 関数を使用することで空のノードに対してオリジナルのデータを渡します。

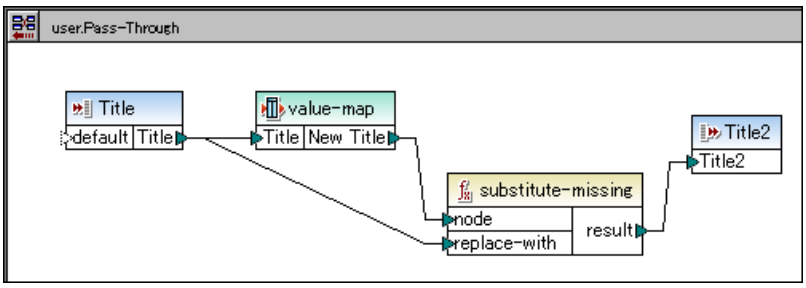
1. Value-Map コンポーネントを選択し、「関数 | 選択からユーザー定義関数を作成」を選択します。



2. 関数名にPass-Through と入力し、「**OK**」をクリックします。



3. ライブラリペインの**core | node 関数** にある**substitute-missing** 関数を挿入し、以下のスクリーンショットにあるような接続を行います。



4. 出力ボタンをクリックして結果を確認します。

マッピングの結果：

- Value-Map コンポーネントにある 2 つの役職がマッピングの結果に変換されます。
- ソースファイルにあるその他の役職が、そのままのかたちでターゲットファイルへ書きこまれます。

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

動作の説明：
Value-Map コンポーネントは入力されたデータを評価します。

- 入力データが Value-Map の最初のカラムにある値の**どれかとマッチ**した場合、データの変換が行われ、その値が substitute-missing 関数のパラメーターへ渡され、その後 Title2 へと渡されます。

- 入力データがカラムのデータに**マッチしない**場合、Value-Map コンポーネントの出力からノードパラメーターへ値が渡されることはなく、**空のノード**が渡されます。

  空のノードが発生した場合、substitute-missing 関数によりオリジナルのノードが Title ノードから取得され、**replace-with** パラメーターへ渡された値が Title2 へと渡されます。

5.10.2 Value- Map コンポーネントプロパティ

動作：

 現在アクティブな行の前に新たな行を**挿入**します。

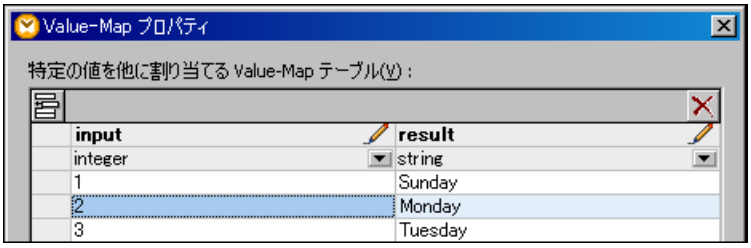
 現在アクティブな行を**削除**します

 カラムヘッダーの**編集**を行います。

行をドラッグすることで順番を変更することもできます。

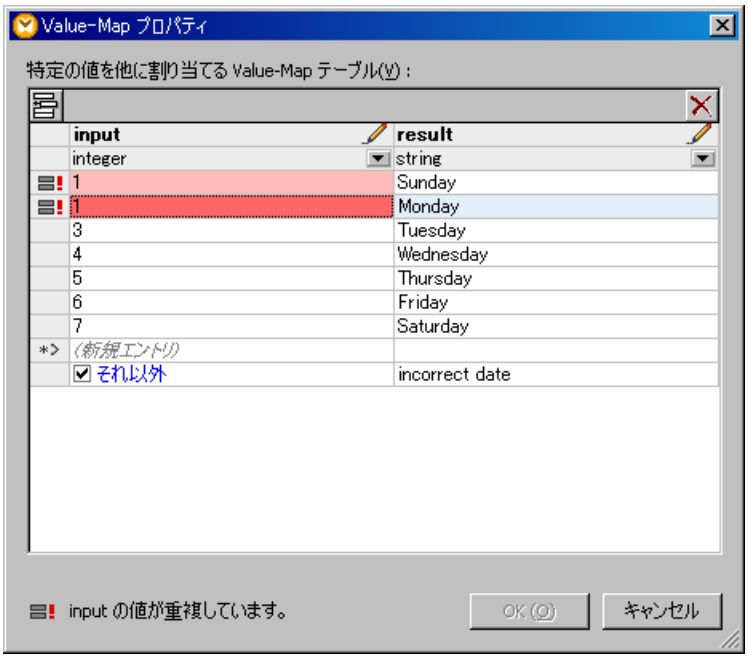
カラムのヘッダーを変更する：

カラムヘッダーをダブルクリックするか鉛筆アイコンをクリックすることで、列名を編集し、より分かりやすい名前に変更することができます。列名はマッピングにも表示されるため、この操作によりコンポーネントの目的を簡単に理解することができるようになります。



ユニークな入力値を使用する：:

入力カラムに入力される値はユニークでなければなりません。同じ値を 2 回以上入力した場合、その値がハイライトされ、修正するよう促されます。

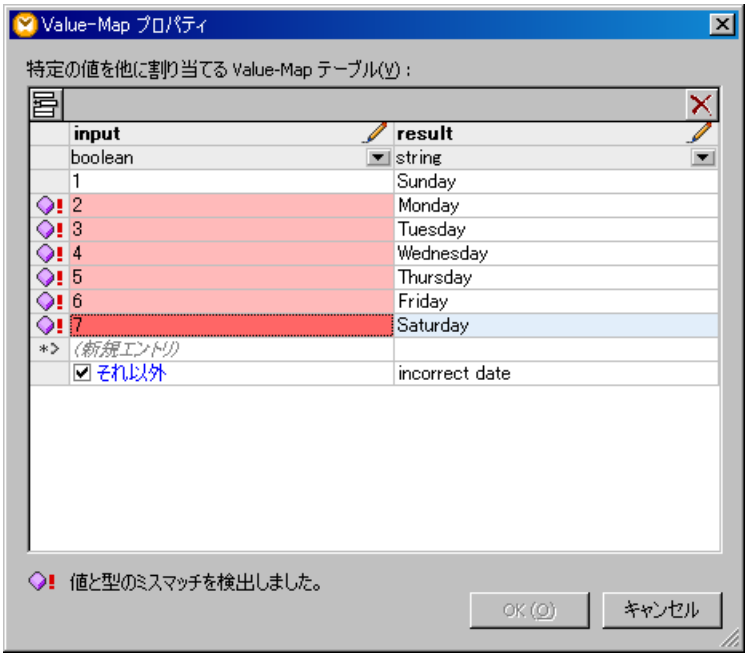


値を修正すると、「OK」ボタンが再度選択可能になります。

入力ならびに出力データ型

コンボボックスを使ってデータ型が選択された際に、入力ならびに出力データ型が自動的にチェックされます。ミスマッチが発生した場合、該当するフィールドがハイライトされ、「OK」ボタンが選択不可能になります。値をサポートするデータ型へ変更してください。

以下のスクリーンショットでは、boolean とstring が選択されています。


Value-Map プロパティ
特定の値を他に割り当てる Value-Map テーブル(V):
input
result
boolean
string
1
Sunday
2
Monday
3
Tuesday
4
Wednesday
5
Thursday
6
Friday
7
Saturday
(新規エントリ)
それ以外
incorrect date
値と型のミスマッチを検出しました。
OK (O)
キャンセル

5.11 ノード名のマッピング

MapForce を使用して、マッピングを作成すると、多くの目的はソースから値を読み取り、ターゲットに値を書き込むことです。しかしながら、ソースからだけではなく、ノード名から値を取得することを希望する場合があります。例えば、ソースXMLから(値ではなく)要素または属性の名前を読み取り、ターゲットXML内で 名前では無く要素または属性の値 に変換することを希望する場合があります。

以下の例を考慮してください：XML ファイルが製品のリストを含む場合、各製品は次のフォーマットを有します：

```
<product>
   <id>1</id>
   <color>red</color>
   <size>10</size>
</product>
```

目的は、各製品の情報を名前と値のペアに変換することです。例：

```
<product>
   <attribute name="id" value="1" />
   <attribute name="color" value="red" />
   <attribute name="size" value="10" />
</product>
```

このようなシナリオには、マッピングからのノード名へのアクセスが必要になります。このトピックの主題であるノード名への動的なアクセスにより、このようなデータ変換を行うことができます。

メモ: node-name と static-node-name コアライブラリ関数を使用して、上記の変換を行うことができます。しかしながら、この場合は、ソースから期待する正確な要素名が必要であり、ターゲットに手動で要素を接続する必要があります。また、これらの関数は、十分ではない場合があります。例えば、名前別にフィルター、または、グループ分けを行う場合、また、ノードマッピングからデータ型を操作する場合などが挙げられます。

ノード名への動的なアクセスは、ノード名を読み取る場合だけではなく、書き込む必要がある場合にも使用することができます。標準マッピングでは、基となるコンポーネントのスキーマからくる、ターゲット内の属性または要素の名前は、マッピングの実行前に既知です。動的なノード名を使用して、マッピングの実行前に既知ではない、新規の属性または要素を作成することができます。具体的には、属性または要素の名前が、MapForce によりサポートされるソースからマッピング自身により与えられます。

> ノードの子要素、または、属性への動的なアクセスを可能にするには、ノードが実際に子要素または属性を持つ必要があり、XML ルートノードであってはなりません。

動的なノード名は、次のコンポーネント型に、または、型からマッピングをする際にサポートされています：

- XML
- CSV/FLF*

* MapForce Professional または Enterprise Edition が必要です。

ノード名への動的なアクセスは次のマッピング言語によりサポートされています：BUILT-IN、XSLT2、XQuery、C#、C++、Java。*

* MapForce Professional または Enterprise Edition が必要です。

ノードへの動的なアクセスに関する情報は、以下を参照してください：ノード名へのアクセスを取得する。順序を追ったマッピングの例に関しては、以下を参照してください：例：要素名を属性値にマップする。

5.11.1 ノード名へのアクセスを取得する

XML コンポーネント内のノードに子ノードが存在する場合、マッピングの名前と各子ノードの値をマッピングで直接取得することができます。このテクニックは、「動的なノード名」へのアクセスと呼ばれます。「動的」とは、処理がランタイム中に素早く行われることを意味し、マッピングが実行される前に既知の静的なスキーマ情報をベースに行われます。このトピックは、ノード名への動的なアクセスの有効化の方法と、その使用方法について説明します。

ソースからデータを読み取る場合、「動的なノード名」とは以下を行うことができることを意味します：

- ノードの全ての子ノード（または属性）のリストをシーケンスとして取得します。MapForce では、"シーケンス" は、ゼロのリスト、または、ターゲットに接続することのできるアイテムであり、ソース内にアイテムが存在すると、ターゲットに同じ数のアイテムを作成します。ですから、例えば、ノードがソース内に5つの属性を持つ場合、ターゲット内に属性に対応する5つの新規の要素を作成します。
- （標準マッピングが行う）子ノードの値を読み取るだけではなく、名前も読み込みます。

ターゲットにデータを書き込む場合、「動的なノード名」は、以下を行うことができることを意味します：

- コンポーネント設定（いわゆる「静的」名前）により与えられた名前とは対照的な、マッピング（いわゆる「動的」名前）により与えられる名前を使用して、新しいノードを作成します。

動的なノード名を説明するために、このトピックは次の XML スキーマを使用しています：**<Documents>\Altova\MapForce2017\MapForceExamples\Tutorial\Products.xsd**。このスキーマは、サンプルインスタンスドキュメント **Products.xml** を伴っています。スキーマとインスタンスファイルをマッピングエリアに追加するには、**「挿入| XML スキーマ/ファイル」**メニューコマンドを選択して、**<Documents>\Altova\MapForce2017\MapForceExamples\Tutorial\Products.xml** を参照してください。ルート要素を選択するように問われると、`products` を選択します。

`product` ノードのための動的なノード名を有効化するには、右クリックし、次のコンテキストメニューコマンドの1つを選択します：

- ノードの属性にアクセスする場合、**「動的な名前を持つ属性を表示」**
- ノードの子要素にアクセスする場合、**「動的な名前を持つ子要素を表示」**

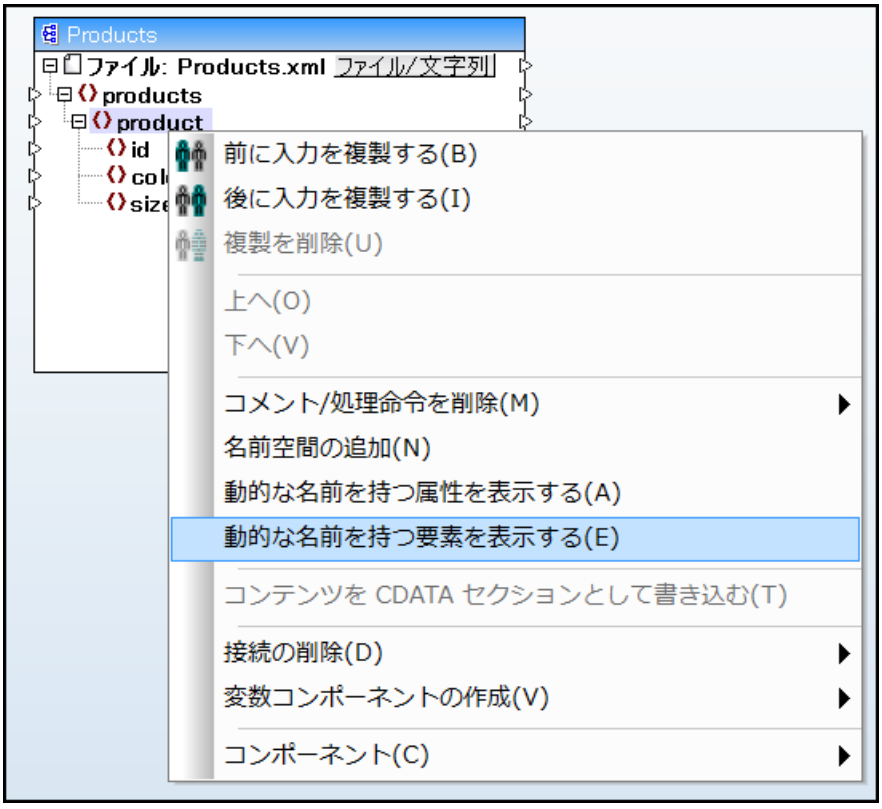


図 .1 (子要素のための)動的なノード名の有効化

メモ: 上のコマンドは、子ノードを持つノードのみに対して使用することができます。また、コマンドはフットノードには使用することができません。

ノードを動的なモードに切り替えると、下に表示されるダイアログボックスに類似したダイアログボックスが表示されます。このトピックの目的のために、オプションを下に表示されるように設定します。これらのオプションは特定の型のノードにアクセスする内で説明されています。

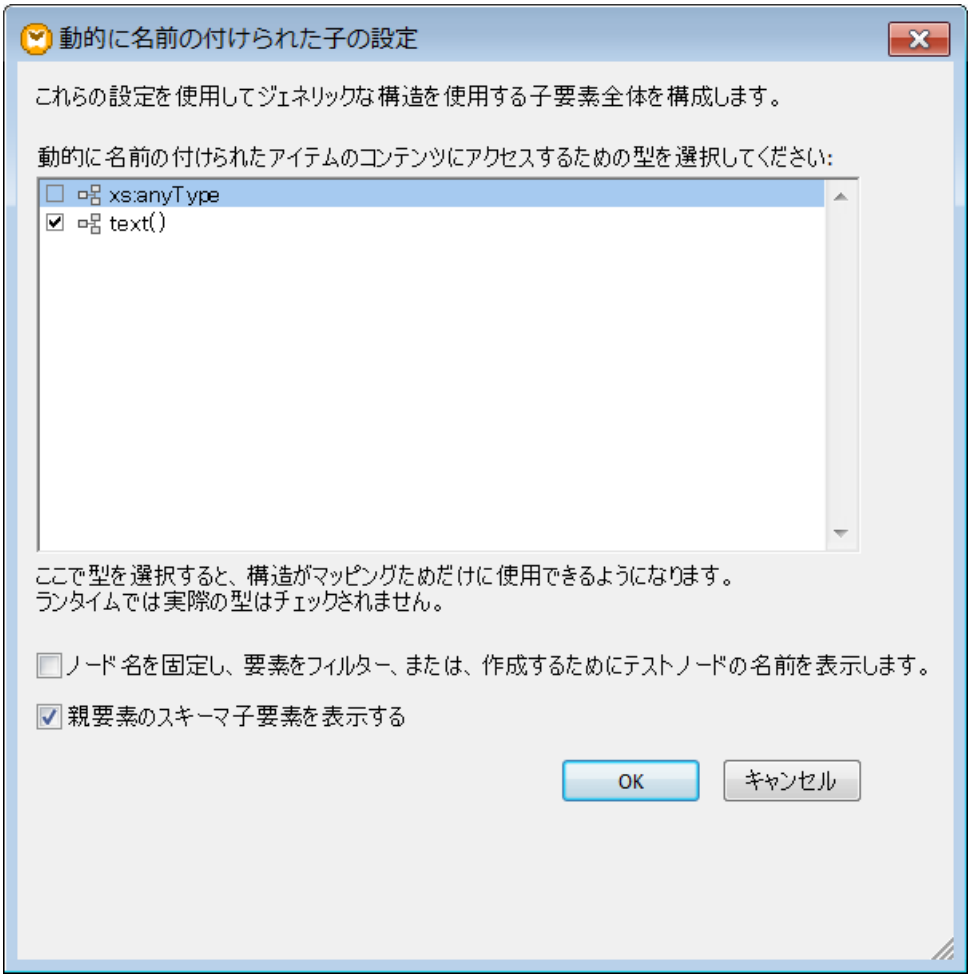


図 2　　動的に名前が付けられた子の設定」ダイアログボックス

図 3は、product ノードのために動的なノード名が有効化されている場合、どのようにコンポーネントが表示されるかを示しています。コンポーネントの外観が大幅に変更されていることにご注意してください。

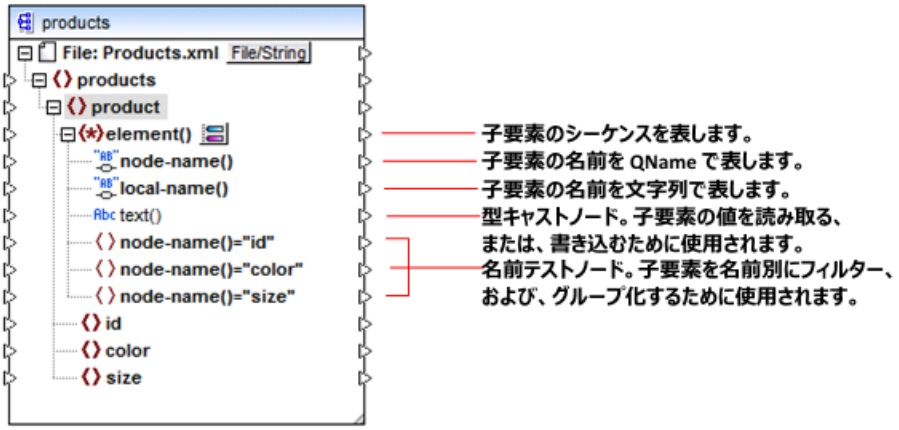


図 3　　(要素のために)動的なノード名が有効化されている場合

コンポーネントを標準モードに切り替えるには、product ノードを右クリックして、コンテキストメニューからオプション **動的な名前を持つ子要素を表示する** を無効化します。

下のイメージは、ノードの属性への動的なアクセスが有効化された時、同じコンポーネントがどのように表示されるかを示しています。product 要素を右クリックして、コンポーネントが取得され、コンテキストメニューから **動的な名前を持つ属性を表示する** を選択します。

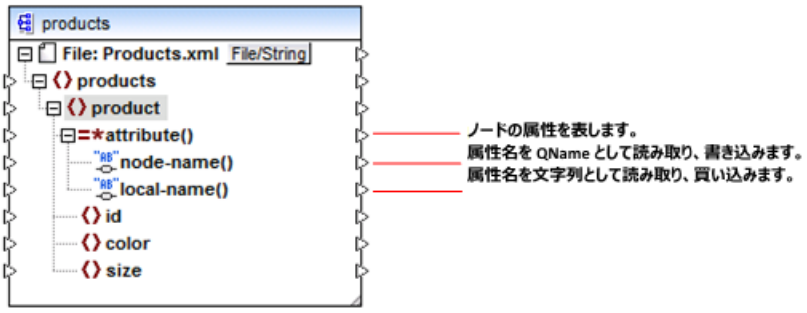


図 4 (属性のために)動的なノード名が有効化されている場合

コンポーネントを標準モードに切り替えるには、product ノードを右クリックして、コンテキストメニューから、 **動的な名前を持つ属性を表示する** オプションを無効化します。

図 3と図 4に示されているように、コンポーネントの外観はノード (この場合は、product) が「動的なノード名」なモードに切り替えられると変更されます。新しい外観は、次のアクションを取る可能性を広げます:

- すべてのノードの子要素または属性のリストを読み取り、または書き込みます。これらは、element() または attribute() アイテムによりそれぞれ提供されます。
- 各子要素または属性の名前を読み取り、または書き込むます。名前は、node-name() とlocal-name() アイテムにより提供されます。
- 要素の場合、各子要素の値を特有のデータ型として読み取る、または書き込みます。この値は、型キャストノード (この場合は、xs:string アイテム)により与えられます。要素のみが型キャストノードを持つことができることに注意してください。属性は、常に、「文字列」型として処理されます。
- 名前別に子要素をグループ化します。

「動的なノード名」モード内で作業することのできるノード型は、下で説明されています。

element()

ターゲットコンポーネントと比較すると、このノードはソースコンポーネント内で異なる振る舞いをします。ソースコンポーネント内で、ノードの子要素をシーケンスとして提供します。図 3element()は、全てのproduct の子要素のリスト(シーケンス)を与えています。例えば、次のXML から作成されたシーケンスは、3つのアイテムを含みます (これは、product 3つのつの子要素が存在するからです):

```
<product>
   <id>1</id>
   <color>red</color>
   <size>10</size>
</product>
```

シーケンス内の各アイテムの実際の名前 と型は、node-name() ノード と型キャストノードによりそれぞれ与えられています 上記を理解するには、ソースからのデータXML をターゲットXML に次のように変換する必要があると仮定してください:

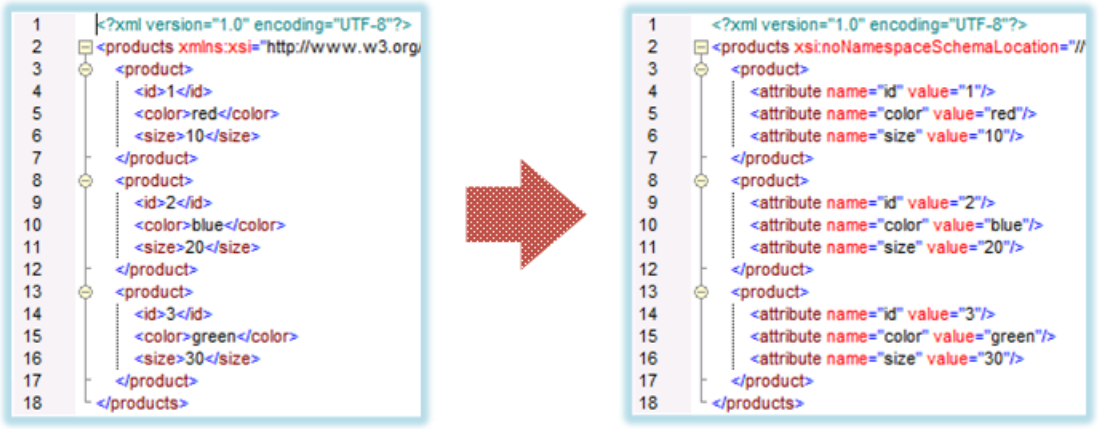


図 6　シーケンス内の各アイテムの実際の名前 と型は、node-name() ノード と型キャストノードによりそれぞれ与えられています 上記を理解するには、ソースからのデータXML を ターゲットXML に次のように変換する必要があると仮定してください:

マッピングの目的を達成するマッピングは以下のようになりますマッピング:

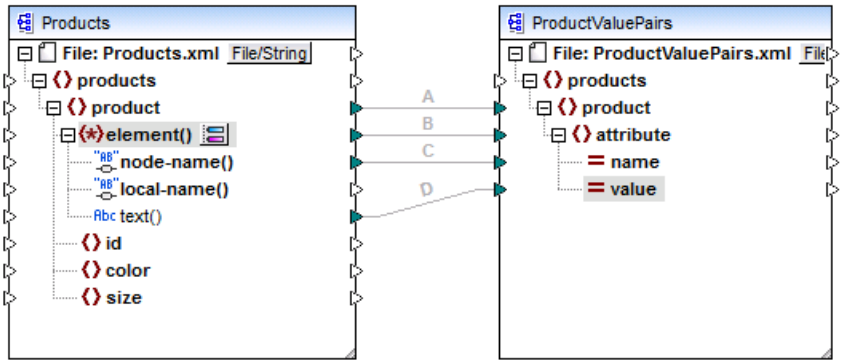


図 7　(MapForce 内で)XML 要素名 を属性値 にマッピングする

node-name() と text() は、シーケンス内の各アイテムの値と実際の名前を与えますが、element() のここでの役割は、product の子要素のシーケンスを提供します。このマッピングは、チュートリアルのサンプルが付随しており、以下で更に詳しく説明されています: 例:要素名を属性値にマップする。

ターゲットコンポーネント内では、element() は、ソース内の各アイテムのマッピングの基本ルールの例外であるとおり自身では何も作成しませんが、1つのターゲットアイテムを作成します。実際の要素 は、(node-name()の値を使用して)型キャストノード と(自身の名前を使用して)名前テストノードにより作成されます。

attribute()

図 4内で表示されているように、このアイテムは、マッピングのランタイムでノードの全ての属性へのアクセスを有効化します。ソースコンポーネント内で、接続されたソースノードの属性をシーケンスとして提供します。例えば、次のXML 内で、シーケンスは、(product には2つの属性が存在するため)2つのアイテムを含みます:

```
<product id="1" color="red" />
```

attribute() ノードは、シーケンス内の各属性の値のみを文字列の型として与えます。各属性の名前はnode-

name() ノードにより与えられます。

ターゲットコンポーネント内では、このノードは、接続されたシーケンスを処理し、シーケンス内の各アイテムのために属性の値を作成します。属性の名前は、node-name() により与えられます。例えば、ソースからのデータXML をターゲット XML を以下のように処理すると想定します：

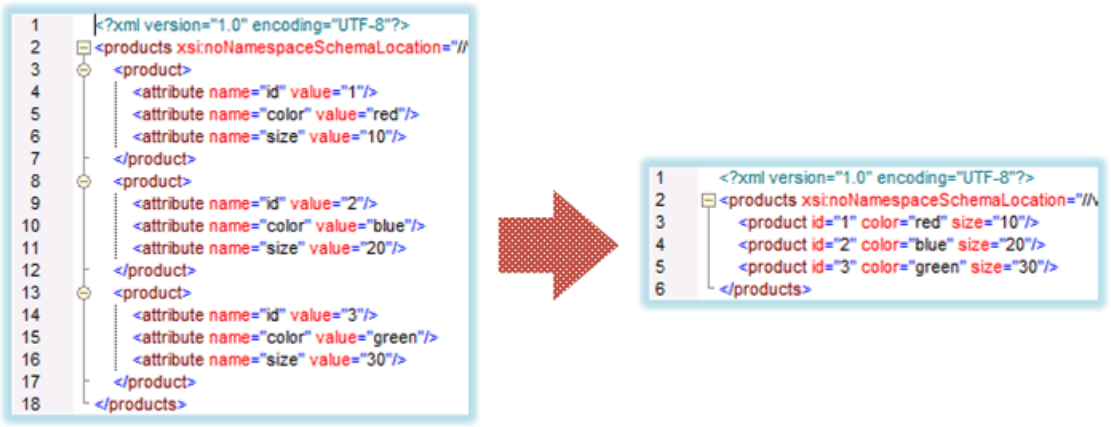


図 8 属性の値を属性名にマッピングする（必須）

この目的を達成するマッピングは以下のようになります：

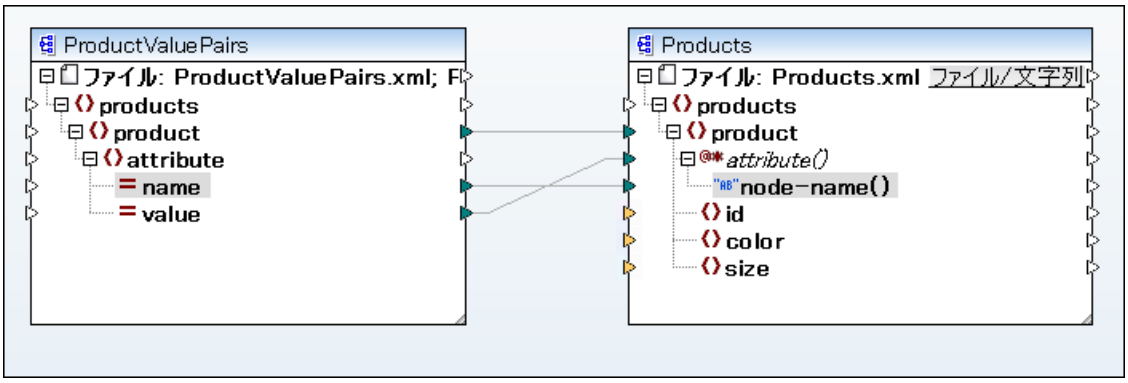


図 9 (MapForce 内で属性の値を属性名にマッピングする

メモ: この変換は、ノードの属性へのアクセスを有効化せずに行うことができます。ここでは、どのようにattribute()がターゲットコンポーネント内で動作するかを表示しています。

このマッピングを再作成する場合は、**<Documents>\Altova\MapForce2017\MapForceExamples\Tutorial** フォルダー内の**ConvertProducts.mfd** マッピングと同じXML コンポーネントを使用しています。マッピングとして使用します。唯一の違いは、ターゲットがソースになり、ソースがターゲットになることのみです。ソースコンポーネントのための入力データには、属性の値を実際に含むXML インスタンスが必要になります。例：

```
<?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>
```

上記のコードリスティングでは、名前空間とスキーマ宣言が、ここでは簡単にするために省略されていることに注意してください。

node-name()

ソースコンポーネント内では、node-name() は、element()の子要素の名前、または、attribute()の要素の名前をそれぞれ提供します。デフォルトでは、提供される名前は、xs:QName の型です。名前を文字列として取得するには、 local-name() ノードを(図 3を参照してください)、を使用します。

ターゲットコンポーネント内ではnode-name() は、element() またはattribute()内に含まれる各要素または属性の名前を書き込みます。

local-name()

このノードは、node-name() と同様の作動をしますが、異なる点は、xs:QName の代わりにxs:string が使用されていることです。

型キャストノード

ソースコンポーネント内では、型キャストノードは、element() 内の各子要素の値を提供しますこのノードの名前と構造は、「動的に名前がつけられた子の設定」ダイアログボックス (図 2)から選択された型により異なります。

ノードの型を変更する場合は、 **「選択の変更」**( ) ボタンを使用して、スキーマワイルドカード (xs:any) を含む、使用できる型から希望する型を選択します。詳細に関しては、次を参照してください:特定の型のノードにアクセスする。

ターゲットコンポーネント内で、型キャストノードは、element() 内に含まれる各子要素の値特有のデータ型として書き込みます。希望するデータ型は **「選択の変更」**( ) ボタンをクリックすることにより選択することができます

名前テストノード

ソースコンポーネント内では、名前テストノードは、ソースインスタンスから名前別に子要素をグループ化、または、フィルターする方法が提供されています。正確な型を使用してマッピングが、インスタンスデータにアクセスしていることを保証するために子要素を名前別にフィルターする必要がある場合があります。 (次を参照してください:特定の型のノードへのアクセス)。

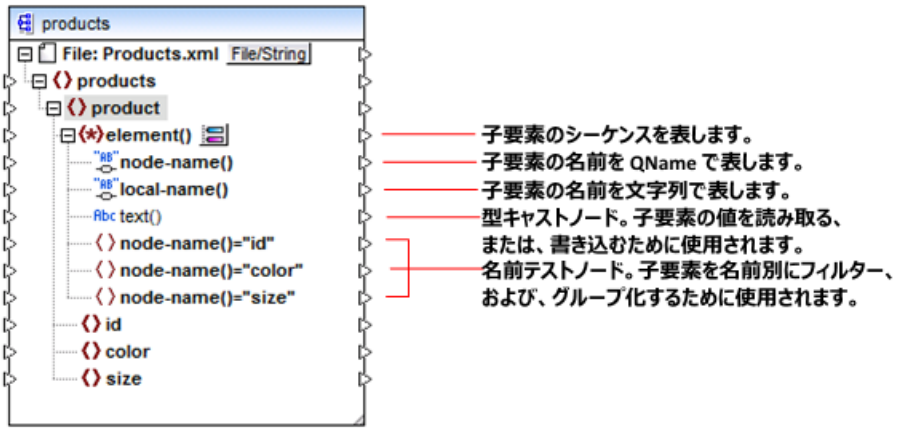
一般的には、名前テストノードは、値とサブ構造の読み取りと書き込みをする通常の要素ノードとほぼ同様の作動をします。しかしながら、動的なアクセスが有効化されていると、マッピングのセマンティクスが異なるため、制限が発生します。例えば、2つの名前テストノードを連結することはできません。

ターゲット側では、名前テストノードは、接続されているソースシーケンスにアイテムが存在するため、出力内と同じ数量の要素を作成します。これらの名前は、node-name()にマップされている値を上書きします。

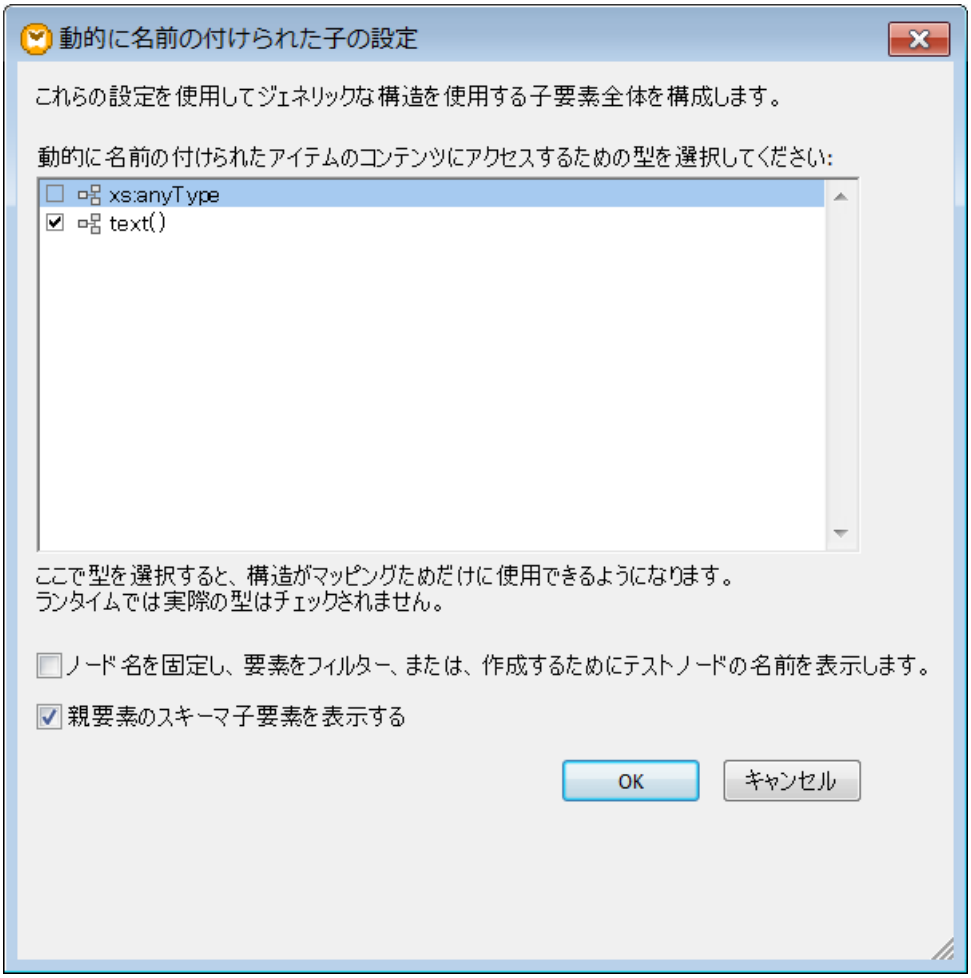
必要であれば、コンポーネントから名前テストノードを非表示にすることができます。 これを行うには、 element() ノードの横の **「選択の変更」**( ) ボタンをクリックします。そして、 「動的な名前を持つ子の設定」ダイアログボックスから、**「名前テストノードの表示」**チェックボックスのチェックを解除します。

5.11.2 特定の型のノードへのアクセス

前のセクションで説明されているとおり、[ノード名へのアクセスの取得](#) は、ノードを右クリックして **動的な名前を持つ子要素を表示する** コンテキストメニューコマンドを選択し、ノードの全ての子要素にアクセスすることができます。特定の型のノードにより値をアクセスすることはできますが、マッピングのランタイムでは、`node-name()` ノードから各子要素の名前にアクセスすることができます。下のイメージでは、型キャストノードは、`text()` ノードの横にあります。



子要素のデータ型は、マッピングのランタイム前にはわかりません。 また、各子要素により異なる可能性があります。例えば、XML インスタンス ファイル内の `product` ノードは、型 `xs:integer` の子要素 `id` と型 `xs:string` の子要素 `size` を持つ可能性があります。特定の型のノードコンテンツにアクセスするには、下に表示されるダイアログボックスがノードの子要素への動的なアクセスを有効化するたびに開かれます。`element()` ノードの横の **選択の変更** ( )ボタンをクリックすることにより、このダイアログボックスを後で開くことができます。



動的に名前がつけられた子の設定」ダイアログボックス

マッピングのランタイムに各子要素のコンテンツにアクセスするには、いくつかのオプションがあります：

1. コンテンツを文字列としてアクセスするには、上のダイアログボックスで**text()** チェックボックスを選択します。この場合、ダイアログボックスが閉じられると、`text()` ノードがコンポーネント上に作成されます。このオプションは、コンテンツが(`xs:int`、`xs:string` などの)単純型 の場合、適切です。例：要素名を属性値にマップするで詳しく説明されています。**text()** ノードは、現在のノードの子ノードがテキストを含むことができる場合表示されることに注意してください。
2. スキーマに許可されている特定の複合型としてコンテンツにアクセスすることができます。カスタム複合型が、選択されているのためにノードグローバルに許可されていると、上のダイアログボックスで使用することが可能になり、横のチェックボックスを選択することができます。上のイメージでは、グローバルに定義されている複合型は存在せず、この選択は使用することができません。
3. コンテンツを型としてアクセスする。これは、高度のマッピングのシナリオで役に立ちます（次を参照してください：更に深い構成にアクセスする」を参照してください）。これを行うには、**xs:anyType** の横のチェックボックスを選択してください。

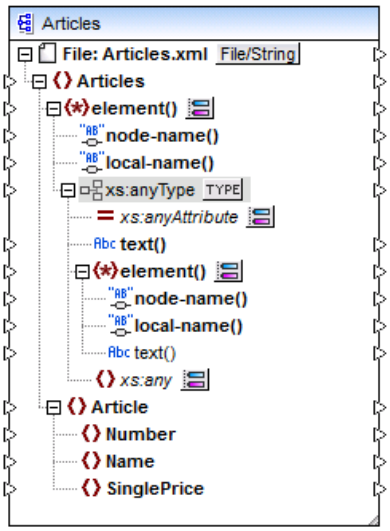
マッピング ランタイムでは、MapForce (型 キャストノードを介して)には、実際のインスタンスノード型に関する情報がないことに注意してください。ですから、マッピングは、正確な型を使用してノードコンテンツにアクセスする必要があります。例えば、ソースXML インスタンスが多種の複合型の子 ノードを持つことを希望する場合、以下を行います：
a) 型キャストノードを一致させる必要が有る複合型に設定します（上のリストの2番目のアイテムを参照してくださ

い)。
b) 一致する必要のあるインスタンスのみから読み込むためにフィルターを追加します。詳細に関しては以下を参照してください:フィルターと条件。

更に深い構成にアクセスする

スキーマワイルドカードの選択により、ノードをノードの直下の子よりもスキーマ内の更に深いレベルでアクセスすることができます。高度なマッピングのシナリオではとても役に立ちます。マッピングは、XML ノードの直下の子しかアクセスすることができないため、例:要素名を属性値にマップする等の簡単なマッピングでは、このテクニックを必要としません。しかしながら、動的に更に深い構造にアクセスする必要がある場合、例えば「孫」にアクセスが必要な場合、以下の方法でアクセスすることが可能です。

1. 新規マッピングを作成する。
2. 挿入メニューから**「XML スキーマ/ファイルを挿入する」**をクリックし、XML インスタンスファイルを参照します。(子の例では、**<Documents>\Altova\MapForce2017\MapForceExamples\Tutorial** フォルダーからの**Articles.xml** ファイルです)。
3. `Articles` ノードを右クリックして、**「動的な名前を持つ子要素を表示する」** コンテキストコマンドを選択します。
4. 「動的に名前がつけられた子の設定」ダイアログボックスから**xs:anyType** を選択します。
5. `xs:anyType` ノードを右クリックして、**「動的な名前を持つ子要素を表示する」**コンテキストコマンドに対して選択します。
6. 「動的に名前がつけられた子の設定」ダイアログボックスから**text()** を選択します。



上記のコンポーネントで、2つの`element()` ノードが存在することに注目してください。2番目の`element()` ノードは、**Articles.xml** インスタンス内の`<Articles>` ノードの孫に動的なアクセスを与えます。

```
<?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

例えば、「子」要素名（Number、Name、SinglePrice）を取得するには、2番目のelement()ノードの下のlocal-name()ノードからターゲットノードに接続を描きます。同様に、「子」要素の値（1、T-Shirt、25）を取得するには、text()ノードから接続を描きます。

この例には適用することはできませんが、実際のシナリオでは、更に深いレベルにアクセスするために、動的なノード名を後のxs:anyTypeノードに対して有効化することができます。

以下の点に注意してください：

- TYPEボタンを使用して、生成された型を現在のスキーマから選択し、異なるノード内に表示することができます。この方法は、生成されたスキーマの型から、または、スキーマの型へマップする必要がある場合のみに役に立ちます（次を参照してください：生成されたXMLスキーマ型）。
- element()ノードの横の**選択の変更**」（）ボタンは、子のトピックで説明されている「動的に名前の付けられた子の設定」ダイアログボックスを開きます。
- xs:anyAttribute属性の横の**選択の変更**」（）ボタンにより、スキーマ内でグローバルに定義された属性を選択することができます。同様に、xs:any要素の横の**選択の変更**」（）によりスキーマ内でグローバルに定義されている要素を選択することができます。これにより、スキーマワイルドカードへ、または、スキーマワイルドカードからのマッピングと同じように動作します（次も参照してください：ワイルドカード -xs:any / xs:anyAttribute）。このオプションを使用する場合、選択された属性または要素がスキーマに従い特定のレベルで存在できることを確認してください。

5.11.3　サンプル：要素名を属性値にマップする

この例は、XMLドキュメントから、要素名をターゲットXMLドキュメント内の属性の値にマップする方法について説明しています。この例には、付随するサンプルマッピングがあり、次のパスを使用して検索することができます：**<Documents>\Altova\MapForce2017\MapForceExamples\Tutorial\ConvertProducts.mfd**。

例を理解するために、XMLファイルは製品のリストを含むと想定しましょう。各製品は次のフォーマットを有します：

```
   <product>
      <id>1</id>
      <color>red</color>
      <size>10</size>
   </product>
```

目的は、各製品の情報を名前と値ペアに変換することです。例：

```
<product>
   <attribute name="id" value="1" />
   <attribute name="color" value="red" />
   <attribute name="size" value="10" />
</product>
```

上記のようなデータマッピングを行うには、この例でも使用されていますが、"ノード名への動的なアクセス"と呼ばれるMapFore 機能を使用しています。マッピングを実行する際、「動的」とは、(値だけではなく)ノード名も読み取ることができ、これらの名前を値として使用することを意味します。必要とするマッピングをいくつかのシンプルなステップを以下に示されるように作成することができます。

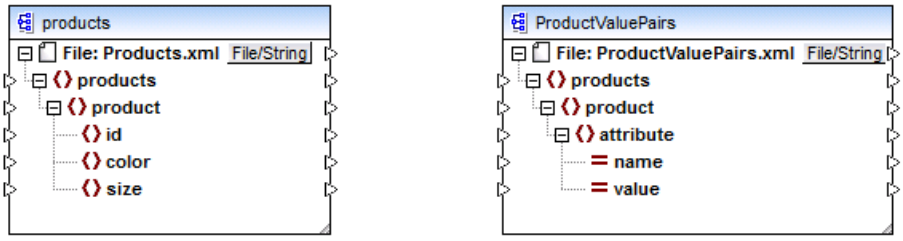
ステップ 1：ソース XML コンポーネントをマッピングに追加する

- **「挿入」**メニューから、**「XML スキーマ/ファイル」**をクリックして、**<Documents>\Altova\MapForce2017\MapForceExamples\Tutorial\Products.xml** を参照します。このXML ファイルは、同じフォルダー内にある**Products.xsd** スキーマを指します。

ステップ 2：ターゲット XML コンポーネントをマッピングに追加する

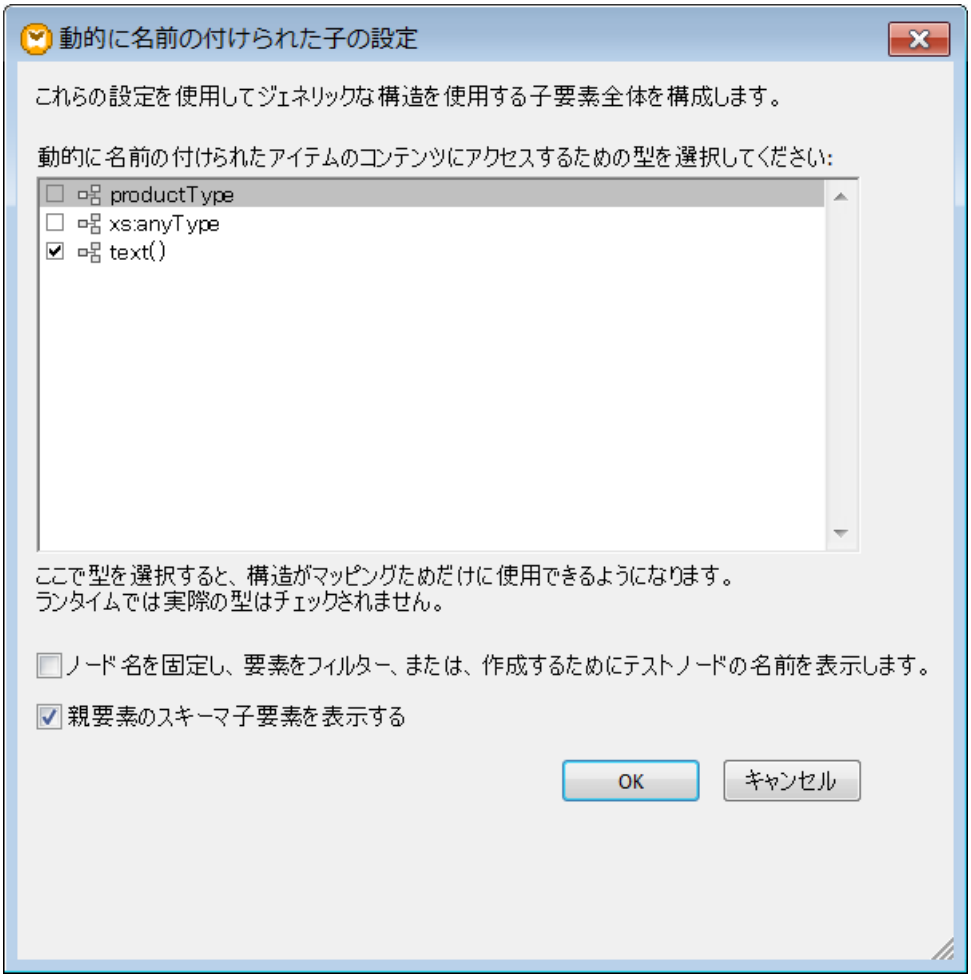
- **「挿入」**メニューから、**「XML スキーマ/ファイル」**をクリックして、次のファイルを参照します：**<Documents>\Altova\MapForce2017\MapForceExamples\Tutorial\ProductValuePairs.xsd**。インスタンスファイルを与えるように問われると、**「スキップ」**をクリックします。ルート要素を選択するようにプロンプトされると、ルート要素として、`products` を選択します。

この時点では、マッピングは、以下のようになります：

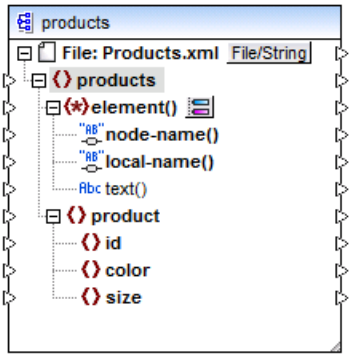


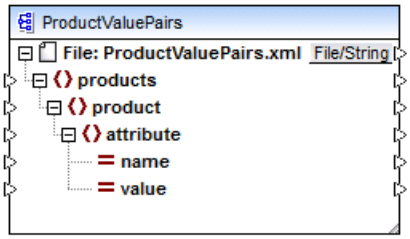
ステップ 3：子ノードへの動的なアクセスを有効化する

1. `products` ノードを右クリックして、コンテキストメニューから、**「動的な名前を持つ子要素を表示する」**を選択します。
2. 開かれるダイアログボックス内から、**text()** を型として選択します。他のオプションをそのままにします。



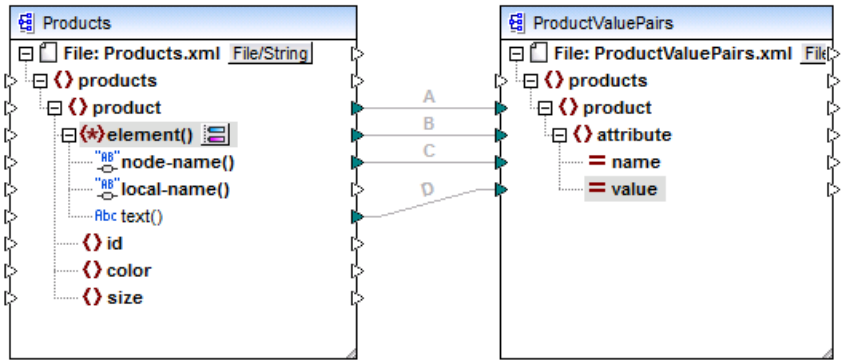
text() ノードがソースコンポーネントに追加されていることにご注意してください。このノードは、それぞれの子アイテムをマッピングに与えます（この場合は、「id」、「color」、「size」の値です）。





ステップ 4：マッピング接続を描く

マッピング接続 A、B、C、D 下に表示されるとおり、描きます。任意で、各接続上からをダブルクリックし、テキスト「A」、「B」「C」、および「D」をそれぞれ、詳細ボックスに入力します。



ConvertProducts.mfd

上で説明されているマッピングでは、接続 A は、ソース内の各製品のために、ターゲット内に製品を作成します。これまでは、これは ノード名を指さない標準的な MapForce 接続でした。しかしながら、接続 B は、`product` の各対応する子要素のために、新規の要素を`attribute` と呼ばれるターゲット内に作成します。

> 接続 B は、マッピング内でとても重要です。この接続の目的である`product` の子要素のシーケンスを、ソースからターゲットに運びます。実際の名前または値を運ばないため、以下として理解される必要があります：ソース**element()** にN 子要素が存在する場合、ターゲット内のアイテムの N インスタンスを作成します。この特定の場合は、ソース内の`product` には、3 つの子要素（`id`、`color` と`size`）が存在します。これは、ターゲット内の各 `product` が`attribute` という名前の3 つの子要素を持つことを意味します。
>
> この例で説明されていませんが、同じルールが、**attribute()** の子要素をマップするために使用されます。ソース **attribute()** アイテムが、N 子属性を持つ場合、ターゲット内のそのアイテムの N インスタンスを作成します。

次に、接続 C は、`product` の各子要素の実際の名前をターゲットにコピーします（文字通り、「id」、「color」、と「size」）。

最後に、接続 D は、製品の各子要素の値を文字列の型として、ターゲットにコピーします。

マッピングの出力をプレビューするには、**「出力」** タブをクリックして、生成された XML を確認してください。期待されるとおり、マッピングの意図とする目的である通り、出力は、データが名前と値のペアとして保管されている、複数の製品を含んでいます。

```
<?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>
```

生成されたマッピング出力

5.11.4 マッピングのルールと戦略

MapForce は、通常、データを直感的にマップしますが、出力の結果が、多すぎるアイテム、または少なすぎるアイテムを含む場合があります。このトピックでは、上記のような問題を避けるための説明がされています。

一般的なルール

通常、ソースとターゲットアイテム間のそれぞれの接続は、以下を意味します：各ソースアイテムのために、1つのターゲットアイテムが作成されます。ソース ノードが単純コンテンツ (例えば、文字列または整数) を含み、ターゲットノードが単純コンテンツを受け入れる場合、MapForce は、コンテンツをターゲットノードにコピーし、必要であれば、データ型を変換します。

次の例外を除いて、上記は全ての接続に対して当てはまります：

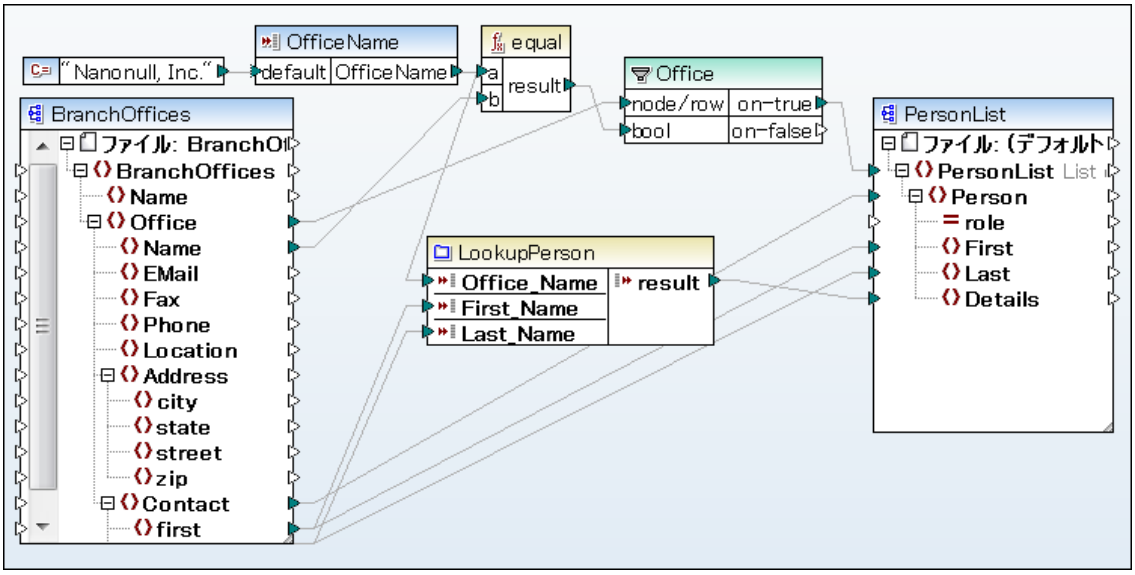
- ターゲット XML ルート要素は、通常、1度のみ作成されます。シーケンスをこの要素に接続すると、要素のコンテンツのみが繰り返され、ルートノード要素自身は繰り返されず、結果は、スキーマに対して有効でない可能性があります。ルート要素の属性はも接続されている場合、XML シリアル化は、ランタイムで失敗するため、ルート要素のシーケンスへの接続は回避されるべきです。複数の出力ファイルを作成して、達成する場合は、ファイル名を作成する関数を使用して、シーケンスを「ファイル」ノードに接続してください。
- ノードの一部は、シーケンスではなく単一の値を受け入れます。 (例えば、XML 属性、、、および ユーザー定義関数内の出力 コンポーネントなど)。

「コンテキスト」と「現在」の アイテム

MapForce は、スキーマ ファイルの構成を、コンポーネント内でマップすることのできるアイテムの階層として表示します。これらの各 ノードは、インスタンス ファイル、または、データベース内で、多数のインスタンスを有する場合があります。もしくは、まったくインスタンスを有さない場合もあります。

例：**PersonListByBranchOffice.mfd** 内のソース コンポーネントを確認してください、(**Contact** の下に単一の ノード **first** が存在します。**BranchOffices.xml** インスタンス ファイル内には、異なる **Office** 親 ノードの下に、異なるコンテンツを有する複数の **first** ノードと **Contact** ノードが存在します。

どのソース ノードが実際に選択され、データがターゲット コンポーネント アイテムにコネクタを使用してコピーされるかは (**ターゲット** ノードの) 現在の**コンテキスト**により決定されます。 .



PersonListByBranchOffice.mfd

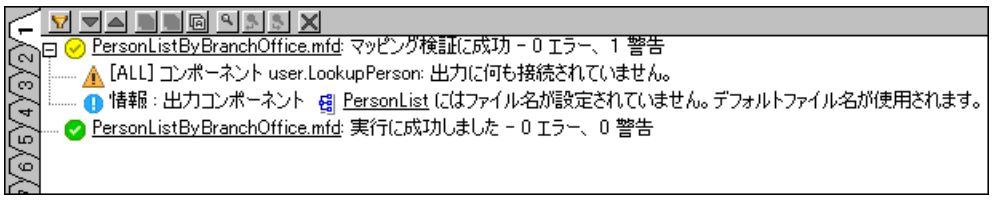
このコンテキストは、**現在のターゲットノード**により定義され、祖先に接続されます：

- 最初は、コンテキストは、ソースコンポーネントのみを含みますが特定のノードは含みません。マッピングを評価する際には、MapForce は、**ターゲット ルート** ノード (PersonList) を最初に処理します。そして、階層順に処理します。
- **ターゲット** ノードへのコネクタは、全てのソースアイテムを直接トレースすることができ、間接的に 2 つのコンポーネント間に存在する場合でもトレースすることができます。ソースアイテムと関数の結果は、このノードのためにコンテキストに追加されます。
- 各新しいターゲットノードに対して、最初は親ノードコンテキストの全てのアイテムを含む新しいコンテキストが確立されます。ターゲット兄弟ノードは、ですから、互いから独立していますが、親ノードの全てのソースデータにアクセスすることができます。

サンプルマッピング (**PersonListByBranchOffice.mfd**) に適用：

- **Office** からフィルター (Office) を介した**PersonList** への接続は、ターゲットドキュメント全体のためのコンテキストとして**単一の**オフィスをコンテキストとします。(これは、PersonList がターゲットコンポーネントのルート要素のためです)。オフィスの名前は、デフォルトで「Nanonull, Inc.」を含む入力コンポーネントにより与えられます。
- ルート要素 PersonList の**子孫**へのすべての接続 *データ*は、選択された単一のオフィスがコンテキスト内に存在するため、自動的にフィルターの条件により影響されます。
- **Contact** から**Person** への接続は、1 つのターゲット Person をソース XML (一般ルール) の**各** Contact アイテムのために作成します。
- **first** から**First** への接続は、現在の Contact の最初の名前を選択し、ターゲットアイテム First に書き込みます。

Contact から**Person** へのコネクタが作成されない場合、複数の First, Last、および Detail ノードを持つ 1 人の Person が作成されます。このような場合、MapForce は、警告を表示し、次のような問題を解決する提案をします：「Contact とPerson を接続して解決することができます」：



シーケンス

MapForce は、スキーマ ファイルの構造を、コンポーネント内のマップすることが可能な階層として表示します。

(ターゲット) コンテキストにより、ソース コンポーネントの各マップすることのできるアイテムは以下を表します :

- 割り当てられた入力 ファイル**単一のインスタンス** ノード
- 入力 ファイルのセットから**複数の インスタンス** ノードのシーケンス

シーケンスが**ターゲット** ノードに接続されている場合、ソース ノードの数と同数のターゲット ノードを作成するためのループが作成されます。

フィルター がシーケンスとターゲット ノードの間に存在する場合、ブール値の条件は、シーケンス内の各アイテムなどの各入力ノードのために チェックされます。具体的には、true を評価する各シーケンス内に少なくとも1つのブール値が存在するかがチェックされます。優先 コンテキスト 設定は、評価の順序を影響します。以下を参照してください。

上記の通り、フィルターの条件は自動的に全ての子孫 ノードに適用されます。

メモ: ソース スキーマが特定のノードが1度のみ発生することを指定する場合、MapForce は、ループを削除し、存在が既知の場合、最初のアイテムのみを取る場合があります。この機能は、ソース コンポーネント内の設定 ダイアログ ボックス (「min/maxOccurs をベースにした入力処理のオプティマイゼーション」チェック ボックス) 内で無効化することができます。

関数 入力 (または、ノーマルな 非 シーケンス 関数) は、ターゲット ノードと同様に作動します。シーケンスがこのような入力に接続されている場合、シーケンス内のアイテムと**同数** 作成されるようにループが関数の呼び出しの周りに作成されます。

シーケンスが**1つ以上** のそのような関数入力に接続されている場合、MapForce は、全ての入力の**デカルト積** を処理するネストされたループを作成します。通常これは、理想的ではないため、複数の (親または他のコンポーネントから単一の現在のアイテムにバウンドされている全ての他のパラメータなどの) アイテムを持つ単一のシーケンスのみを関数に接続してください。

メモ: 空のシーケンスがこのような関数 (例、concat) に接続されている場合、出力 ノードを生成しない、空のシーケンスが結果として出力されます。入力データが存在しないために、ターゲット 出力に結果がない場合、「substitute-missing」 関数 を使用して、値を代替することができます。

シーケンス入力 を持つ関数のみが入力シーケンスが**空** の場合、結果を生成することのできる関数です:

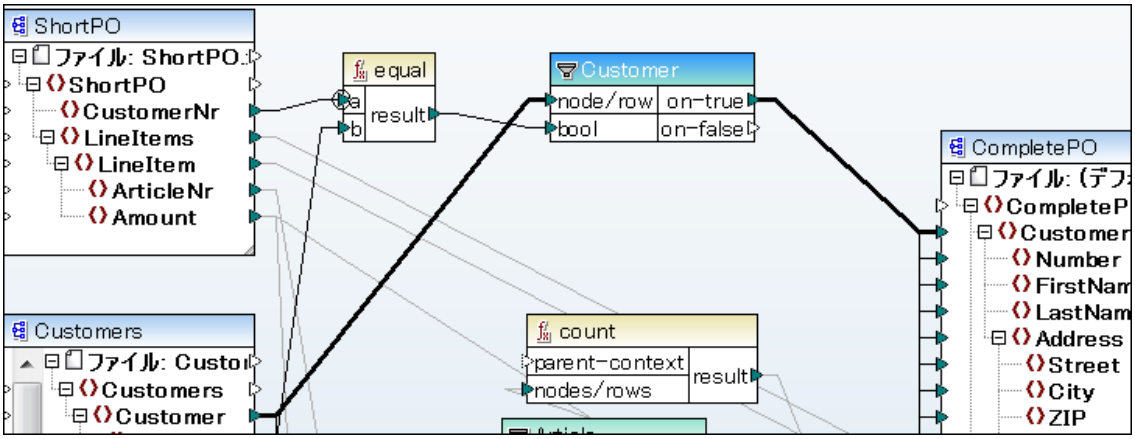
- `exists`, `not-exists` と`substitute-missing` (また、最初の3つのエイリアスである`is-not-null`, `is-null` と`substitute-null`)
- 集計 関数 (`sum`、`count` など)
- シーケンスを受け入れる通常のユーザー定義 関数 (例、非インライン関数)

このような関数のシーケンス 入力は、常に、祖先のコンテキスト内の現在のターゲット ノードとは別に評価されます。コンポーネント をフィルターする、関数は、他の接続に影響を与えません。

優先 コンテキスト

通常、関数 パラメータは、上から下に評価されますが、**優先 コンテキスト** 設定を使用して1つのパラメーターが他のパラメーターより先に評価されるように設定することができます。

filter 条件のブール値入力に接続されている関数では、優先 コンテキストは、比較関数自身のみではなく フィルターの評価にも影響を与えます。このため、2つのソースシーケンスを組み合わせることができます (次を参照 :CompletePO.mfd、CustomerNo および Number)。

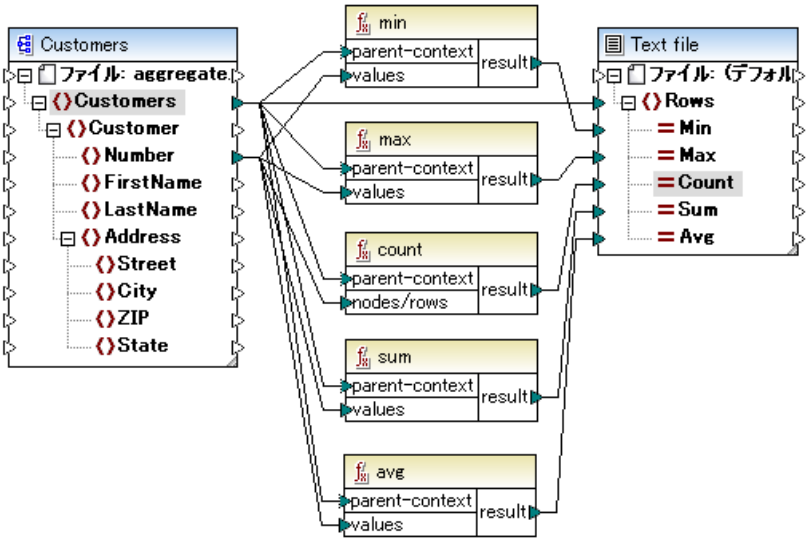


この例では、優先 コンテキストは、ShortPO/CustomerNr が、Customers コンポーネントからCustomer ノードを反復しフィルターする前に評価されるように強制します。 次も参照してください:優先コンテキストノード アイテム

コンテキストのオーバーライド

集計 関数 の一部には、任意 「親コンテキスト」 入力があります。

この入力が接続されていない場合、この関数は効果がなく 関数は、(ターゲットノードの親のコンテキスト内で) 通常のコンテキストでシーケンス入力のために評価されます。



parent-context 入力がソース ノードに接続されている場合、関数は、**各** 「parent-context」 ノードに対して評価され、個別の結果を各発生のために作成します。次も参照してください:マッピングコンテキストのオーバーライド。

同じ ソースコンポーネントの複数のノードをコンテキストに移動させる

これは、特定のケースで必要とされ、中間関数を使用して移動することができます。

5.11.4.1　マッピングコンポーネントの処理順序の変更

MapForce は、複数のターゲットコンポーネントを有するマッピングをサポートします。各ターゲットコンポーネントには、特定のコンポーネントのためのマッピングの結果をプレビューすることのできるプレビューボタンがあります。

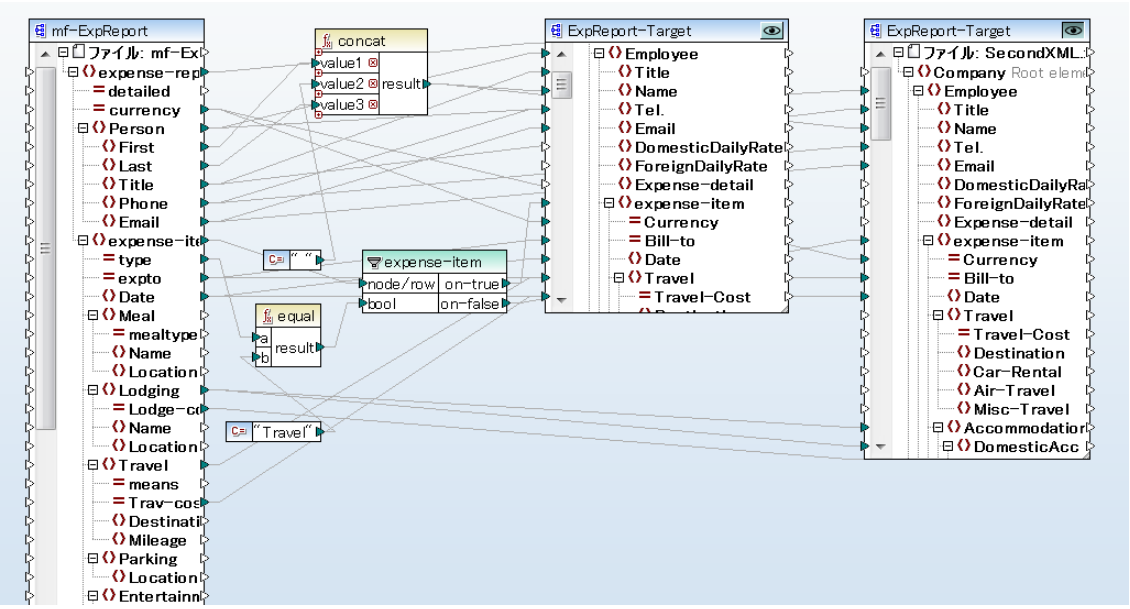
マッピングがコマンドライン、または、生成されたコードから実行されると、現在のアクティブなレビューに関わりなく、フルマッピングが実行され、各ターゲットコンポーネントのための出力が生成されます。

ターゲットコンポーネントが処理される順序は、マッピングウィンドウ内のターゲットコンポーネントの位置を変更することにより直接影響することができます。コンポーネントの**位置** は、左上と同じように定義されています。

ターゲットコンポーネントは、スクリーン上のY-X 位置に従い、上から下、左から右に処理されます。

- 2つのコンポーネントが、同じ垂直方向 位置を持つ場合、左端のコンポーネントが優先されます。
- 2つのコンポーネントが、同じ水平方向 位置を持つ場合、一番高い位置のコンポーネントが優先されます。
- コンポーネントが、完全に同じ位置を持つシナリオは少ないですが、定義されている順序を保証する、変更不可能な、ユニークな内部コンポーネントID が自動的に使用される場合はあります。

下のスクリーンショットは、**<Documents>\Altova\MapForce2017\MapForceExamples\Tutorial** フォルダー内のチュートリアルサンプル **Tut-ExpReport-multi.mfd** を表しています。両方の ターゲットコンポーネント (ExpReport-Target) は、同じ**垂直方向** 位置を有し、プレビューボタンが右のターゲットコンポーネント上でアクティブ化されています。



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

XSLT2 を選択して、コードを生成した場合：

- 左端のターゲットコンポーネントが最初に処理され、**ExpReport.xml** ファイルを生成します。
- 右のコンポーネントが次に処理され、**SecondXML.xml** ファイルを生成します。

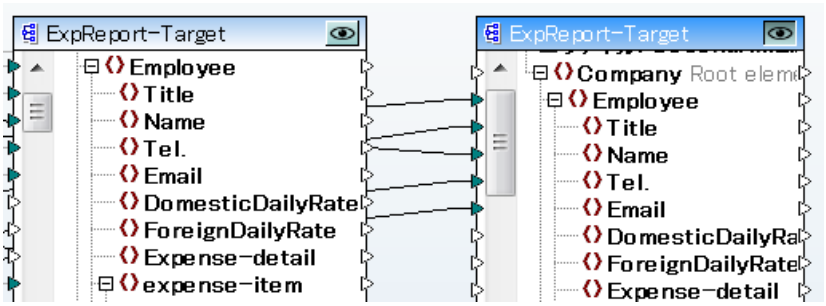
DoTransform.bat ファイル (指定された出力 フォルダー内で)開いて、生成されるシーケンス出力ファイルを確認することができます。**ExpReport-Target.xml** が、バッチファイルにより生成される最初の出力で、**SecondXML.xml** が2番目の生成される出力です。

```
@echo off

AltovaXML /xslt2 "MappingMapToExpReport-Target.xslt" /in
"C:/Altova/MapForce2012/MapForceExamples/Tutorial/mf-ExpReport.xml" /out
"C:/Altova/MapForce2012/MapForceExamples/Tutorial/ExpReport-Target.xml" %*
IF ERRORLEVEL 1 EXIT/B %ERRORLEVEL%
AltovaXML /xslt2 "MappingMapToExpReport-Target2.xslt" /in
"C:/Altova/MapForce2012/MapForceExamples/Tutorial/mf-ExpReport.xml" /out
"C:/Altova/MapForce2012/MapForceExamples/Tutorial/SecondXML.xml" %*
IF ERRORLEVEL 1 EXIT/B %ERRORLEVEL%
```

処理シーケンスのマッピングの変更：

1. 左のターゲットコンポーネントをクリックして、右の1つ下に移動します。



2. コードを再生成して、**DoTransform.bat** ファイルを確認します。

```
@echo off

AltovaXML /xslt2 "MappingMapToExpReport-Target.xslt" /in
"C:/Altova/MapForce2012/MapForceExamples/Tutorial/mf-ExpReport.xml" /out
"C:/Altova/MapForce2012/MapForceExamples/Tutorial/SecondXML.xml" %*
IF ERRORLEVEL 1 EXIT/B %ERRORLEVEL%
AltovaXML /xslt2 "MappingMapToExpReport-Target2.xslt" /in
"C:/Altova/MapForce2012/MapForceExamples/Tutorial/mf-ExpReport.xml" /out
"C:/Altova/MapForce2012/MapForceExamples/Tutorial/ExpReport-Target.xml" %*
IF ERRORLEVEL 1 EXIT/B %ERRORLEVEL%
```

SecondXML.xml は、バッチファイルにより最初に生成される出力で、**ExpReport-Target.xml** は2番目に生成される出力です。

チェーンされたマッピング

上記の同じ処理シーケンスがチェーンされたマッピングのために続きます。チェーンされたマッピンググループは、1つのユニットとしてみなされます。単一のチェーンされたマッピングの中間または、最後の ターゲットコンポーネント位置を変更しても、処理シーケンスに影響を与えません。

複数の「チェーン」または、複数の ターゲットコンポーネントがマッピング内に存在する場合、各グループの**最後の** ターゲットコンポーネントの位置が何か最初に処理されるかを決定します。

- 最後 2つの ターゲットコンポーネントが同じ垂直方向 位置を持つ場合、左端のコンポーネントが優先されます。
- 最後 2つの ターゲットコンポーネントが同じ水平方向 位置を持つ場合、一番高い位置にあるコンポーネントが優先されます。
- コンポーネントが、完全に同じ位置を持つシナリオは少ないですが、定義されている順序を保証する、変更不可能な、ユニークな内部コンポーネントID が自動的に使用される場合はあります。

5.11.4.2 優先コンテキストノード/アイテム

スキーマ内の異なったアイテムに関数を適用する場合、MapForce は、コンテキストノードが何であるか知る必要があります。全ての他のアイテムは、これに対して処理されます。これはアイテム(またはノード)を優先コンテキストと設定することにより達成されます。

関連しないアイテムをマッピングする際、優先コンテキストは、実行の優先順序を決めるために使用されます。

マッピングは常に上から下に実行されます。2つのテーブルをループまたは検索する場合、それぞれのループが連続的に処理されます。関連しない要素をマッピングする場合、優先コンテキストを設定しないとMapForce は、どのループを最初に実行するか知らないため、自動的に最初のテーブルまたはデータソースを選択します。

解決方法：
どのデータソースが最初にループ または、検索されるかを決定し、そのソースデータへのコネクタに優先コンテキストを設定します。

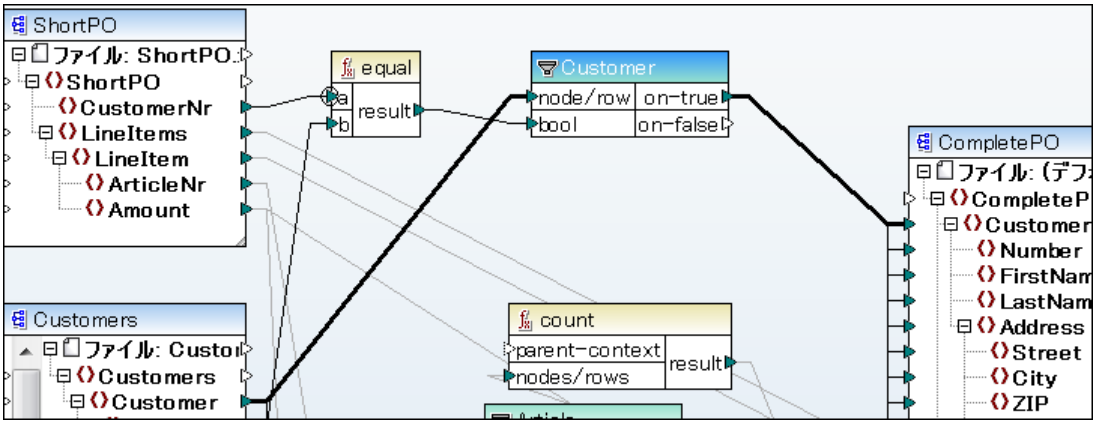
CompletePO.mfd ファイルは **...\MapForceExamples** フォルダー内にあります。

複数のソースコンポーネントがこのサンプルには存在することに注意してください。**ShortPO、Customers、および Articles** は、すべて関連した XML インスタンスファイルを持つスキーマです。それぞれからのデーターは CompletePO スキーマ /XML ファイルにマップされます。優先コンテキストアイコンは、丸枠で囲まれています。

- ShortPO 内の **CustomerNr** は、Customers ファイル内のアイテム **Number** と比較されます。
- **CustomerNr** は **優先コンテキスト**として設定され、equal 関数の **a** パラメーター 内に置かれています。
- **Customers** ファイル内の**同じ**数値が (**1度**)検索されます。**b** パラメーター は、Customers ファイルからの Number アイテムを含んできます。
- 数値が検索された場合、結果は **filter** 関数の **ブール** パラメーターにパスされます。
- **node/ row** パラメーターは、ブールパラメーター が true の場合、すなわち、同じ数値が検索された場合、**Customer** データを "bn- true" にパスします。
- 残りの customer データは以下の通りパスされます :Number, FirstName、LastName アイテムはすべてのターゲットスキーマ内の対応するアイテムに接続されます。

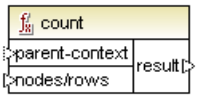
equal 関数の**b** パラメーター(すなわち、アイテムNumber) を **優先コンテキスト**」と設定すると以下の結果が発生します :

- MapForce は、最初の Number を検索し**b** パラメーター にロードします。
- **a** パラメーター内の **CustomerNr** に対してチェックします。
- 等値ではない場合、次の Number を **b** にロードし、また、a に対してチェックします。
- ShortPO 内の数値を検索するために、すべての Number に対して反復します。



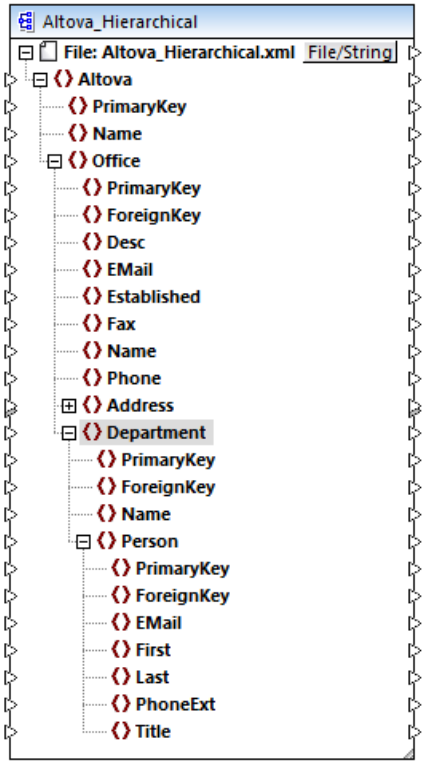
5.11.4.3 マッピングコンテキストのオーバーライド

マッピングの一部では、希望するマッピングの出力を達成するために、マッピングのコンテキストをオーバーライドする必要がある場合があります。このため、コンポーネントの一部では、必要な場合マッピングコンテキストを影響することのできる構造内の任意の`parent-context` アイテムを提供する場合があります。このようなコンポーネントの例は集計関数と変数です。



任意のparent-context を使用した集計関数

マッピングコンテキストの重要性を理解するために、複数のレベルを持つネストされたノードを含むマッピング XML ファイルに追加します。 **挿入** メニューで、**「XML スキーマ/ファイル」**をクリックし、以下のファイルを参照します：**<Documents>\Altova\MapForce2017\MapForceExamples\Altova_Hierarchical.xml**。

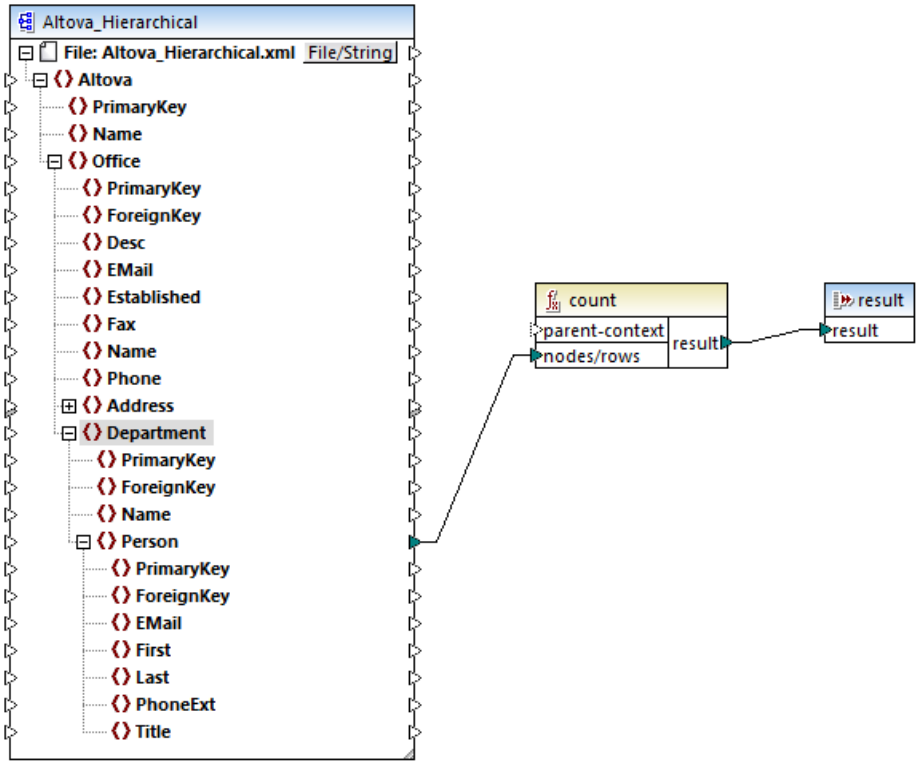


Altova_Hierarchical.xml

重要な点は、上のXML ファイル内では、`Office` 親ノードには複数の`Department` ノードが含まれており、各 `Department` には、複数の`Person` が含まれています。If you open the 実際のXML ファイルをXML エディター内で開くには、オフィスと部署の社員の所属を確認してください：

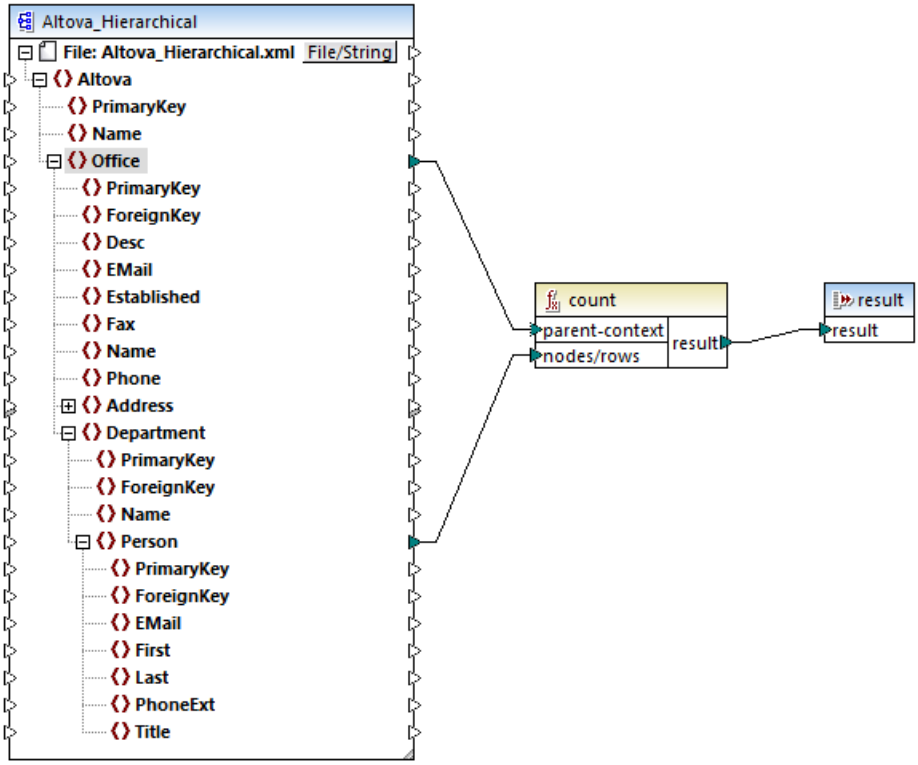
オフィス	部署	人数
Nanonull, Inc.	Administration	3
	Marketing	2
	Engineering	6
	IT & Technical Support	4
Nanonull Partners, Inc.	Administration	2
	Marketing	1
	IT & Technical Support	3

マッピングが全ての部署の全ての社員を含むと想定します。この目的を達成するためには、core | aggregate 関数から **`count`** 関数を追加し、データを以下の様にマップします：



マッピングをこの時点でプレビューすると、出力は、全ての部署内の社員の総数である21 です。**count** 関数には、まだ接続されていない任意のparent-context アイテムが含まれています。この結果、**count** 関数の親コンテキストは、ソースコンポーネントのデフォルトのルートノードです（この場合、Altova アイテム）。これは、全ての部署の全ての社員が **count** 関数の対象と考えられています。マッピングのルールと戦略で説明されているとおり、デフォルトでは、マッピングコンテキストはこのように作動します。マッピングのシナリオの大半はこれだけで十分です。

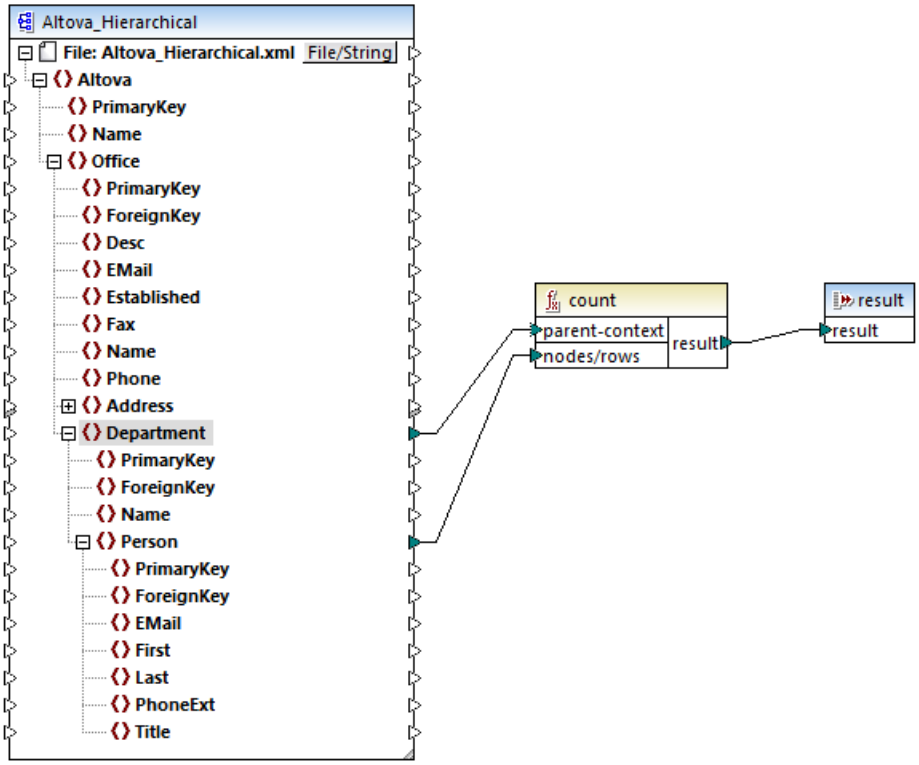
しかしながら、デフォルトの マッピングコンテキストを必要な場合はオーバーライドすることもできます。これを行うには、Department ノードからparent-context アイテムへの接続を下に表示されるように追加します。



上に示されるようにマッピングを変更することにより、マッピングに各オフィスのコンテキスト内で社員のレコードを反復するように命令します。このため、マッピングをプレビューすると、出力は、`15*` に成ります。これは最初のオフィス「Nanonull, Inc.」内の社員数です。社員ノードが（各オフィスのために1度ごと）2回計算されていることがわかります。各オフィスの社員数は、それぞれ15人と6人です。しかしながら、最初の結果のみが返されています（これは、関数が値のシーケンスを返すことができず、簡単な値のみを返すからですbecause 関数）。

* *マッピングのターゲット言語がXSLT 1.0.以上と仮定しています*

下に示されるようにマッピングコンテキストを`Department`, に変更することができます。今度は、社員のレコードが各部署のコンテキスト内で数えられます（すなわち、部署の総数に対応する 7回）。また、最初のオフィスの最初の部署内の社員数に対応する、出力が3である最初の結果のみが返されます。



このマッピングは多くがまだ発生していませんが、この時点では、マッピングの出力を`parent-context` アイテムがどのように影響するかを説明しています。この点を考慮して、変数などを含む他のマッピング内の`parent-context` をオーバーライドすることができます。次も参照してください: サンプル：レコードのグループ化およびサブグループ化.

Chapter 6

データソースとターゲット

6 データソースとターゲット

このセクションは、MapForce がマッピングに必要なソースとターゲットコンポーネントの型について説明します。

- XML とXML スキーマ
- HL7 v3.x からXML スキーマへ

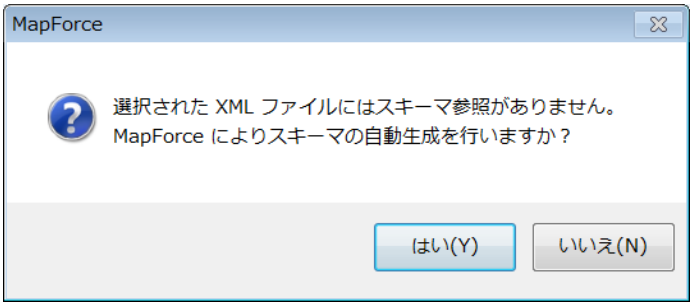
6.1 XML と XML スキーマ

このドキュメントのはじめに、XML と XML スキーマファイルをソースまたはターゲットコンポーネントとして、使用する簡単なマッピングのサンプルについて説明しました。このセクションはマッピング内で XML コンポーネントの使用についての詳細について説明します。以下のトピックが含まれます：

- XML コンポーネント設定
- DTD をスキーマコンポーネントとして使用
- 派生した XML スキーマ型とマッピング
- QName サポート
- Nil 値 /Nillable
- コメントと処理命令
- CData セクション
- ワイルドカード -xs:any

6.1.1 XML スキーマの生成

スキーマがない場合、MapForce は、自動的に XML スキーマを既存のファイルをベースに生成することができます。マッピングエリアにスキーマを持たない XML ファイルを追加すると、次のダイアログボックスが表示されます。

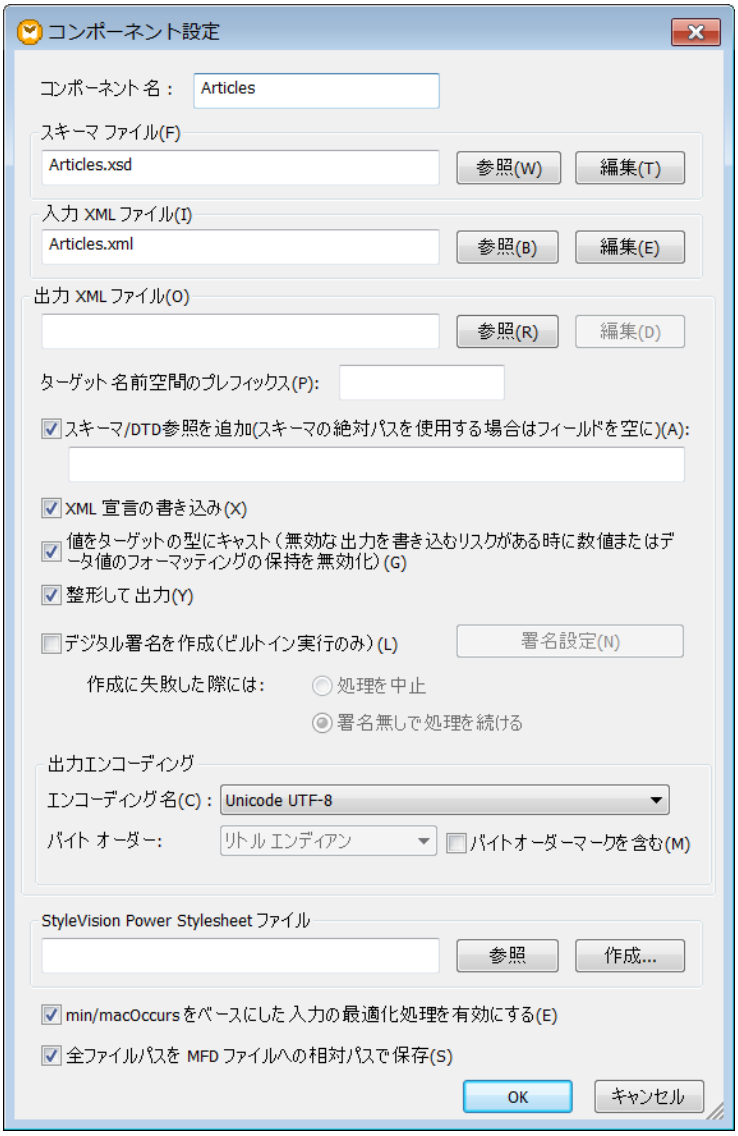


MapForce が XML ファイルからスキーマを生成する場合、要素 属性は、XML インスタンスドキュメントから推定されなければならず、予期しないものの可能性があります。生成されたスキーマがインスタンスデータを正確に表していることを確認することが奨励されます。

6.1.2 XML コンポーネント設定

マッピングエリアに XML コンポーネントを追加した後、コンポーネント設定ダイアログボックスから適用することのできる設定を構成することができます。コンポーネント設定 ダイアログボックスを以下の方法で開くことができます：

- マッピング上のコンポーネントを選択し、 **「コンポーネント」** メニューから **「プロパティ」** をクリックします。
- コンポーネントヘッダーをダブルクリックします。
- コンポーネントヘッダーを右クリックして、 **「プロパティ」** をクリックします。



XML コンポーネント設定 ダイアログボックス

使用することのできる設定は、以下のとおりです .

コンポーネント名	コンポーネント名はコンポーネントを作成する際に自動的に生成されます。ですが、後に名前を変更することが可能です。 コンポーネント名が自動的に生成され、インスタンスファイルをその後に選択した場合、コンポーネント名もオプションで更新するようにMapForce は、プロンプトします。 コンポーネント名 はスペース(例 : "ソースXML ファイル") またフルストップ文字も含むことができます (例 : "Orders.EDI")。コンポーネント名は、スラッシュ、バックスラッシュ、コロン、二重引用符、行頭および末尾スペースを含んではなりません。全般的に、コンポーネントの名前を変更するさいの以下の影響を考慮してください :

	• マッピングを FlowForce Server にデプロイする場合、コンポーネント名は一意である必要があります。 • コマンドラインに入力できる文字のみを使用することが奨励されます。コマンドラインと Windows 内では、それぞれの文字で異なったエンコードを使用する場合があります。
スキーマファイル	MapForce によりデータを検証とマップするために使用された XML スキーマファイルの名前とパスを指定します。 スキーマファイルを変更するには、 **参照]** をクリックして、新しいファイルを選択します。XMLSpy 内のファイルを編集するには、 **編集]** をクリックします。
入力 XML ファイル	MapForce がデータを読み込む XML インスタンスファイルを指定します。このフィールドはソースコンポーネントにとり重要で、最初にコンポーネントを作成した際に入力され、XML インスタンスファイルに割り当てられます。 ソースコンポーネント内で、インスタンスファイル名は XML ルート要素とレファレンススキーマを検出するため、また選択されたスキーマに対して検証するために使用されます。 ファイルのロケーションを変更するには、 **参照]** 新しいファイルを選択します。XMLSpy 内のファイルを編集するには、 **編集]** をクリックします。
出力 XML ファイル	MapForce がデータを書き込む XML インスタンスファイルを指定します。このフィールドはターゲットコンポーネントにとり重要です。 ファイルのロケーションを変更するには、 **参照]** をクリックして、新しいファイルを選択します。XMLSpy 内のファイルを編集するには、 **編集]** をクリックします。
ターゲット名前空間のためのプレフィックス	ターゲット名前空間のためにプレフィックスを入力することを許可します。プレフィックスを割り当てる前に、ターゲットスキーマ内でターゲット名前空間が定義されている必要があります。
スキーマ/DTD レファレンスの追加	参照された XML スキーマファイルを XML 出力のルート要素に対してパスを追加します。このフィールドに入力されたスキーマのパスは、`xsi:schemaLocation` 属性または DTD が使用される場合、`DOCTYPE` 宣言 内の生成されたターゲットインスタンスファイルに書き込まれます。 このフィールドにパスを入力することにより、XML インスタンスファイルにより参照されたスキーマファイルがどの箇所に位置するかを決定することができます。これにより、マッピングが実行される際、出力インスタンスがマッピングのデスティネーションで検証されることを保証できます。また、**http://** アドレスや絶対及び相対パスをこのフィールドに入力することもできます。 このオプションを無効化することにより、XML インスタンスを参照された XML スキーマまたは DTD から、切り離すことができます (例：結果する XML 出力を基となる XML スキーマを持たない相手に送信する場合など)。
XML 宣言の書き込み	このオプションにより生成された出力から XML 宣言 を表示しないようにすることができます。デフォルトでは、オプションが有効化されていると、XML 宣言 が出力に表示されます。 この機能は MapForce ターゲット言語と実行エンジンでサポートされています。

	<table><tr><th>ターゲット言語 /実行エンジン</th><th>出力がファイルの場合</th><th>出力が文字列の場合</th></tr><tr><td>XSLT, XQuery</td><td>はい</td><td>いいえ</td></tr></table>
ターゲット型に値をキャストする	Aターゲット XML スキー型がマッピングで使用されるかを定義します。またはターゲットコンポーネントにマップされる全てのデータが**文字列**値として扱われるかを定義します。デフォルトでは、この設定 が有効化されています。 このオプションを無効化することにより、正確なフォーマットの値を保持することができます。例 : これは、数値内で特定の小数点を必要とするスキーマ内のパターンファセットを満足させるためにの場合役にたちます。 マッピング関数を使用して、数値を文字列として必要とされるフォーマット内で使用し、この文字列をターゲットにマップすることができます。 このオプションを無効化にすると、無効な値の検出も無効化されます。例 数値フィールドに文字が入力された場合。
整形出力	出力 XML ドキュメントに構造化された外見を与えるために整形出力します。それぞれの子 ノードは、単一タブ文字のそれぞれの親のオフセットです。
出力エンコード	出力インスタンスファイルの次の設定を指定することを許可します : • エンコード名 • バイトオーダー • バイトオーダーマーク (BOM) 文字が含まれるか否か。 デフォルトでは、**新しいコンポーネントのためのデフォルトのエンコード**」オプション内で定義されたエンコードを持つ新しいコンポーネント。**「ツール	オプション」**からこのオプションにアクセスし、タブを生成します。 マッピングがXSLT 1.0/2.0 を生成すると、**バイトオーダーマーク**チェックボックスを有効化しても、これらの言語はバイトオーダーマークをサポートしないため効果はありません。
StyleVision Power スタイルシートファイル	このオプションにより、Altova StyleVision スタイルシートファイルを選択または作成することができます。このようなファイルは、XML インスタンスファイルからのデータをHTML、RTF などの多種のレポートに適したフォーマットで出力することができます。 以下も参照 : コンポーネント上で相対パスを使用する
min/maxOccurs をベースに入力処理の最適化の有効化	このオプションにより、`minOccurs` と`maxOccurs="1"` を持つ必要な属性または子要素 などの1つのアイテムのみを含むシーケンスを特別に処理することが許可されます。この場合、シーケンスの最初のアイテムが抽出され、アイテムは (シーケンスとしてではなく)直接動的な値として処理されます 入力データがスキーマに対して場合、**有効でない場合、**エラーメッセージと共にマッピングを停止する空のシーケンスがマッピング内で生じる可能性があります。このような**無効な入力**を処理するには、このチェックボックスを無効化にします。
MFD ファイルに相対的なすべてのファイルパスを保存する	このオプションが有効化されると、MapForce は、コンポーネント設定ダイアログボックスに表示された、MapForce Design (.mfd) ファイルの場所に

	相対したファイルパスを保存します。以下も参照：コンポーネント上で相対パスを使用する。

6.1.3 "スキーマ" コンポーネントとしてを DTD 使用する

MapForce 2006 SP2 以降では、ソースならびにターゲットコンポーネントにて名前空間を意識したDTD がサポートされます。名前空間 URI がDTD の "xslns" 属性宣言から抽出され、マッピングを行うことが可能になります。

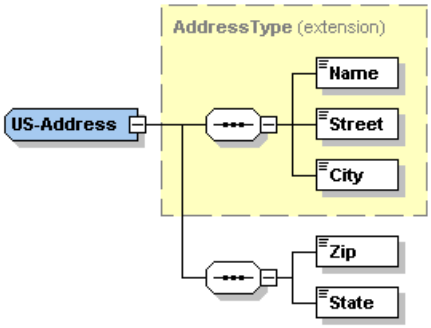
しかしStyleVision により使用される DTD の様に、名前空間 URI を持たずに xmlns* 属性を含む DTD も存在します。このような DTD は、MapForce で使用することができるように拡張する必要があります。以下に示されるように、DTD を修正して名前空間 URI を含む xmlns 属性を定義する必要があります:

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

6.1.4 派生した XML スキーマの型

MapForce では複合型の派生型に対するマッピングがサポートされます。派生型は、**xsi:type** 属性により特定の派生型を指定するXML スキーマの複合型です。

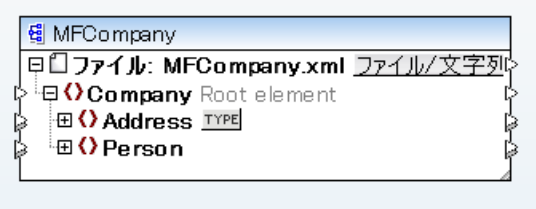
以下のスクリーンショットは、XMLSpy にて派生型の "US-Address" の定義を示したものです。基底型（またはオリジナルとなっている複合型）はこの場合 AddressType となります。`Zip` と`State` の2 つの要素を新たに追加することで、派生型のUS-Address が作成されます。



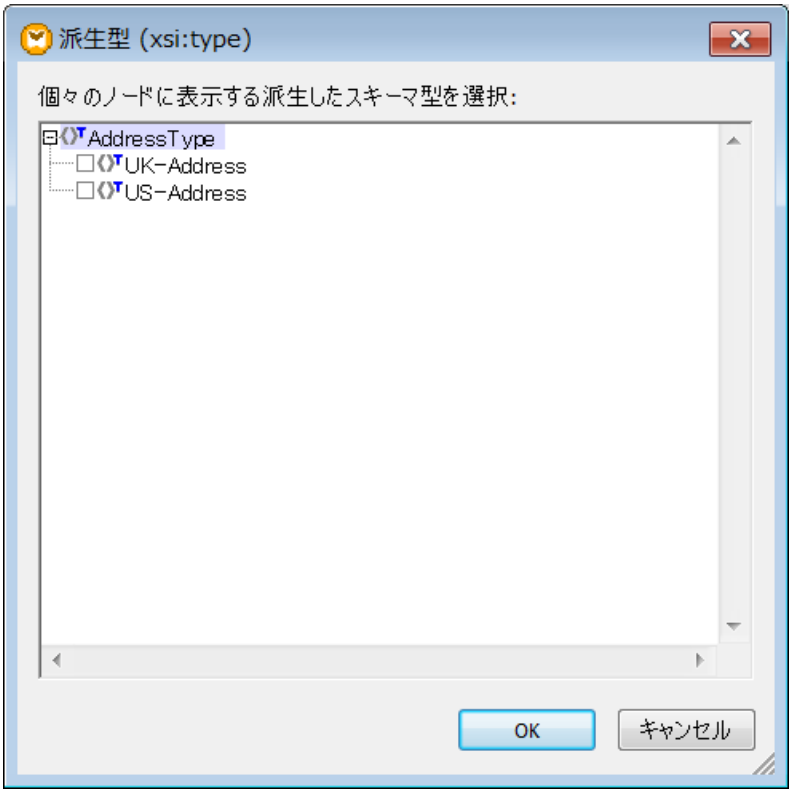
生成された型のサンプル（XMLSpy スキーマビュー）

次のサンプルは、派生したXML スキーマ型からデータをマップ、または、スキーマ型へのデータのマップの方法を説明しています。

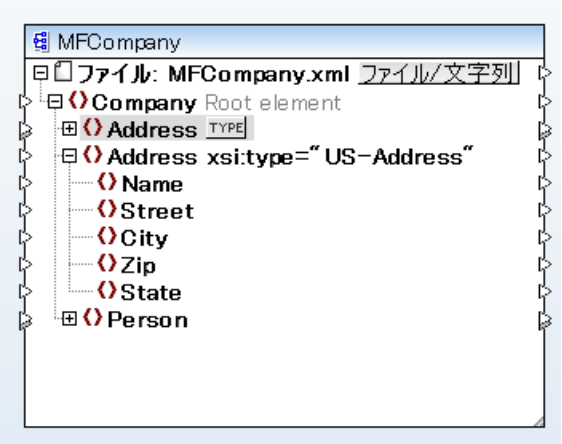
1. **「挿入」** メニューから **「XML スキーマ/ファイル」** をクリックします。以下のXML スキーマ:**<Documents>\Altova\MapForce2017\MapForceExamples\Tutorial\MFCompany.xsd** を開きます
2. インスタンスファイルを提供するようにプロンプトされると、**「スキップ」** をクリックして、ルート要素をとして`Company` 選択します。



3. `Address` 要素の横の TYPE ボタンをクリックします。このボタンは、スキーマ内のこの要素のために存在する派生した型であることを示しています。



4. 使用する派生した型の横のチェックボックスを選択し、(この場合、`US-Address`)、「OK」を押して確認します。新規の要素 `Address xsi:type="US-Address"` がコンポーネントに追加されます。



これらのアイテムに対して直接、または、US-Address から、マッピングを行うことが可能になります。

派生型ダイアログボックスにて複数のアイテムを選択することで、複数の派生型を追加/挿入することができます。各ノードにはそれぞれ独自のxsi:type 要素が与えられます。

6.1.5 QNames

MapForce は、マッピングの実行ランタイムでXML ファイルからデータを読み込む際にQName (修飾名) プレフィックス (http://www.w3.org/TR/xml-names/#ns-qualnames) を解決します。

QNames は、XML インスタンスドキュメント内の名前空間 URI を参照および省略するために使用されます。QNames には、2つの種類があります :プレフィックスを持つ、またはプレフィックスを持たない QNames.

PrefixedName	Prefix ':'	LocalPart
UnPrefixedName		LocalPart LocalPart が要素 または属性の名前の箇所

例 :下のリストでは、<x:p/> がQName です :

- プレフィックス 'x' は名前空間 "**http://myCompany.com**" の省略形です。
- p は、ローカルパートです。

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

MapForce は、**xpath2 | qname-related 関数** ライブラリにQName に関連した関数の一部を含みます。

6.1.6 Nil の値 / Nillable

XML スキーマ仕様に従えば、nillable="true" 属性をスキーマ内の要素に対して指定することで、(コンテンツのデータ型により空のコンテンツが許されていない場合でも) コンテンツを含まない要素でも妥当とみなすことができるようになります。XML インスタンスドキュメント内に記述される属性の xsi:nil="true" により、要素は存在するが、コンテンツは含まれない (つまり空要素である) ことが示されます。このセクションはどのように、MapForce がソースとターゲットコンポーネント内の nil 要素を扱うかを説明します。

'xsi:nil' と 'nillable' の比較

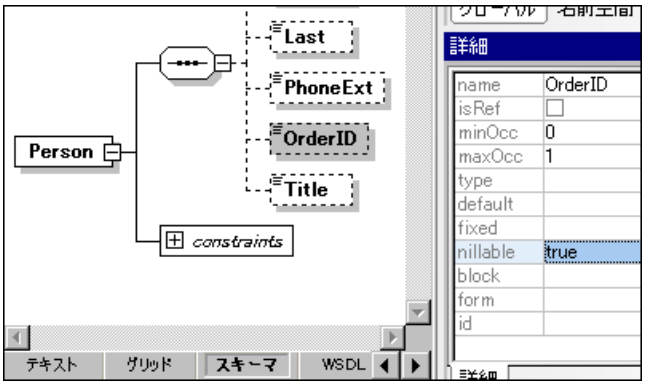
xsi:nil="true" 属性はXML **インスタンス**ファイルにおいてのみ定義されます。

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

xsi:nil 属性は MapForce のマッピングにて視覚的（かつ明示的）に示されることは無く、自動的に処理されます。通常、null 値が許容されたノード（xsi:nil="true" 属性が含まれるノード）は存在するが、コンテンツには何も含まれない状態になります。

`xsi:nil` 属性は、通常自動的に処理されるため、MapForce グラフィカルなマッピングで明示的に表示されません。特に "nilled" ノード（`xsi:nil="true"` 属性を持つ）が存在する場合、そのコンテンツは存在しません。

`nillable="true"` 属性は、XML **スキーマ**内で定義されます。MapForce では、ソースとターゲットコンポーネント内に存在することができます。



マッピングソースにある null 値が許容された要素

マッピングにて null 値を許容する（nillable な）データを XML 要素から読み取った場合、xsi:nil 属性が自動的にチェックされます。xsi:nil の値が true となっている場合、コンテンツは存在しないものとして扱われます。

nillable なソース要素から**単純コンテンツ**の（属性は含まれるが子要素を含まない）nillable なターゲット要素に対して、**ターゲット優先**マッピングでマッピングを作成し、xsi:nil がソース要素にセットされている場合、xsi:nil 属性がターゲット要素にも挿入されます（例：`<OrderID xsi:nil="true"/>`）。

全てコピーマッピングを nillable なソース要素から nillable なターゲット要素へマッピングして、xsi:nil がソース要素にてセットされている場合、xsi:nil 属性がターゲット要素にも挿入されます（例：`<OrderID xsi:nil="true"/>`）。

ソース要素の xsi:nil に true がセットされているかを明示的にチェックするには、**`is-xsi-nil`** 関数を使用します。「nil 化」された要素に対して true が返され、それ以外のノードに対して false が返されます。

nil 化された（存在しない）ソース要素の値を別の値で代用するには、**`substitute-missing`** 関数を使用してください。

メモ：

- コンテンツが無くても要素ノードが実際存在するため、**`exists`** 関数を nill 化されたソース要素に接続すると、TRUE を返します。
- `xsi:nil` が設定されている箇所で（**`multiply`** と **`concat`** などの）単純型の値を期待する関数を要素に対して使用すると、要素コンテンツが存在しなく、結果を取得することはできません。これらの関数はソースノードが存在しないかのように振舞います。

マッピングターゲットとしての Nillable 要素

nillable なソース要素から**単純コンテンツ**の（属性は含まれるが子要素を含まない）nillable **なターゲット要素に対して、ターゲット優先**マッピングでマッピングを作成し、xsi:nil がソース要素にセットされている場合、xsi:nil 属性がターゲット要素にも挿入されます（例：`<OrderID xsi:nil="true"/>`）。xsi:nil="true" 属性が XML ソース要素にセット

されていない場合、通常の方法で要素コンテンツがターゲット要素へマッピングされます。

マッピングが（子要素を持つ）**複合型**のnillable 要素に対して行われている場合、xsi:nil 属性が自動的に書き込まれることはありません。これはマッピングが行われた後で子要素が書き込まれるか分からない為です。**全てコピー**接続を定義することで、ソース要素からxsi:nil 属性をコピーすることができます。

空のシーケンスまたはデータベースのNULL 値をターゲット要素へマッピングする場合、それがnillable であるかどうかにかかわらず、要素が作成されることはありません。

空のターゲット要素をxsi:nil="true" 属性がセットされた状態で（強制的に）作成するには、`set-xsi-nil` 関数をターゲット要素へ直接接続してください。この方法は単純型ならびに複合型のターゲット要素に対して使用することができます。

`substitute-missing-with-xsi-nil` 関数を使用することで、マッピングソースから値を得ることができない場合にxsi:nil をターゲットへ挿入することができます。これはソースノードが全く存在しない場合、または（multiply のような）計算に対してnil 化されたソースノードが与えられ、結果が得られなかった場合などに使用することができます。

この関数はデータベースのNULL 値からxsi:nil="true" を持つ要素を作成するのに使用することができます。この関数は単純型のコンテンツを持つノードに対してのみ使用することができます。

メモ:

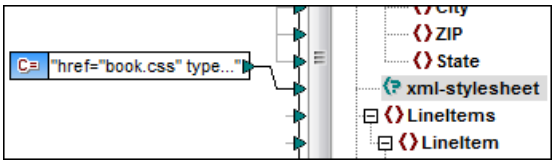
- `xsi:nil` を生成する関数は、（`if-else` 関数などの）値に対して動作する関数またはコンポーネントにパススルーされることはできません。

6.1.7 コメントと処理命令

コメントと処理命令はターゲットXML コンポーネントに挿入することができます。処理命令は、XML ドキュメントを更に処理するアプリケーションに情報をパスするために使用されます。コメントと処理命令は、全てコピーマップグループのためのノードのためには定義することができないことにご注意してください。

処理命令の挿入方法：

1. ターゲットコンポーネント内の要素を右クリックして、コメント/処理命令を選択し、メニュー（前、後）から処理命令オプションを選択します。
2. 処理命令（ターゲット）名をダイアログ内に入力し、「OK」を押して確認します。例：xml-stylesheet。これによりこの名前をコンポーネントツリーに追加します。

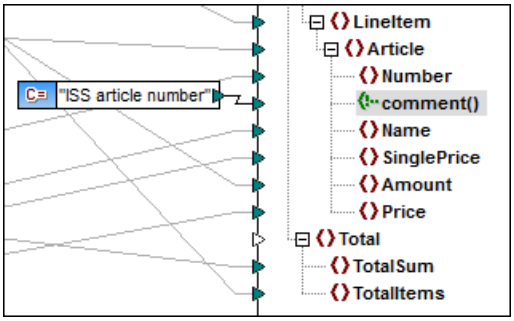


3. 以下を使用することができます。例：処理命令属性の値を与えるために定数コンポーネントを使用する。例：`href="book.css" type="text/css".`

メモ：
複数の処理命令は、ターゲットコンポーネント内の要素の前または後ろに追加することができます。

コメントの挿入：

1. ターゲットコンポーネント内の要素を右クリックして、コメント処理命令を選択し、メニューからコメントオプション(前、後ろ)を選択します。



これは、コンポーネントツリーにコメントノード(<!--comment()>)を追加します。

2. 定数コンポーネントを使用して、コメントコンテキストを与えます、または ソースノードをコメントノードに接続します。

メモ:
単一のターゲットノードには1つのコメントのみを追加することができます。複数のコメントを作成するには、入力関数の複製を使用します。

コメント処理命令の削除：

- 対応するノードを右クリックして、コメント処理命令を選択し、フライアウトメニューから「削除」コメント処理命令を選択します。

6.1.8 CDATA セクション

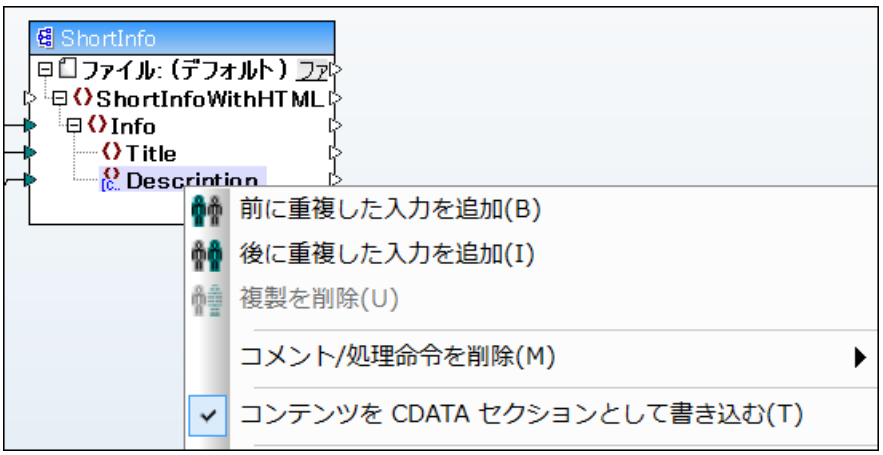
CDATA セクションは、通常マークアップとして解釈される文字を含むテキストのブロックをエスケープするために使用されます。CDATA セクションは "<![CDATA[" で開始し、"]]>" で終わります。

ターゲットノードは、CDATA セクションとして受信される入力データを書き込むことができます。ターゲットノードコンポーネントは以下であることができます：

- XML データ
- データベースフィールド内に埋め込まれたXML データ
- XBRL ターゲット内の型指定されたディメンションのXML 子要素

CDATA セクションの作成：

1. CDATA セクションとして定義するターゲットノードを右クリックして、"CDATA セクションとしてコンテンツを書き込む"を選択します。



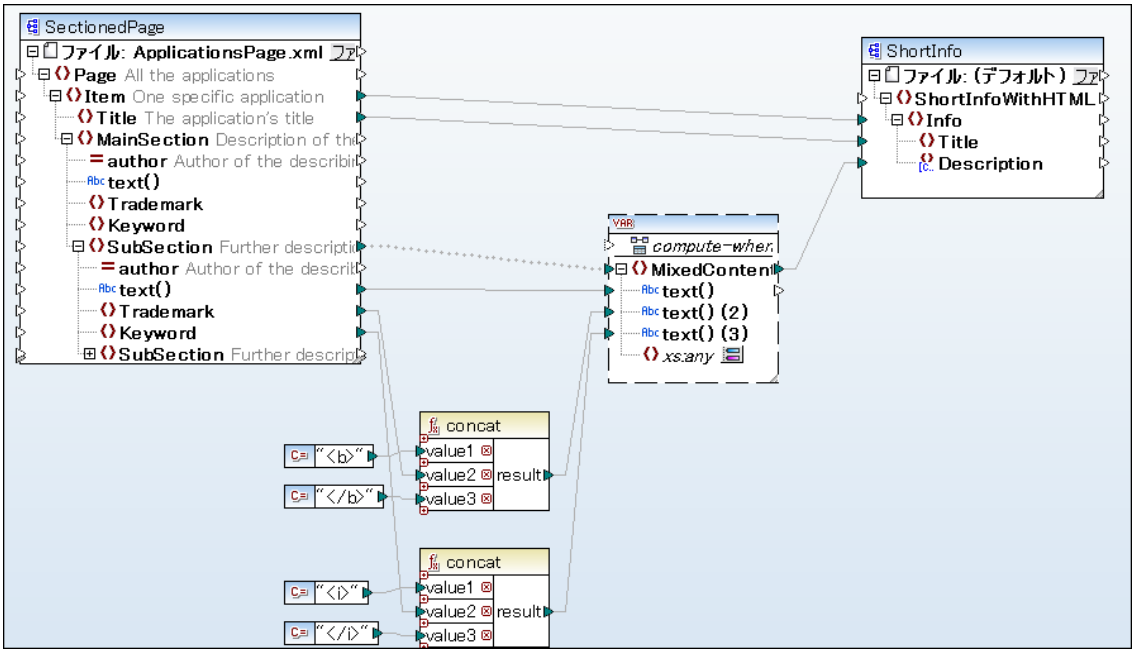
入力データは、区切り記号]]> で終了されたCDATA セクションを含まない用に警告プロンプトを表示します。「**OK**」をクリックしてプロンプトを閉じます。
下に表示される[C.. アイコンでは、要素タグがこのノードがCDATA セクションとして定義されていることを示しています。

メモ:
CDATA セクションは、複製されたノード、および、xsi:type ノード上でも定義されることができます。

例:
...\MapForceExamples フォルダー内で使用することのできる**HTMLinCDATA.mfd** マッピングファイルは、CDATA セクションが有効利用できる例を表示しています。

この例:

- 太字の開始 () と終了 () タグは**Trademark** ソース要素のコンテンツに追加することができます。
- イタリックの開始 () と終了 () タグは**Keyword** ソース要素のコンテンツに追加することができます。
- 結果のデータは、サブセクション要素コネクタがソース優先 (複合型コンテンツ) ノードとして定義されているため、ソースドキュメント内に表示されるために複製 **text()** ノードにパスされます。
- 複合型ノードの出力は、CDATA セクションとして定義されているShortInfo ターゲット コンポーネント内の**詳細** ノードにパスされます。

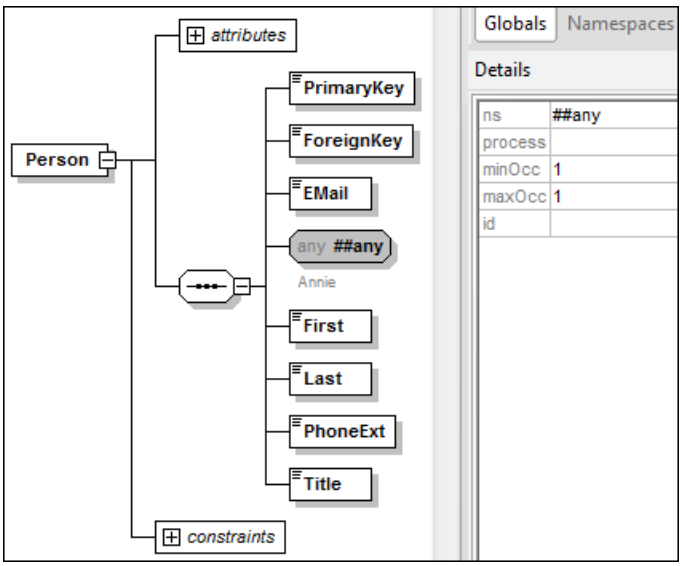


出力」ボタンをクリックすることにより、マークアップされたテキストを含むCDATA セクションが表示されます。

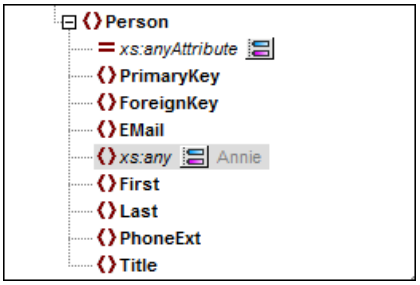
```
7     <Info>
8       <Title>MapForce</Title>
9       <Description><![CDATA[Altova <b>MapForce</b> 2014 Enterprise Edition is the premier <i>XML</i>
/ <i>database</i> / <i>flat file</i> / <i>EDI</i> data mapping tool that auto-generates mapping code in
<i>XSLT</i> 1.0/2.0, <i>XQuery</i>, <i>Java</i>,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

6.1.9 ワイルドカード - xs:any / xs:anyAttribute

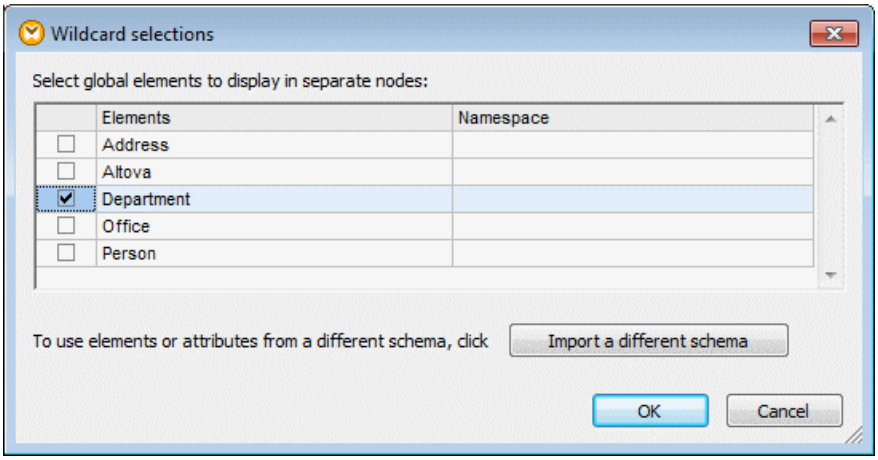
ワイルドカードxs:any (およびxs:anyAttribute) は、すべての要素/属性をスキーマから使用することを許可します。スクリーンショットはXMLSpy のスキーマビュー内の "any" 要素を表示しています。



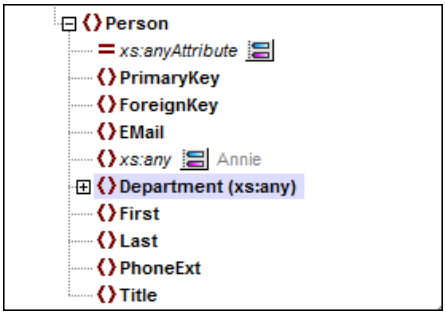
MapForce 内では、選択の変更 (excel1-compicon) ボタンがxs:any (またはxs:anyAttribute)要素の横に表示されます。



クリックすると、選択の変更 ボタンexcel1-compicon が開かれ、"ワイルドカードの選択 "ダイアログボックスが開かれます。エントリは、現在のスキーマ内で宣言されているグローバルな要素と属性を表示します。



クリックすると、選択の変更 ボタンexcel1-compicon が開かれ、"ワイルドカードの選択 "ダイアログボックスが開かれます。エントリは、現在のスキーマ内で宣言されているグローバルな要素と属性を表示します。



. これらのノードから、または、ノードへ、他の要素と同様にマップすることができます。

コンポーネント上では、ワイルドカード要素、または、属性は、名前に追加されている**(xs:any)** テキストとして識別されることができます。

ワイルドカード要素を削除するには、選択の変更 (excel1-compicon) ボタンをクリックして、"ワイルドカードの選択 "ダイアログボックスから選択の解除を行います。

ワイルドカードと動的なノード名

ワイルドカードから、または、ワイルドカードへのマッピングは、通常、使用することのできる要素または属性がインスタンス内に表示されており、XML インスタンスがコンポーネントの XML スキーマにより宣言されている場合（または、外部のスキーマからインポートされている場合）に適しています。しかしながら、インスタンス内に表示される要素または属性が多すぎ、スキーマ内で宣言されるには多すぎる場合があります。<message> の子要素の数が任意である次のインスタンスの場合を考慮してみてください：

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

上記のようなシチュエーションでは、ノード名への動的なアクセスがワイルドカードの使用よりも適しています（ノード名のマッピング を参照）。

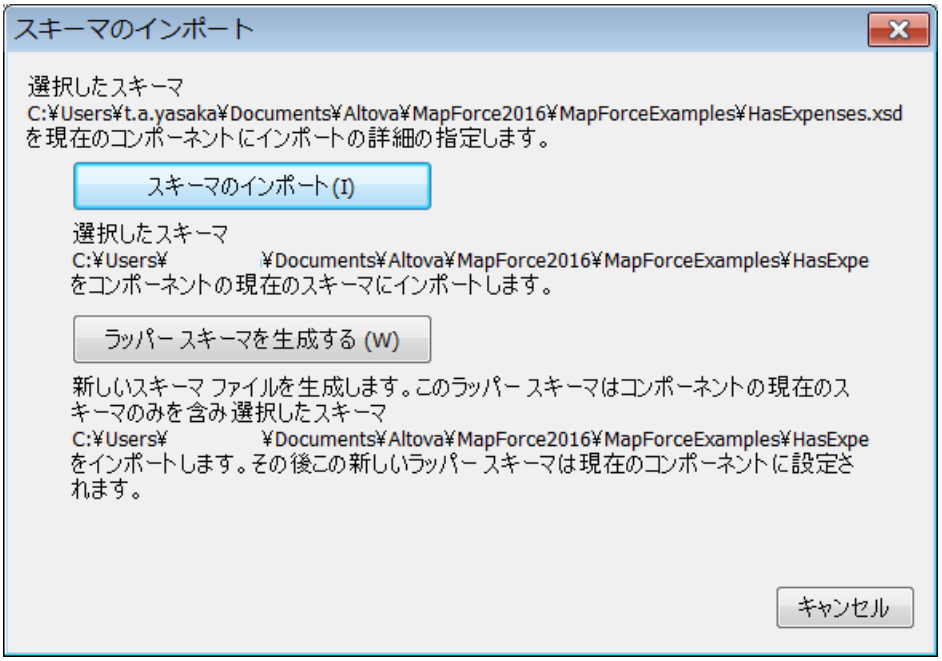
異なるスキーマからワイルドカードとして要素を追加する

コンポーネントに割り当てられた要素以外のスキーマからの要素は、ワイルドカードとして使用することができます。このような要素をコンポーネント上で表示するには、"ワイルドカードの選択 "ダイアログボックス上の**異なるスキーマのインポート**ボタンをクリックします。これにより以下の2つのオプションを持つ新規ダイアログボックスが開かれます：

1. スキーマのインポート

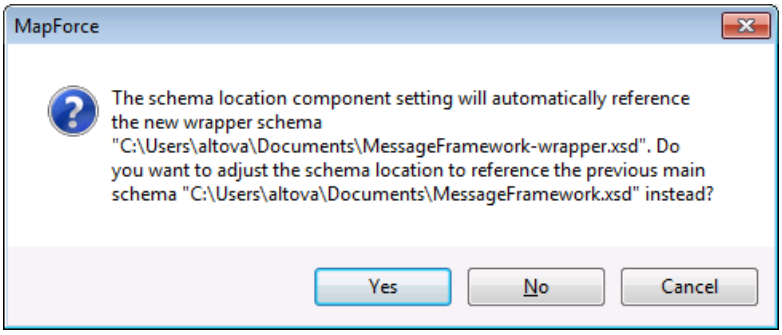
2. ラッパースキーマの生成

例えば、下のイメージは、**HasExpenses.xsd** という外部スキーマを現在のスキーマにインポートしようと試みると何が起こるかを表しています。



スキーマのインポート オプションは外部スキーマをコンポーネントに割り当てられている現在のスキーマにインポートします。このオプションはディスク上のコンポーネントの既存のスキーマをオーバーライドすることに注意してください。現在のスキーマがディスクではなくURL により開かれたリモートのスキーマである場合、変更することはできません (URL からコンポーネントを追加するを参照してください)。この場合、ラッパースキーマの生成オプションを使用してください。

ラッパースキーマの生成オプションは "ラッパー" スキーマという名前の新しいスキーマファイルを作成します。このオプションの利点は、コンテンツの既存のスキーマが変更されない点です。代わりに、既存のスキーマとインポートされるスキーマの両方を含む(ラッパースキーマという)新しいスキーマが作成されます。このオプションを選択すると、ラッパースキーマを保存する場所を問われます。デフォルトでは、ラッパースキーマは somefile-wrapper.xsd. という書式の名前を持ちます。ラッパースキーマを保存した後、デフォルトでコンポーネントに割り当てられ、ダイアログボックスがプロンプトします :



【はい】 をクリックして、前のスキーマに戻します。それ以外の場合は、**【いいえ】** をクリックして新規に作成された、コンポーネントに割り当てられたラッパースキーマを保持するためにいいえをクリックします。

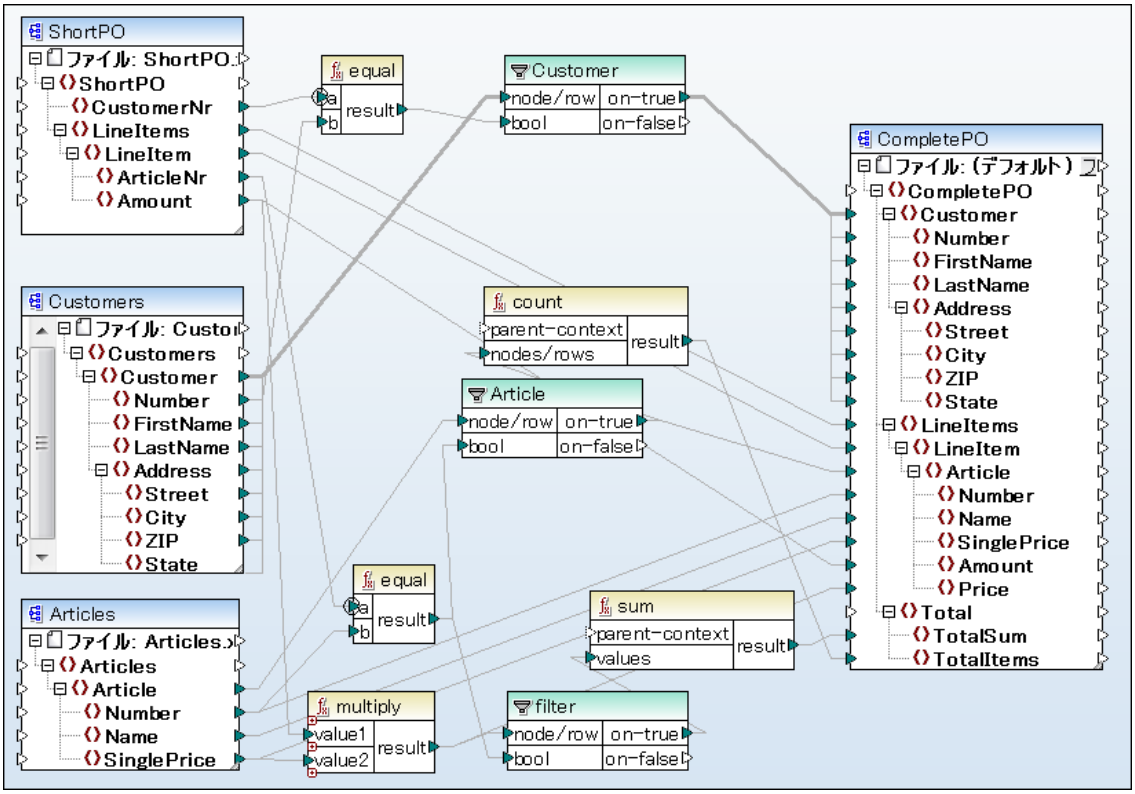
6.1.10 複数のスキーマからのデータのマージ

MapForce では複数のファイルを 1つのターゲットファイルへマージすることができます。

以下にあるサンプルでは、複数のソースコンポーネント (複数のスキーマ) から得られたデータをターゲットスキーマにマージします。同一のスキーマをベースにした複数のファイルをマージする方法については、[複数の入力または出力ファイルを動](#)

的に処理を参照ください。

...\MapForceExamples フォルダーにある **CompletePO.mfd** ファイルでは、 3つのファイルを 1つの XML ファイルへマージする方法を確認することができます



複数のソースコンポーネントデータが 1つのターゲットXML ファイル (CompletePO) へ組み合わされているという点に注目してください。

- **ShortPO** はXML *インスタンスファイル*に関連付けられた*スキーマ*で、顧客番号と品物に関するデータ (LineItem ならびにAmount) だけが収められています。(このファイルには顧客番号が 3の顧客一人だけが収められています。)
- **Customers** はXML *インスタンスファイル*に関連付けられた*スキーマ*で、顧客番号と、Name やAddress といった顧客に関する詳細情報が収められています。
- **Articles** はXML *インスタンスファイル*が関連付けられた*スキーマ*で、品物名やその数、単価といった品物に関する情報が収められています。
- **CompletePO** は、 3つのXML *インスタンスファイル*から得られた*データ*を収めるための、*インスタンスファイル*を持たない*スキーマファイル*となります。このファイルの階層構造により、XML *データ*のマージと出力を行うことが可能になります。

このスキーマファイルはXMLSpy のようなXML エディターにより作成されており、MapForce から生成されたものではありません (CompletePO.xml *インスタンスファイル*が存在する場合、*スキーマファイル*の生成を行うことも可能です)。

CompletePO の構造はソースとなるXML ファイルの構造を組み合わせたものとなります。

フィルターコンポーネント (Customer) を使用することで、ShortPO とCustomer XML ファイルで顧客番号が一致する*データ*を検索することができ、関連する*データ*をターゲットのCompletePO コンポーネントへ渡すことができます。

- "equal" 関数により、ShortPO の **CustomerNr** が Customer 以下にある **Number** と比較されます。
- ShortPO には 1つの顧客情報 番号が 3) しか含まれていないため、顧客番号が 3となって

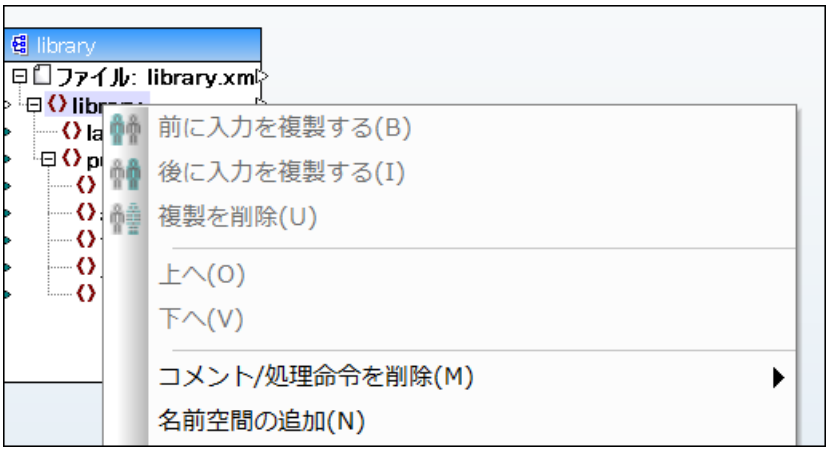
いる顧客の情報だけがフィルターコンポーネントには渡されます。
- フィルターコンポーネントの **node/ row** パラメーターは、bool パラメーターが true の時に **Customer** データを "on- true" ノードへ渡します。つまり、顧客番号が 3の時だけ出力が行われます。
- 品物データは、他にある 2つのフィルターコンポーネントによりターゲットスキーマへ渡されます。

6.1.11 カスタム名前空間の宣言

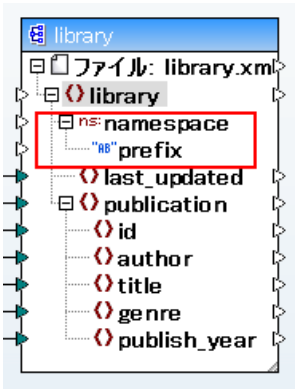
デフォルトでは、マッピングがXML 出力を生成する際に、各要素と属性の名前空間（または、名前空間のセット）が自動的にMapForce によりターゲットコンポーネントと関連するスキーマから生成されます。これは、MapForce 内のデフォルトの振る舞いで、XML 出力の生成を含むマッピングのシナリオの大半に適切です。

しかしながら、結果のXML 出力内で要素の名前空間の管理を更に行う場合、マッピングから要素の名前空間を手動で宣言することが必要な可能性があります。

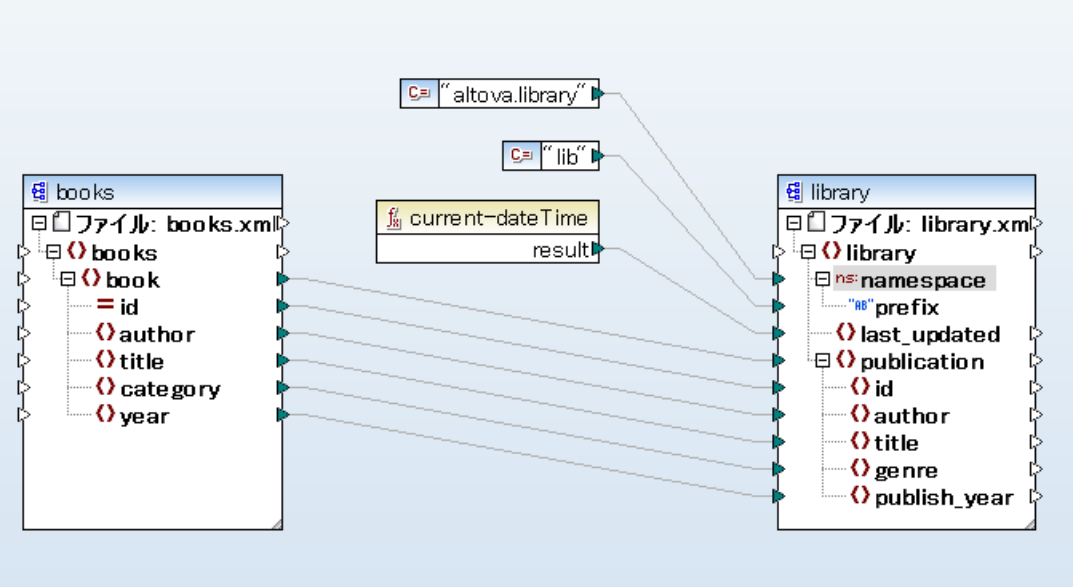
この仕組みを理解するには、 **<Documents>\Altova\MapForce2017\MapForceExamples\Tutorial** 内にある**BooksToLibrary.mfd** マッピングを開きます。`library` ノードを右クリックし、コンテキストメニューから **名前空間の追加**」を選択します。



`library` ノードの下で次の2つの新しいノードが使用することができるようになっていることに注意してください：`namespace` と`prefix`.



マッピングからの文字列の値をマップすることができます。下のイメージでは、名前空間「altova.library」とprefix「lib」を与える、2つの定数が（ **挿入 | 定数**」メニューコマンドを使用して定義されています：



生成された出力内の結果としては、`xmlns:<prefix>="<namespace>"` 属性が、`<prefix>` と `<namespace>` がマッピング (この場合は定数) から来る値である箇所の要素に追加されます。生成された出力は、以下のようになります (ハイライトされている部分に注意してください):

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

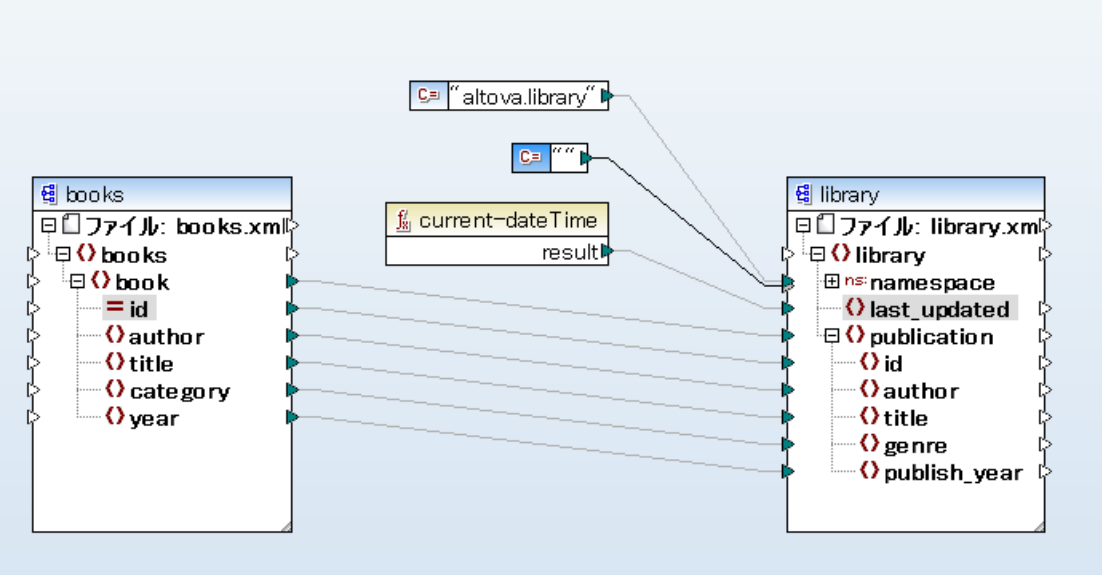
メモ: カスタム名前空間の宣言 (と **名前空間の追加」**コマンド) は、ターゲット XML コンポーネントに取り有益であり、要素のみに適用することができます。属性に対して、 **名前空間の追加」**コマンドは、属性とワイルドカードノードに対しては使用することができません。ノードの子への動的なアクセスが有効化されている場合でも、または、 **全てをコピー」**接続を使用してノードがデータを受け取るノードの場合でも使用することはできません (ノード名のマッピングを参照してください)。

必要に応じて、複数の名前空間を同じ要素のために宣言することができます。これを行うには、ノードをもう一度右クリックし、コンテキストメニューから **名前空間の追加」**を選択します。新規のプレフィックスと名前空間の値に接続することのできる名前空間とプレフィックスノードの新規のペアを使用できるようになります。

以前に追加された名前空間の宣言を削除するには、`ns:namespace` ノードを右クリックし、コンテキストメニューから **名前空間の削除」**を選択します。

空の値が与えられる場合でも、`namespace` と`prefix` 入力コネクタの両方がマップされている必要があります。

(`xmlns="mydefaultnamespace"` のフォーマットの) デフォルトの名前空間を宣言する場合、`prefix` に対して空の文字列をマップします。このケースをマップ上で確認する場合は、2番目の定数が空の文字列になるように上のサンプルマッピングを編集します。



結果の出力は以下のようになります：

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

属性の名前のためにプレフィックスを作成する場合、例えば、`<number prod:id="prod557">557</number>`、ノードの属性への動的なアクセスを有効化する、または`<number>` のために`prod:id` 属性を持つようにスキーマを編集することにより属性の名前のためにプレフィックスを作成することができます（次を参照してください：ノード名のマッピング）。

Chapter 7

関数

7 関数

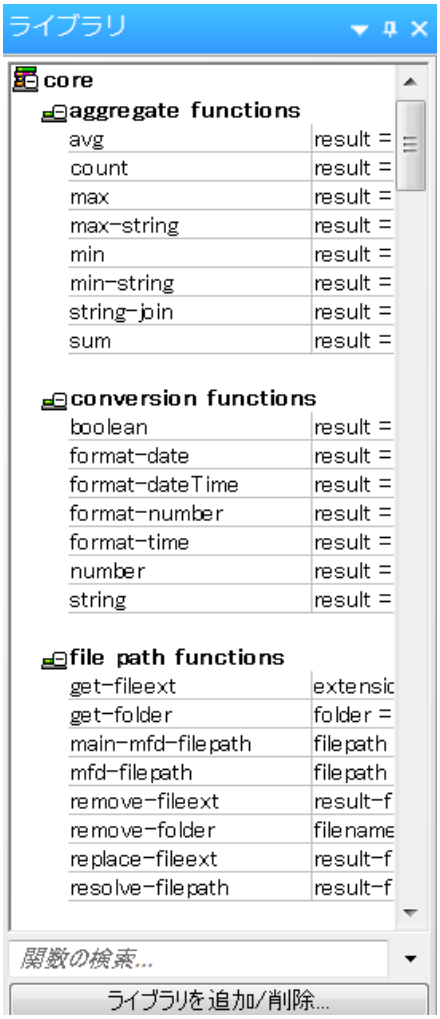
関数は、特定のノードに従いデータを変換するパワフルな方法を示します。 このセクションは、(MapForce のビルドインの関数または、外部のソースからの再利用にかかわらず) 関数との作業方法について説明します。以下のロードマップを使用して特定のタスクに関連した関数にアクセスしてください:

...を実行するには	以下を参照してください...
MapForce 内の関数との作業方法を学ぶ	関数と作業
マッピング内のオペレーションの優先順位の変更方法を学ぶ	優先コンテキスト ノード アイテム
MapForce 内で自身の関数を作成する	ユーザー定義関数
外部 XSLT 関数を使用する	カスタム XSLT 関数の追加
すべてのビルドイン MapForce 関数を表示する、または特定の関数の詳細を検索する	関数ライブラリ リファレンス

7.1 関数と作業

マッピング内で関数を使用する:

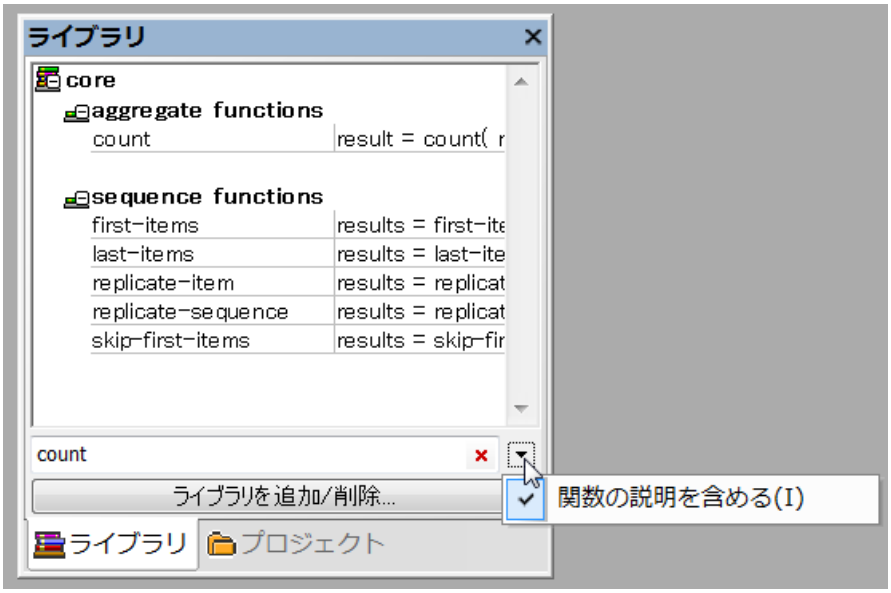
1. 変換言語を選択する (変換言語の選択を参照してください)。
2. ライブラリウィンドウ内の必要な関数をクリックし、マッピングエリアにドラッグします。



ライブラリウィンドウ内で関数を検索する:

1. ライブラリウィンドウの下の部分にある、テキストボックスに関数名を入力します。

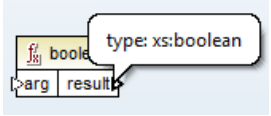
デフォルトでは、MapForceは、関数を名前と説明テキストにより検索します。関数の説明を検索から除外する場合は、**関数説明を含める**オプションを無効化します。



検索をキャンセルする場合、「**Esc**」キーを押す、または × をクリックします。

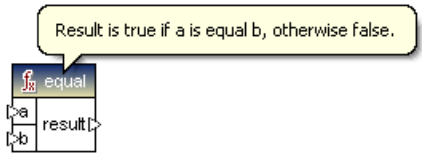
関数入力 または 出力引数のデータ型を確認する:

1. 「**ヒントの表示**」 ツールバーボタンが有効化されていることを確認してください。
2. 関数の引数部分にマウスを移動します。



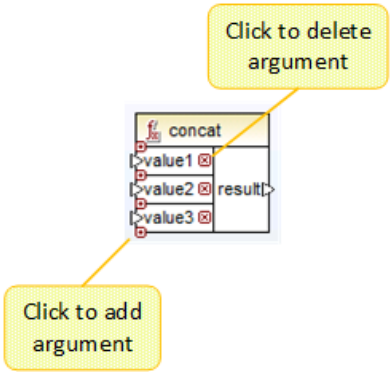
関数の詳細を表示する：

1. **「ヒントの表示」** ツールバーボタンが有効化されていることを確認してください。
2. 関数の上にあるマウスのポイントを移動します（『ライブラリ』ペインとマッピングエリアの両方でこの機能を使用することができます）



(適用することのできる関数のための) 関数引数の追加と削除 ：

- 追加または削除するパラメーターの横の **「パラメーターの追加」**（⊙）または **「パラメーターの削除」**（⊗）をクリックします。



⊙ シンボルに接続をドロップすると、パラメーターが自動的に追加され、接続します。

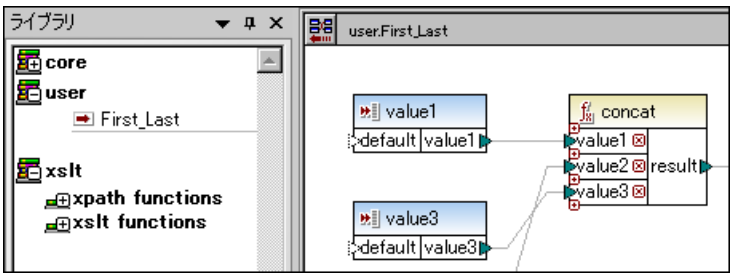
現在アクティブなマッピング内で関数の全ての発生を検索する：

- ライブラリウィンドウを右クリックして、コンテキストメニューから **「全ての呼び出しの検索」** を選択します。検索の結果はメッセージウィンドウに表示されます。

7.2 ユーザー定義関数

MapForce ではメインマッピングウィンドウと同じように、視覚的な方法によりユーザー定義関数を作成することができます。

これらの関数はライブラリウィンドウにて利用できるようになり（例：First_Last）、デフォルトで用意されている関数と同様の方法で使用することができます。関数を使用することで、ブロック単位でマッピングの管理を行い、別のマッピングにて再利用することができます。



ユーザー定義関数は、メインのマッピングとともに ***.mfd** ファイルに収められます。

ユーザー定義関数では、**入力**ならびに**出力**コンポーネントを使用することで、メインマッピング（又は他のユーザー定義関数）からユーザー定義関数へ情報が受け渡され、ユーザー定義関数におけるマッピングの結果が返されます。

ユーザー定義関数では、例えばルックアップ関数を実装する際に便利な XML スキーマ（ユーザー定義関数内部で使用される）「ローカル」ソースコンポーネントを含めることができます。

ユーザー定義関数には任意の数の入力ならびに出力を含めることができ、これら入出力コンポーネントは、単純型の値、または XML ノードとして定義することができます。

ユーザー定義関数は、以下のような状況で利用することができます：

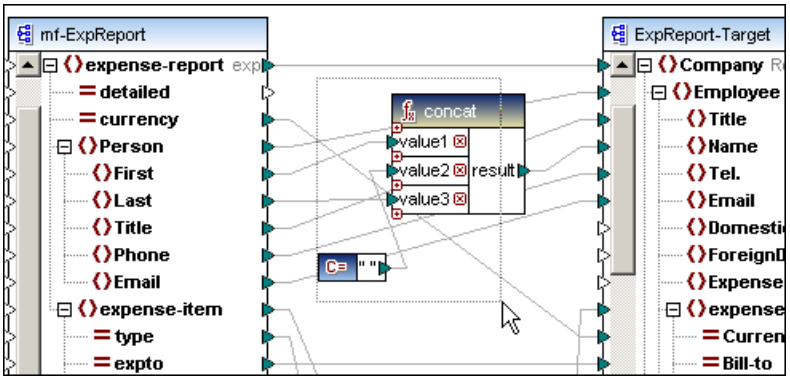
- 複数の関数を 1 つのコンポーネントへまとめる。例えば、特定のフィールドをフォーマットしたり、値のルックアップを行う
- これらコンポーネントを任意の数だけ再利用する。
- マッピングファイルをライブラリとしてロードすることで他のマッピングからユーザー定義関数をインポートする。
- インライン関数 を使用することで、複雑なマッピングを個々に編集可能な細かいパーツへ分割する。
- 再帰的なユーザー定義マッピング を作成することで、再帰的なスキーマをマッピングする。

ユーザー定義関数は何も無い状態から作成することができるほか、マッピングタブに既に表示されている関数からも作成することができます。

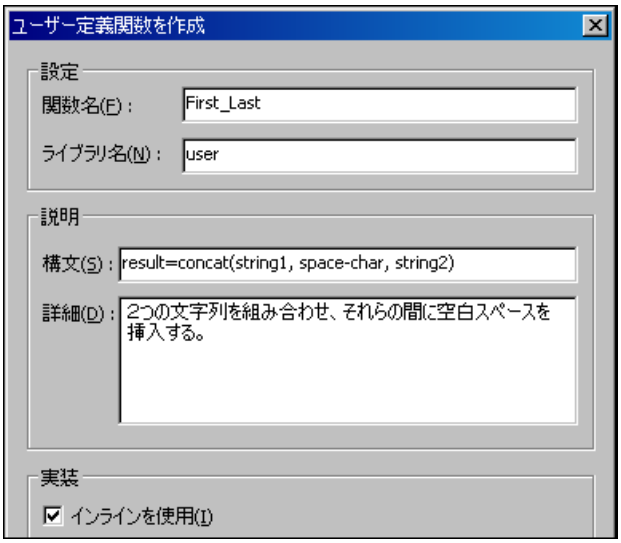
以下のサンプルは ...\MapForceExamples\Tutorial\ フォルダーに収められている**Tut-ExpReport.mfd** ファイルを使用します。

既存のコンポーネントからユーザー定義関数を作成する：

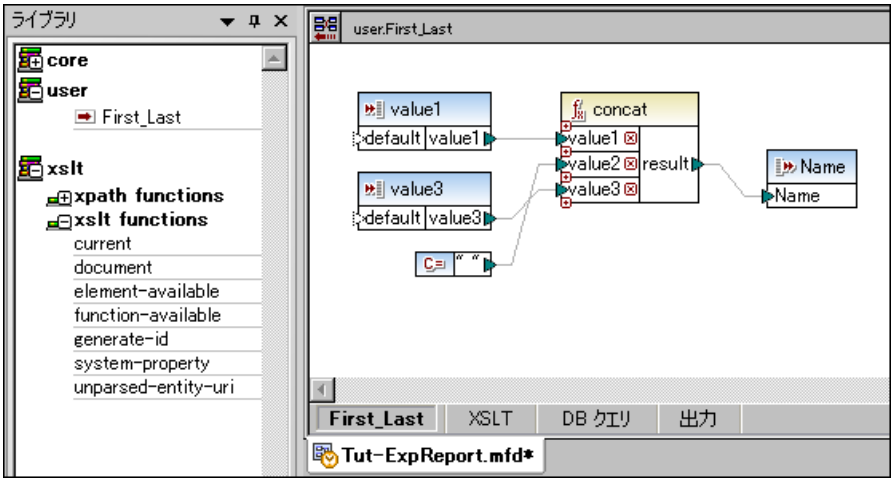
1. マウスポインターをドラッグして、"concat" と定数コンポーネントを選択します（Ctrl キーを押下しながらコンポーネントを複数選択することもできます）。



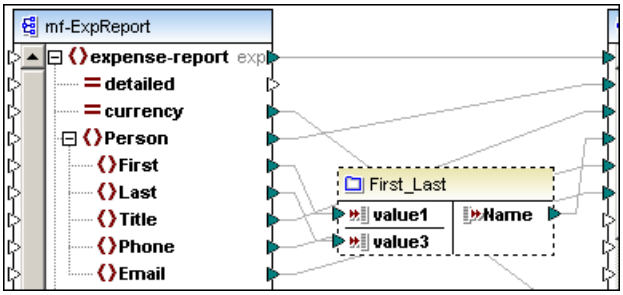
2. メニューオプションから「関数 | 選択からユーザー定義関数作成」を選択します。
3. 新たなユーザー定義関数の名前を入力します（例：First_Last）。
   メモ：ユーザー定義関数の名前に使用可能な文字は、大文字と小文字のアルファベット（a-z, A-Z）、数値（0-9）、そしてアンダースコア（_）、ハイフォンダッシュ "-"と コロン ":" となります。
4. 構文ならびに詳細フィールドにて、新たな関数に関する情報を入力することができ「OK」ボタンをクリックすることで確定します。入力されたテキストは、関数上にマウスオーバーされた際にツールチップとして表示されます。"user" というライブラリ名がデフォルトで与えられます。勿論、このフィールドで独自のライブラリ名を定義することもできます。



関数グループを構成する個々の要素が、関数名のタブに表示されます。新たに作成されたライブラリ "user" がライブラリペインに表示され、その下に "First_Last" という関数名が表示されます。



ホームボタン をクリックすることで、メインマッピングウィンドウへ戻ることができます。複数あったコンポーネントが First_Last という名前で 1 つの関数コンポーネントへ統合されたのか確認できます。入力ならびに出力パラメーターは自動的に接続されていることが確認できます。



インラインのユーザー定義関数は破線の境界線により表示されます。詳細に関しては[インラインユーザー定義関数](#) を参照ください。

ライブラリペインからマッピングウィンドウへ関数名をドラッグアンドドロップすることで、その関数をマッピングにて使用することができます。別のマッピングで関数を使用する方法については、[ユーザー定義関数の再利用](#) を参照ください。

ユーザー定義関数を開く

ユーザー定義関数を開くには以下を行います：

- ユーザー定義関数コンポーネントのタイトルバーをダブルクリックする。
- ライブラリペインにて、目的のユーザー定義関数をダブルクリックする。

これにより、関数の中にあるコンポーネントが、関数名のタブにて表示されます。ホームボタン をクリックすることで、メインのマッピングへ戻ることができます。異なる *.mfd ファイルのユーザー定義関数を (メインマッピングウィンドウにて) ダブルクリックすることで、その MFD ファイルが新たなタブにて表示されます。

ユーザー定義関数のナビゲーションを行う

様々なタブ又はユーザー定義関数のタブを MapForce にてナビゲーションすることで、移動の履歴が自動的に生成され、進む/戻るアイコンをクリックすることで複数のタブ間を移動することができます。履歴はセッションごとに管理されており、複数の MFD ファイル間を移動することもできます。

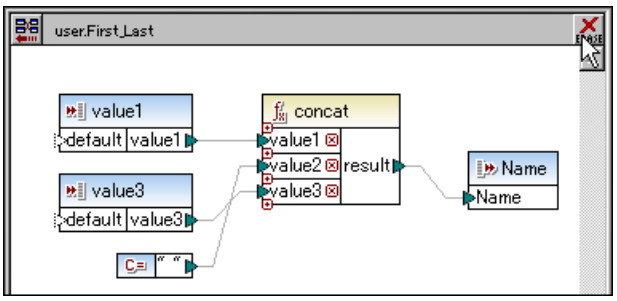
 ホームボタンをクリックすることで、ユーザー定義関数からメインマッピングのタブへ戻ります。

 戻るボタンにより、履歴内にある 1 つ前のマッピングへ移動します。

 進むボタンにより、履歴内にある 1 つ後のマッピングへ移動します。

ライブラリからユーザー定義関数を削除する：

1. ライブラリウィンドウにて目的のユーザー定義関数をダブルクリックします。
2. タイトルバーの右端にある **「削除」**ボタンをクリックすることで、その関数を削除することができます。



ユーザー定義関数を再使用 (インポート) する。

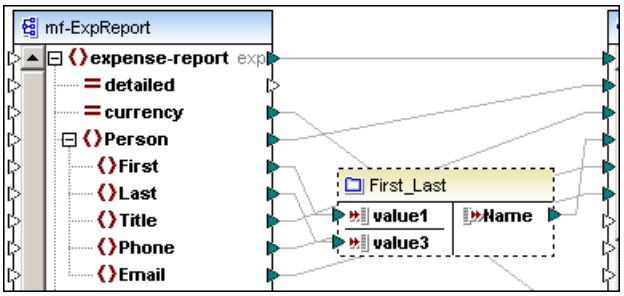
あるマッピングで定義されたユーザー定義関数は、他のマッピングにてインポートすることができます：

1. ライブラリウィンドウの下部に配置されている**ライブラリの追加/削除**をクリックします。
2. 追加ボタンをクリックし、インポートするユーザー定義関数を含んだ ***mfd** ファイルを選択します。 ユーザー定義関数はライブラリウィンドウに表示されます。デフォルトのライブラリ名を使用してユーザー定義関数が作成された場合、ライブラリ名は "user" です。それ以外の場合、ユーザー定義関数を作成した際に指定したライブラリ名を検索してください。
3. インポートされた関数をマッピングへドラッグすることで、その関数を利用することができます。

> ユーザーを作成する場合、複数の *mfd ファイル内で同じライブラリ名 を指定することができます。この場合、使用することのできるソースからの関数は、ライブラリウィンドウ内の同じライブラリ名の下にリストされます。しかしながら、現在アクティブな ドキュメント内の関数のみがダブルクリックにより編集することができます。

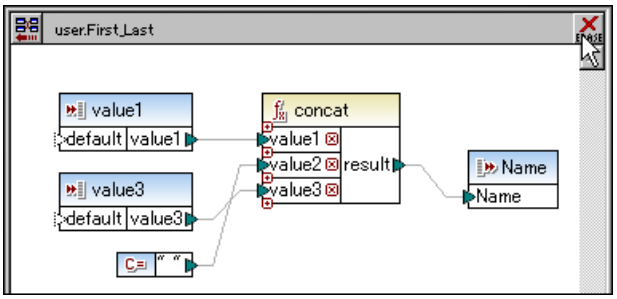
インポートされた関数内の変更はインポートされるマッピングにもライブラリ***.mfd** ファイルを保存する際に適用されます。

ユーザー定義関数内におけるパラメーターの順序



ユーザー定義関数内のパラメーターの順序は直接影響することができます：

- 入力ならびに出力パラメーターの順序は、上部から下部にかけて引数の位置（パラメーターコンポーネントの左上ピクセルの位置）により決定されます。
- 2 つのパラメーターが 垂直の向きに同じ高さにある場合、左側にあるパラメーターが先に先に来る引数として処理されます。
- 仮に 2 つのパラメーターが全く同じ位置にある場合、順序の決定に内部的に指定されているコンポーネント ID が自動的に使用されます。



メモ：

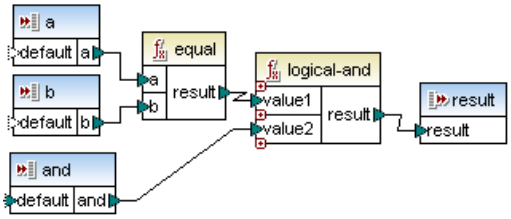
- コンポーネントの位置ならびに大きさの変更は「やり直し不可」なアクションです。
- 新たに追加された入力ならびに出力コンポーネントは、最後の入力または出力コンポーネントの下部にて作成されます。
- 複合型と単純型のパラメーターを混合することもできます。パラメーターの順序は、コンポーネントの位置により決定されます。

7.2.1 関数のパラメーター

関数パラメーターは、**入力**ならびに**出力コンポーネント**としてユーザー定義関数内にて表示されます

入力コンポーネント/パラメーター：**a** ならびに **b**

出力コンポーネント/パラメーター：**result**

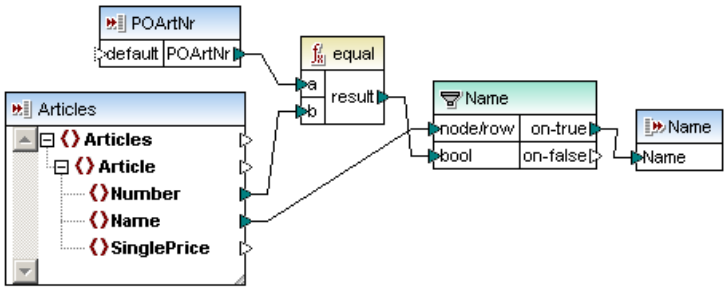


入力パラメーターはメインマッピングからユーザー定義関数へ値を受け渡すために使用され、出力パラメーターはユーザー定義関数からメインマッピングへ値を返すために使用されます。ユーザー定義関数は、他のユーザー定義関数からも呼び出すことができるという点に留意してください。

単純型のパラメーターと複合型のパラメーター

ユーザー定義関数の入力ならびに出力パラメーターは異なる型で定義することができます：

- 単純型の値（例：string や integer）
- 複合型のノードツリー（例：属性や子ノードを含むXML 要素）



入力パラメーターの**POArtNr** では、"string" 単純型の値が受け渡されます。

入力パラメーターの**Articles** は、複合型のXML ドキュメントノードツリーです。

出力パラメーターの**Name** はstring の単純型です

メモ：

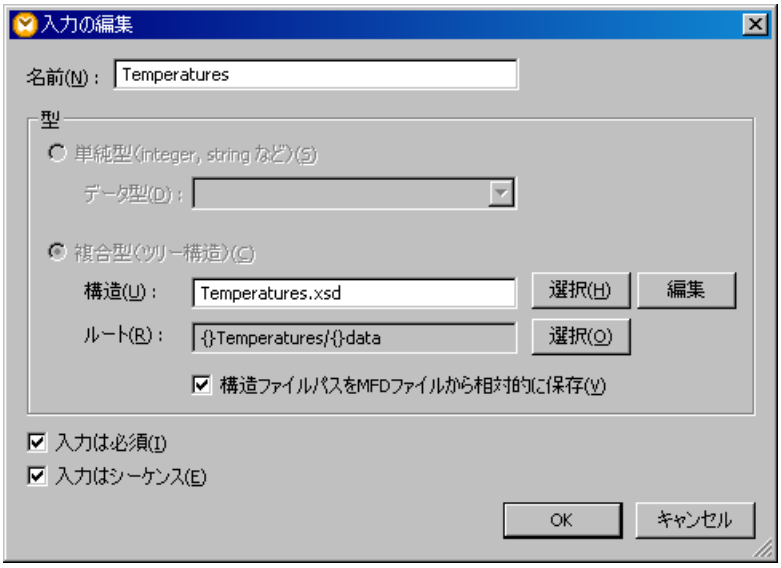
上に示されるユーザー定義関数は、..\MapForceExamples フォルダー内に収められている **PersonListByBranchOffice.mfd** ファイルにより確認することができます。

シーケンス

シーケンスは値の範囲またはシーケンスにより構成されるデータのことです。コンポーネントパラメーターダイアログボックスでは、ユーザー定義された単純ならびに複合型の**パラメーター**（入力と出力）をシーケンスとして定義することができます。

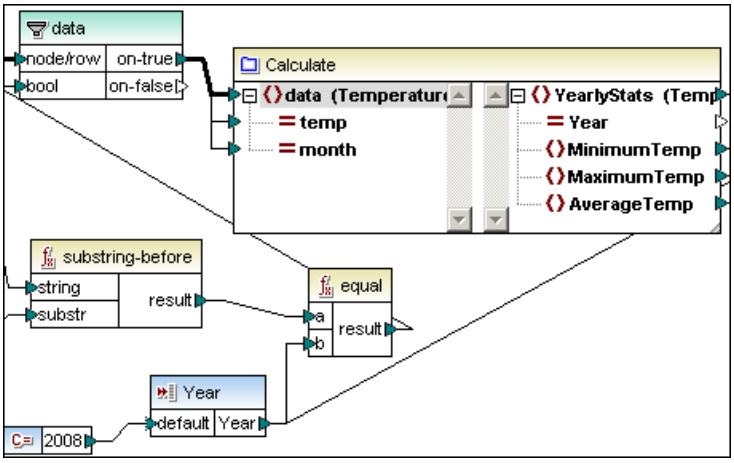
min や max、avg といった集計関数とこのような種類の入力を組み合わせることで、入力シーケンスから特定の値を 1 つ得ることができます。

「入力はシーケンス」チェックボックスが有効になっている場合、コンポーネントは入力をシーケンスとして扱います。このチェックボックスが無効になっている場合、入力は単一の値として扱われます。

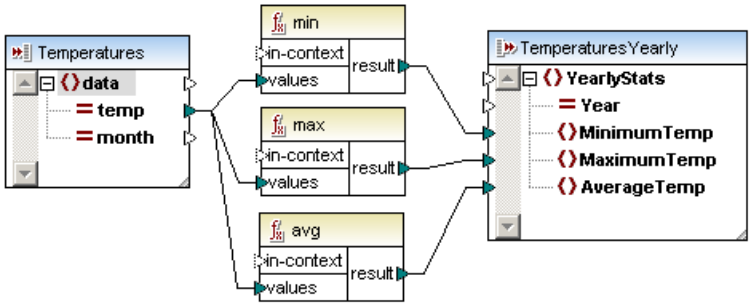


入力データの種類がシーケンスかそうでないかにより、関数が呼ばれる**回数**が決定されます。

- **シーケンス**パラメーターへの接続が作成されている場合、ユーザー定義関数への呼び出しが**一度**だけ行われ、シーケンス全体がユーザー定義関数へ渡されます。



スクリーンショットには、..\MapForceExamples フォルダーに収められている "**InputIsSequence.mfd**" マッピングに含まれているユーザー定義関数の "Calculate" が示されています。以下に示される**Temperatures**入力コンポーネントはシーケンスとして定義されています。



- **非シーケンス**パラメーターとして接続が行われている場合、シーケンス内にある各アイテムごとに、ユーザー定義関数が**一度**呼ばれます。

メモ:
ユーザー定義関数がインライン型として定義されている場合、入力/出力パラメーターのシーケンス設定は無視されます。

空のシーケンスを非シーケンスパラメーターへ接続すると、関数は全く呼ばれなかったものとして扱われます!

ソースの構造が**オプション**のアイテムであるか、フィルターの条件によりマッチしたアイテムが得られなかった場合に、上に記述したような状況が起こりえます。このような状況を回避するには、関数への入力を行う前にsubstitute-missing 関数を使用してシーケンスが空にならないことを保証するか、パラメーターをシーケンスとしてセットし、関数内部でシーケンスが空だった時の処理を追加する必要があります。

関数により複数の値が出力コンポーネントへ与えられ、出力コンポーネントがシーケンスとして定義されていない場合、最初の結果だけが関数が呼ばれた際に使用されます。

7.2.2 インラインと一般的なユーザー定義関数

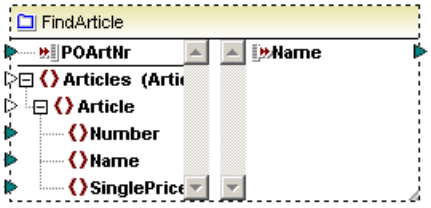
インライン関数はコードが生成された際に実装されるという点で、通常の関数とは根本的に異なるものです。

- **インライン**型関数のコードは、ユーザー定義された関数が呼び出される/使用される**全ての場所**に挿入されます。
- **通常**関数のコードは、関数呼び出しとして実装されます。

  従って、インライン関数は関数内部の実装により置き換えられたかのような振る舞いをします。この仕様は、複雑なマッピングを小さな内部パーツに置き換える際に適したものです。

メモ:
インライン関数を使用することで、生成されたプログラムコードの量が飛躍的に増加します!ユーザー定義関数に対応するコードが、関数が呼び出される/使用される全ての箇所に挿入され、通常の関数を使用するのに比べてコードの大きさが飛躍的に増加します。

インラインユーザー定義関数は破線の境界により表示されます:



インラインユーザー定義関数では、以下が**サポート**されます:
- 関数内部における**複数の出力コンポーネント**

以下はサポートされません:
- パラメーターに対する優先コンテキストの設定
- インラインユーザー定義関数に対する再帰的な呼び出し

一般的な ユーザー定義関数 (**実線**の境界線で表示される非インライン関数):

一般的な（非インライン）ユーザー定義関数では以下の項目が**サポート**されます:

- 関数内における**単一の出力コンポーネント**
- **再帰的な**呼び出し（再帰を終了する分岐に接続されているIf-Else 条件を与えるなどして、終了条件を指定する必要があります）
- 優先コンテキストパラメーター

メモ:

通常の関数において複数の出力コンポーネントは**サポートされません**が、**作成することはできます**。しかし、コードの生成やマッピング結果のプレビューを行おうとする際にエラーメッセージが表示されます。

関数内で再帰を使用しない場合、関数の種類を「インライン」に変更することができます。

以下の項目はサポートされません:

- 非シーケンスの**入力**コンポーネントからフィルターコンポーネントへの直接接続
- シーケンスや、単一の入力コンポーネントから接続された集計関数（例：exists、substitute-missing、sum、group-by）

コード生成

一般的なユーザー定義関数の実装は、呼び出し可能なXSLT テンプレートまたは関数として一度生成されます。入力がパラメーターとして渡され、出力が関数（コンポーネント）の戻り値となっている箇所では、対応するユーザー定義関数コンポーネントの**関数呼び出し**コードが生成されます。

ランタイムにおいて、全ての入力パラメーター値は最初に評価され、各入力データに対して関数の呼び出しが行われます。この処理の詳しい情報については関数パラメーターのセクションを参照ください。

ユーザー定義関数の「種類」を変更する:

1. ユーザー定義関数をダブルクリックして、関数を構成するコンポーネントを確認します。
2. メニューオプションから「関数 | 関数設定」を選択し、「インラインを使用」チェックボックスにチェックを入れます。

ユーザー定義関数と全てコピー接続

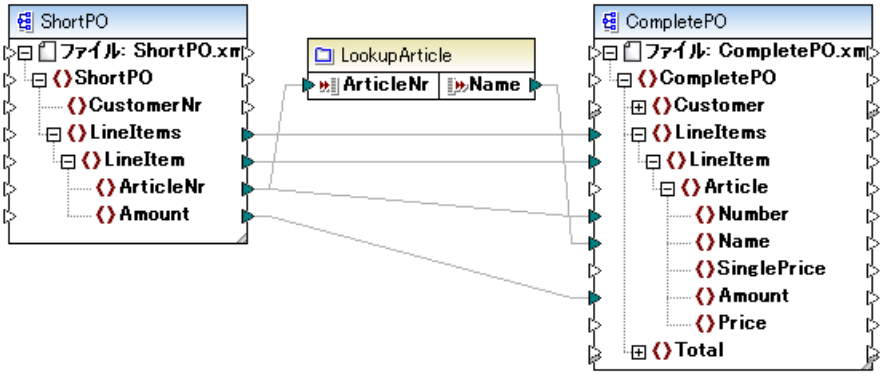
スキーマと複合型のユーザー定義関数パラメーター間にて全てコピー接続を作成する場合、両方のコンポーネントが同一のスキーマをベースにしたものである必要があります！同一のルート要素である必要はありません。複合型出力コンポーネント－定義のセクションからサンプルを確認することができます。

7.2.3 単純型のルックアップ関数を作成する

以下のサンプルは **...\ MapForceExamples** フォルダーに収められている **lookup- standard.mfd** ファイルにて確認することができます。

目的：
以下にあるような、一般的なルックアップ関数を作成する：

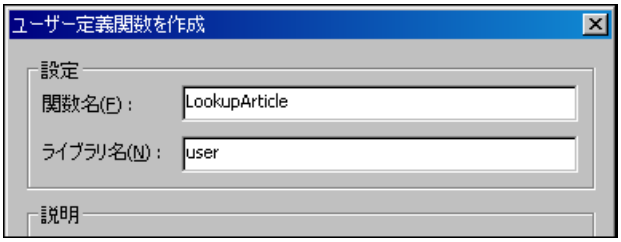
- Articles XML ファイルから得られるArticles/Number データを、別のXML ファイル（この例ではShortPO）にあるArticle の番号と比較します。



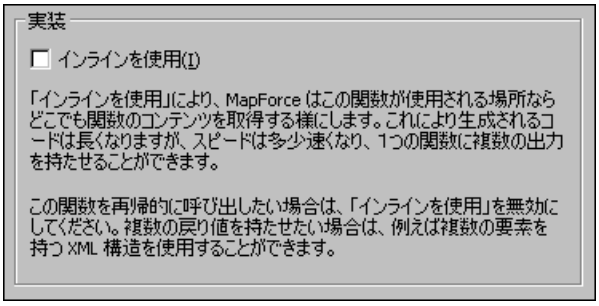
- ShortPO.xsd を挿入し、ソースXML ファイルとしてShortPO.xml を割り当てます。
- CompletePO.xsd スキーマファイルを挿入し、ルート要素としてCompletePO を選択します。
- 以下に記述されている方法により、新たなユーザー定義関数を挿入します。

ユーザー定義関数の作成：

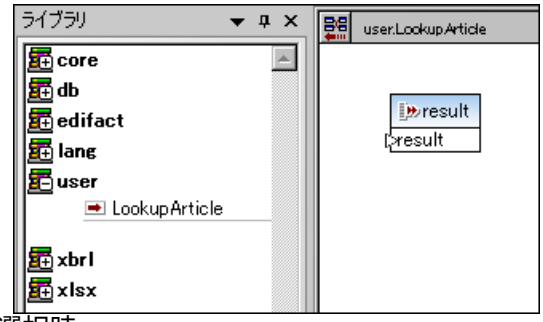
1. メニューオプションから「関数 | ユーザー定義関数の作成」を選択します。
2. 関数の名前を入力します（例：LookupArticle）。



3. 「インラインを使用」チェックボックスのチェックを外し、「OK」ボタンをクリックして確定します。



"result" という名前の出力コンポーネントだけがタブの中に表示されます。

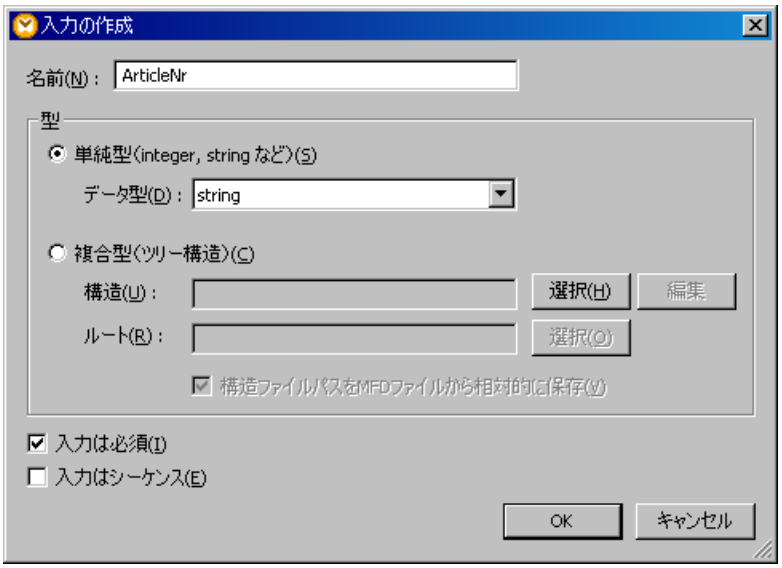


Java 選択時

これがユーザー定義関数を定義するための作業エリアになります。

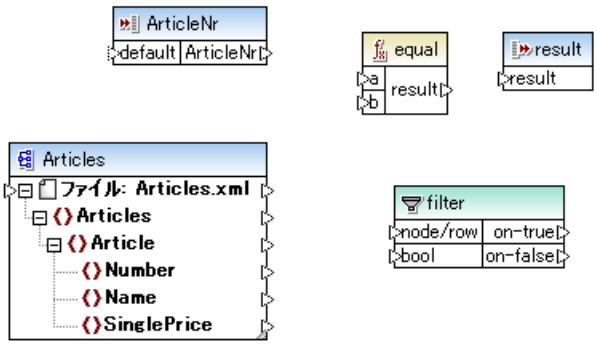
"user" という名前のライブラリが、ライブラリペインにて新たに作成され、"LookupArticle" という名前の関数がその下に表示されます。

4. **スキーマ/XML ファイルの挿入**アイコン をクリックすることで、**Articles** スキーマを挿入し、同名のXML ファイルをデータソースとして選択します。
5. **入力挿入**アイコン をクリックして、入力コンポーネントを挿入します。
6. 入力パラメーターの名前（この例ではArticleNr）を入力し、**「OK」**をクリックします。



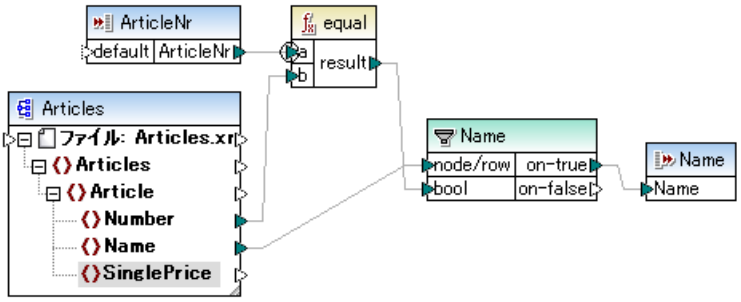
ユーザー定義関数の入力として機能するユーザー定義関数の入力アイコンが表示されます。

7. core ライブラリのlogical 関数グループから**"equal"** コンポーネントをドラッグすることで、新たな関数を挿入します。
8. フィルターの挿入アイコン をクリックして、**フィルター**コンポーネントを挿入します。



以下に示されているマッピングを参考に、ユーザー定義関数のマッピングを行います。以下の点に注意してください：

9. **a** パラメーターを右クリックして、コンテキストメニューから**優先コンテキスト**を選択します。
10. 出力コンポーネントを**ダブルクリック**して、出力パラメーターの名前を変更します（この例では、"**Name**" に変更します）。



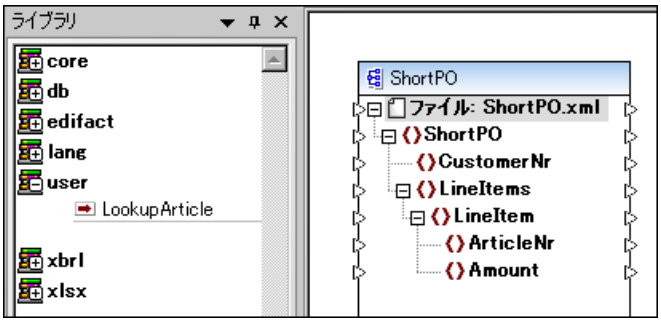
これでユーザー定義関数の定義が完了しました。

メモ:
入力または出力コンポーネントをダブルクリックすることでダイアログボックスが表示され、入力パラメーターの名前やデータ型を変更したり、関数に入力アイコンが表示されるか（入力は必須）、また入力をシーケンスとして扱うかを指定することができます。

このユーザー定義関数は、以下のような特徴を持ちます：

- ShortPO XML ファイルからデータを受け取るArticleNr という名前の**入力**を 1 つ備えている。
- ShortPO ArticleNr の値と、（この目的のために関数に挿入されている）Articles XML *インスタンスファイル*から得られたArticle/Number を**比較**する。
- ***フィルター***コンポーネントを使用することで、比較の結果がtrue の場合にArticle/Name レコードを出力コンポーネントへ渡す。
- Name という名前の出力コンポーネントを 1 つ備えており、Article Name レコードをCompletePO XML ファイルへ渡す。

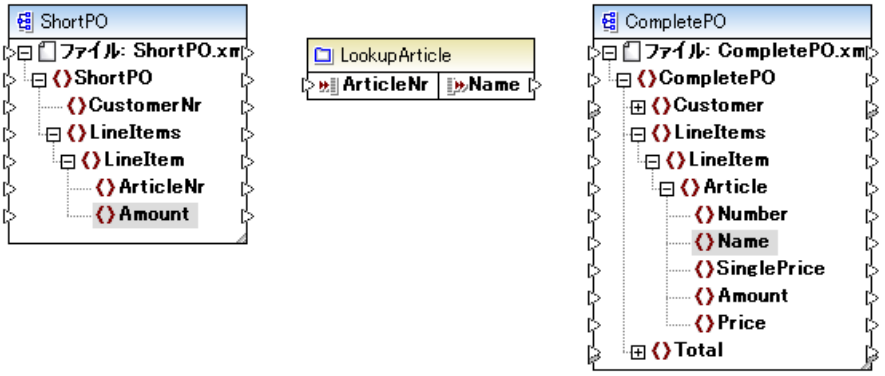
11. ホームアイコン  をクリックしてメインマッピングへ戻ります。
ユーザー定義関数のLookupArticle が、user ライブラリ以下に表示されます。



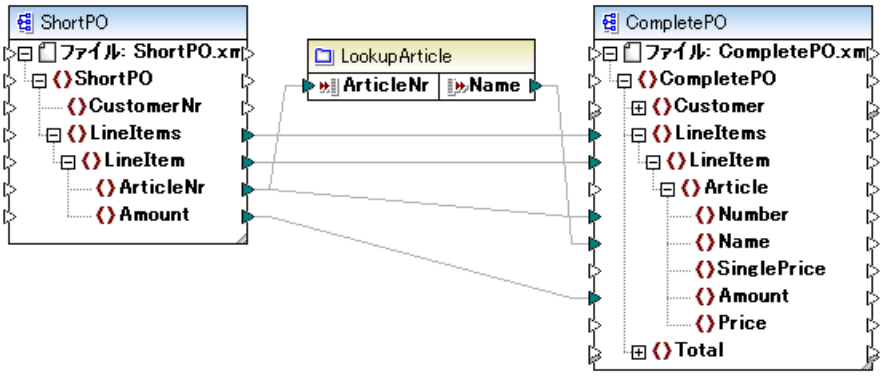
12. LookupArticle 関数をマッピングウィンドウへドラッグします。

ユーザー定義関数が以下のように表示されます。

- タイトル（関数バー）に、関数名の "LookupArticle" が表示されます。
- 入力ならびに出力アイコンが、それぞれの名前とともに表示されます。



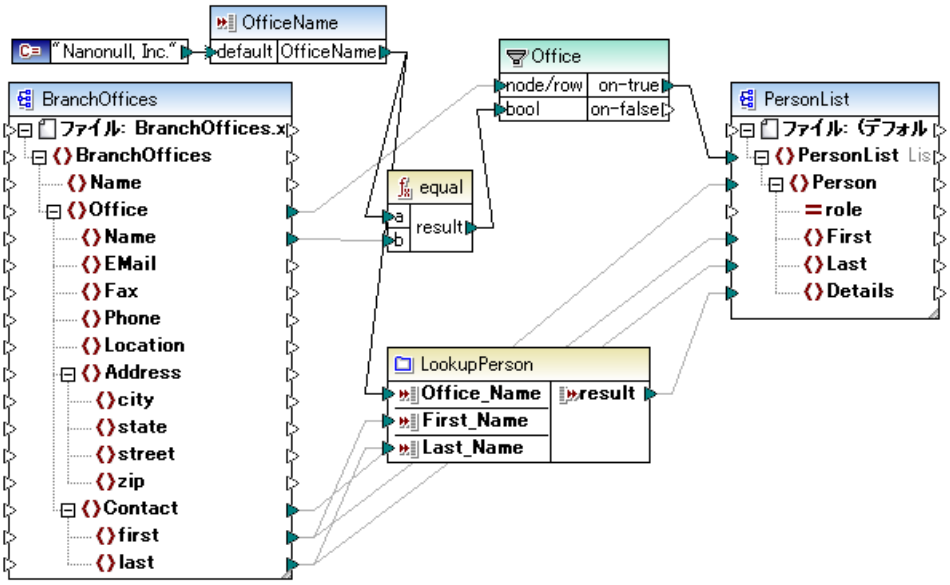
13. 以下のスクリーンショットに示されるようにマッピングを作成し、出力タブをクリックしてマッピングの結果を確認します。



7.2.4 ユーザー定義関数 - サンプル

<Documents>\ Altova\ MapForce2017\ MapForceExamples フォルダーに収められている **PersonListByBranchOffice.mfd** ファイルを例にして、以下にある機能の詳細を解説します：

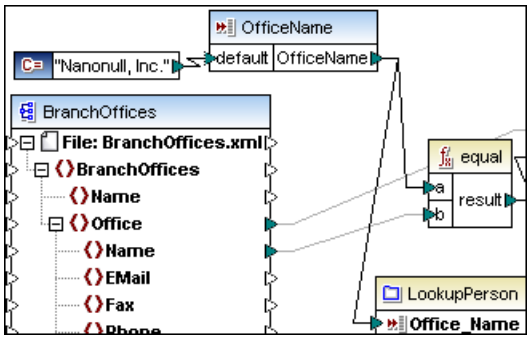
- ネストされたユーザー定義関数（例：**LookupPerson**）
- 文字列の出力を生成するルックアップ関数（例：**LookupPerson**）
- **デフォルト**値が与えられている**オプション**の入力パラメーター（例：LookupPerson コンポーネントに含まれる **EqualAnd** コンポーネント）
- 生成されたマッピングコードを実行する際に、コマンドラインの引数としても使用することができる入力パラメーター



構成することのできる入力パラメーター

入力パラメーター (OfficeName) では、以下の方法により、マッピングが実行された際にデータを渡すことができます：

- 生成されたコードを実行する際に、**コマンドライン**の引数として与える (例：Mapping.exe /OfficeName "Nanonull Partners, Inc.")。
- 内蔵の実行エンジンを使って出力ウィンドウでデータのプレビューをする際に、**プレビュー**値として与える。

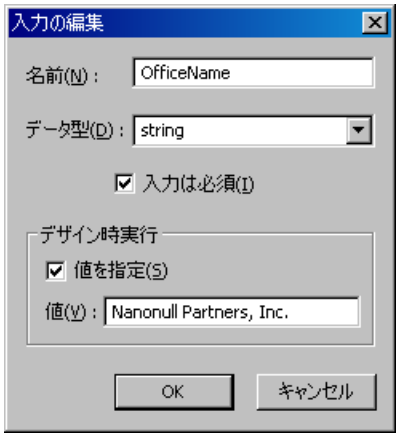


入力値を定義する：

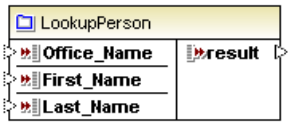
1. 入力コンポーネントをダブルクリックして、デザイン時実行グループにある「値」テキストボックスにて別の値 (例："Nanonull Partners, Inc.") を入力し、**「OK」**をクリックして確定します。
2. 出力タブをクリックして、マッピングの結果を確認します。
   従業員の一覧が表示されます。

   このダイアログボックスで入力されたデータは、**「プレビュー」**モードで (つまり出力タブをクリックした際に) のみ使用される点に注意してください。値が入力されなかった場合、または「値を使用」チェックボックスが無効になっている場合、入力アイコンへのデータマッピングが使用されます。

   詳細については、入力コンポーネントを参照ください。



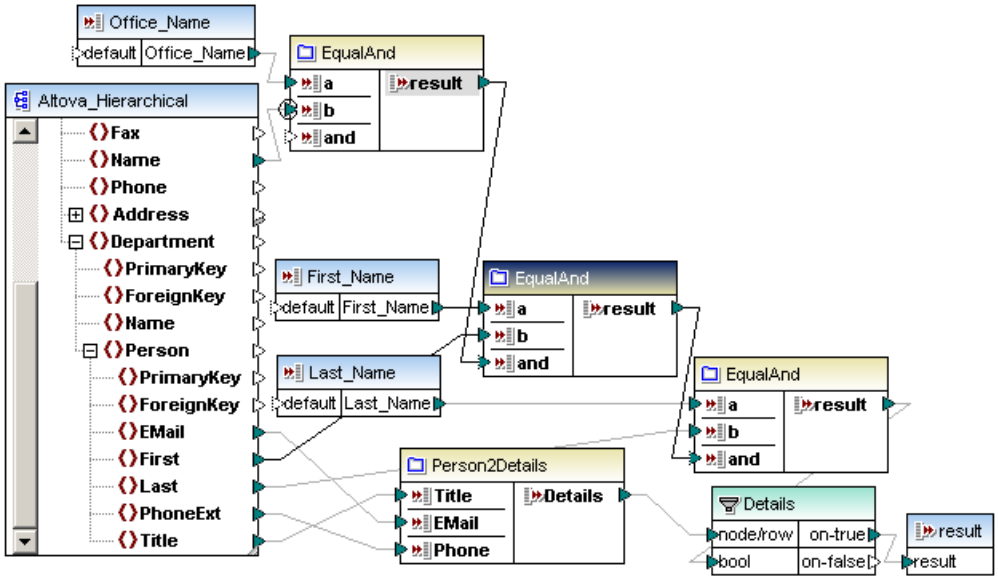
LookupPerson コンポーネント



ユーザー定義関数のコンポーネントをダブルクリックすることで、以下に示されるコンポーネントが表示されます。このコンポーネントは以下のように動作します：

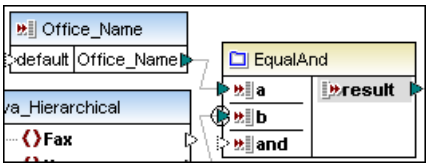
- **入力**コンポーネントと**EqualAnd** ユーザー定義関数コンポーネントにより、Office とBranchOffice.xml のFirst ならびにLast 名が、Altova_Hierarchical.xml ファイルにある同じ名前のフィールドと**比較**されます。
- **Person2Details** ユーザー定義関数を使用することで、Email、PhoneExt、Title アイテムが**連結**されます。
- EqualAnd から得られた結果が全てtrue の場合、処理された従業員のデータを**出力**コンポーネントへ**受け渡し**ます（フィルターコンポーネントにてture かどうかの判定が行われます）。

ユーザー定義関数は、（極論すれば空の文字列でも）常に何らかの値を出力します！この例では、フィルターコンポーネントのbool 値がfalse となった場合、Peson2Details コンポーネントの代わりに空の文字列が出力されます。



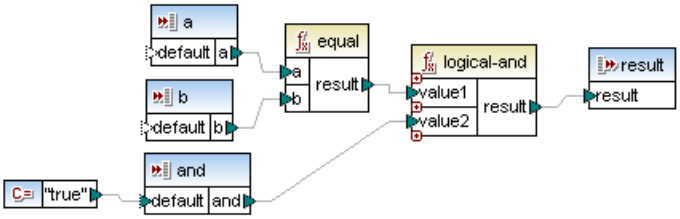
- 3 つの**入力**コンポーネントであるOffice_Name、First_Name、Last_Name によりBranchOffices.xml ファイルからのデータが受け取られます。
- **EqualAnd** コンポーネントにより 2 つの値が比較され、更にデフォルト値を持った**オプション**の比較値が与えられます。
- Person2Details コンポーネントにより、3 つのフィールドが組み合わされ、その結果がフィルターコンポーネントへ渡されます。

EqualAnd コンポーネント



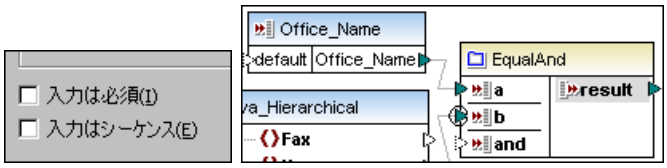
ユーザー定義関数のコンポーネントをダブルクリックすることで、以下に示されるコンポーネントが表示されます。このコンポーネントは以下のように動作します：

- **a** と**b** という 2 つの入力パラメーターを比較し、その結果をlogical-and コンポーネントへ渡します。**b** パラメーターが優先コンテキストとして定義されていることに注目してください（アイコンを右クリックして指定することができます）。これにより、入力パラメーター**a** により与えられた特定オフィスの従業員データが最初に処理されるようになります。
- 最初の比較結果と、オプションの入力パラメーターである "and" を**logical-and** により比較します。
- 最終的にlogical-and から得られた比較結果のboolean 値が出力パラメーターへ渡されます。



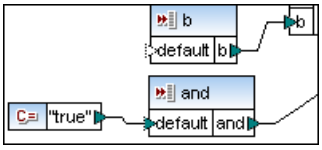
オプションのパラメーター

上に示されるEqualAnd ユーザー定義関数の "and" パラメーターをダブルクリックして、「入力は必須」チェックボックスのチェックを外すことで、そのパラメーターをオプションと指定することができます。



「入力は必須」チェックボックスの**チェックが外されている**場合：

- このユーザー定義関数の入力アイコンに対して、マッピングを行う必要が無くなります。例えば、LookupPerson 関数内にある最初のEqualAnd 関数の**and** 入力パラメーターはマッピングされていません。
- ユーザー定義関数内部にあるコンポーネントからマッピングを行うことで、**デフォルト**値を与えることができます。例えば、"true" という値を含む定数コンポーネントを使用することができます。

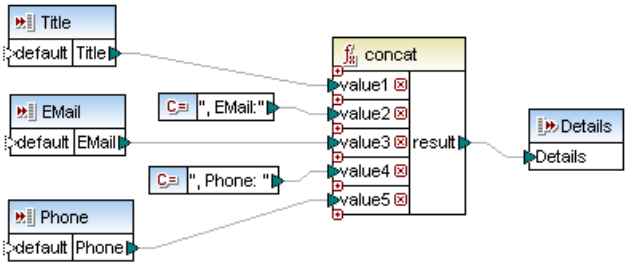


- 他のアイテムからオプションとなっている入力コンポーネントへマッピングを行うことで、デフォルト値の値がオーバーライドされます。例えば、LookupPerson 関数内部にある 2 番目の EqualAnd 関数には、最初の EqualAnd ユーザー定義関数の "result" パラメーターの値が与えられます。

Person2Details コンポーネント

このユーザー定義関数コンポーネントをダブルクリックすることで、以下に示されるコンポーネントが表示されます。このコンポーネントは以下のように動作します：

- 3 つの入力を連結して、result 文字列を出力パラメーターへ渡します。
- 出力パラメーターをダブルクリックすることで、パラメーター名を変更したり (Details)、データ型を選択することができます (string)。



7.2.5 複合型のユーザー定義関数 - XML ノードを入力に使用

以下のサンプルは…\MapForceExamples フォルダーに収められている**lookup-udf-out.mfd** ファイルにて確認することができます。このセクションでは、複合型の入力コンポーネントを持つインラインのユーザー定義関数の作成方法について説明します。

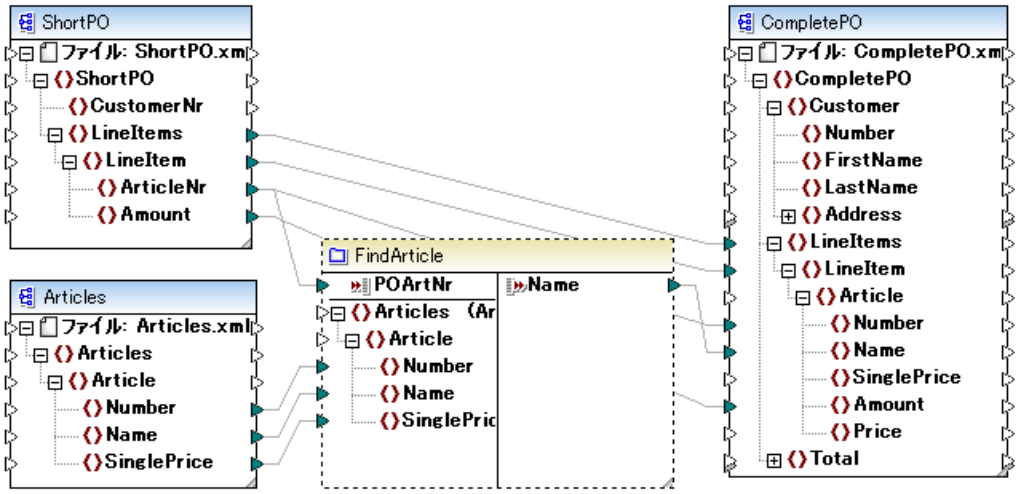
ユーザー定義関数の "FindArticle" は 2 つのパーツから構成されることに注意してください。

以下の入力パラメーターを含む左半分：

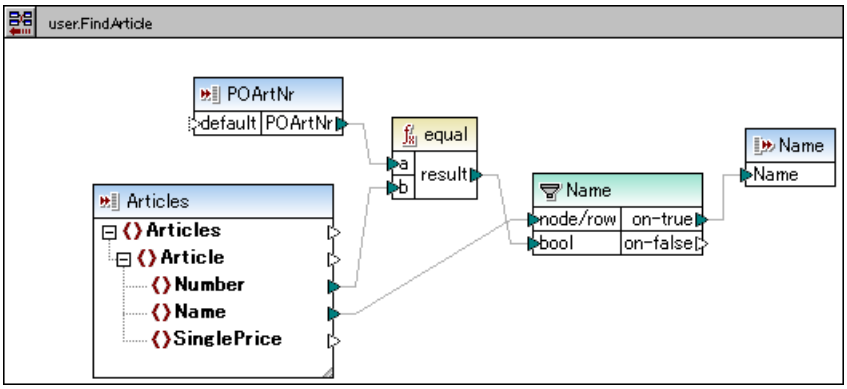
- 単純型の入力パラメーター：**POArtNr**
- XML 子ノードへ直接マッピングする複合型入力コンポーネント **Articles**

以下のコンポーネントを含む右半分：

- **"Name"** という名前を持つ単純型出力パラメーター



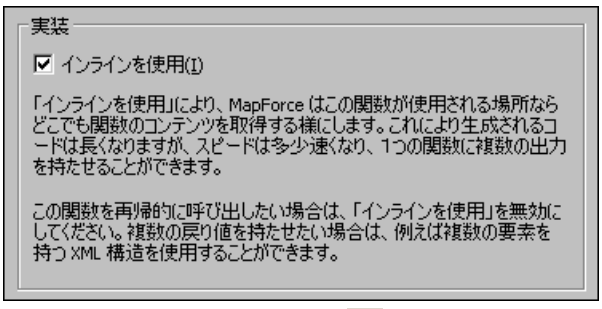
.ユーザー定義関数を行使絵するコンポーネントを以下のスクリーンショットに示します。左側に2つの入力コンポーネントがあり、右側には出力コンポーネントが表示されています



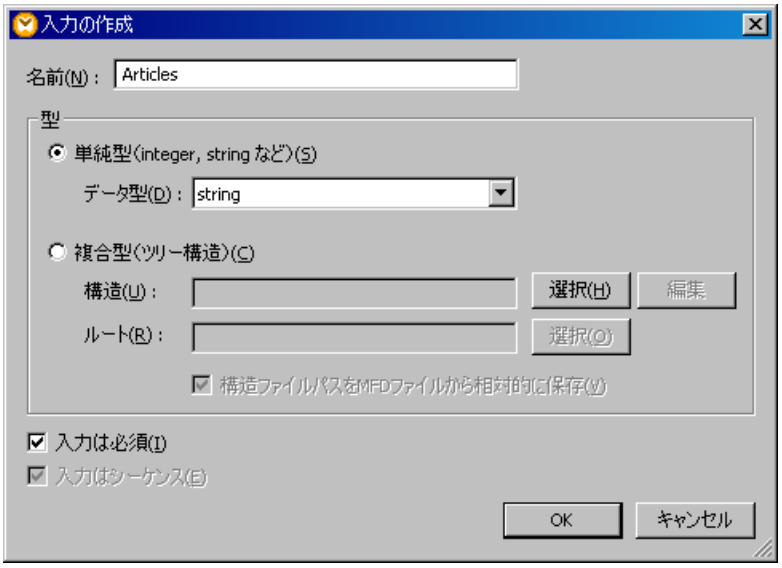
7.2.5.1 *複合型の入力コンポーネント - 定義*

XML 構造を入力パラメーターとする関数の作成方法は、以下のとおりです：

1. 通常の方法でユーザー定義関数を作成します。メニューオプションから「関数 | ユーザー定義関数の作成」を選択し、**「OK」**をクリックすることで確定します。インラインを使用チェックボックスが自動的に選択されていることに注目してください。



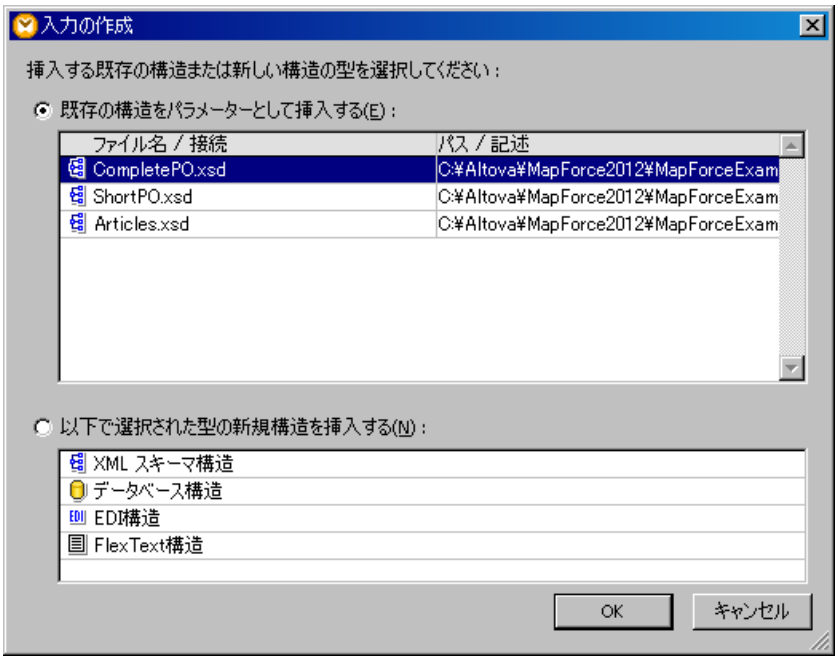
2. アイコンバーにある**入力挿入**アイコン  をクリックします。
3. 入力コンポーネントの名前を名前フィールドに入力します。



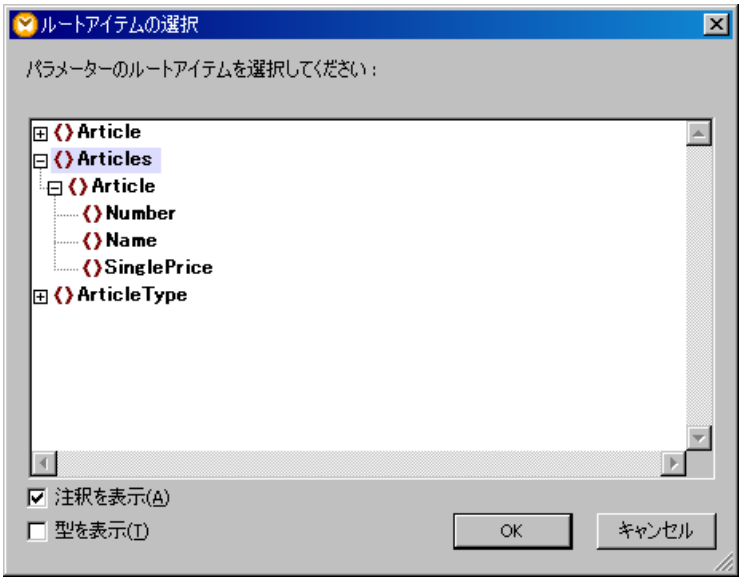
4. **複合型（ツリー構造）**ラジオボタンをクリックして、構造フィールドの隣にある**選択**ボタンをクリックします。これによりダイアログボックスが開かれます。

   上部のリストボックスには、マッピングにおける既存のコンポーネントが表示されます（サンプルファイルを開いた場合、3 つのスキーマが表示されます）。このリストには、アクティブなマッピングに挿入された全てのコンポーネント（XML スキーマ ファイル）。

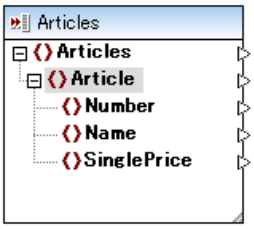
   下部のリストボックスでは、複合型のデータ構造を新たに選択することができます（例：XML スキーマ、データベース ファイル）。



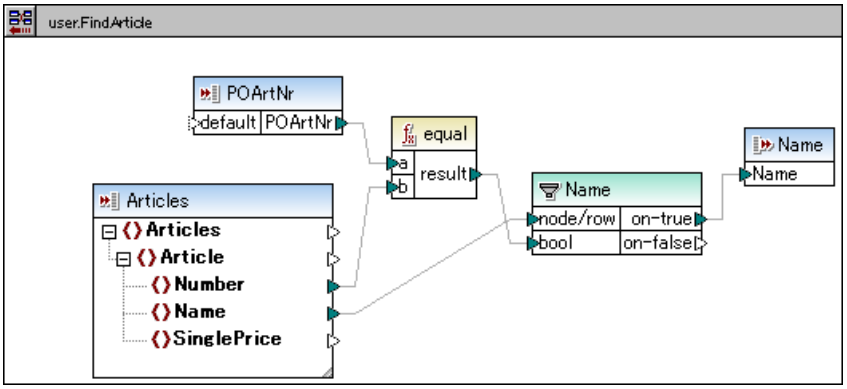
5. 「新規構造を挿入する」ラジオボタンを選択し、XML スキーマ構造エントリーを選択した後、**「OK」**ボタンをクリックします。
6. ダイアログボックスから**Articles.xsd** を選択します。
7. コンポーネントのルート要素として使用する要素（例：Articles）をクリックした後、**「OK」**をクリックします。



Articles コンポーネントがユーザー定義関数へ挿入されます。コンポーネント名の左側に入力アイコン が表示されていることに注目してください。コンポーネントが複合型入力コンポーネントとして使用されることを表しています。



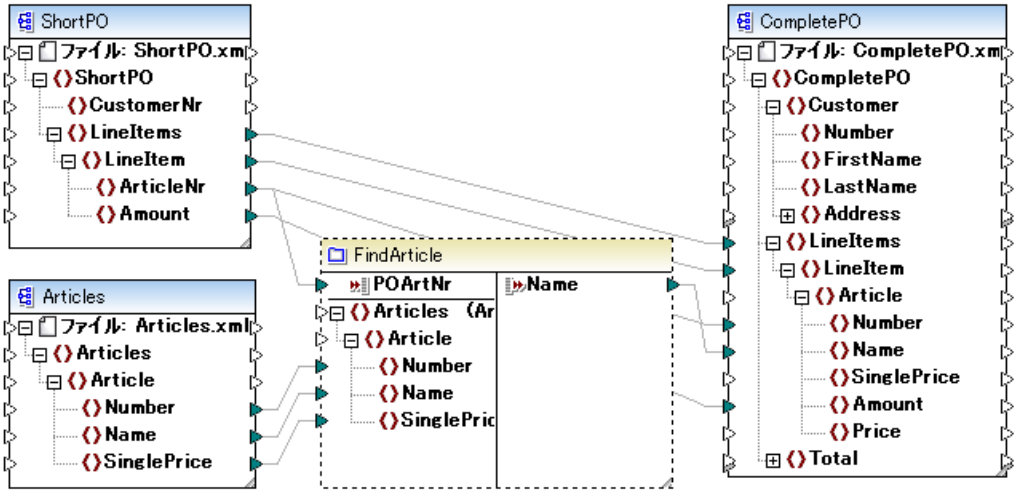
8. 以下のスクリーンショットに示されるように、その他のコンポーネントを挿入します。POArtNr という名前の 2 番目の入力コンポーネント、フィルターコンポーネント、equal 関数を挿入し、出力コンポーネントの名前を Name に変更します。以下のスクリーンショットに示されるように接続を作成します。



- **Articles** 入力コンポーネントは、ユーザー定義関数の**外から**データを受け取ります。このコンポーネントに対するマッピングは、ユーザー定義関数の外側で表示される入力アイコンから行うことができます。
- データを与えるために使用される XML **インスタンス**ファイルを、ユーザー定義関数の内部にて複合型入力コンポーネントに割り当てることはできません。
- 別の入力コンポーネント POArtNr により、ShortPO ArticleNr のデータが与えられ、**Article | Number** のデータと比較されます。
- フィルターコンポーネントにより、同一のレコードがフィルタリングされ、出力コンポーネントに渡されます。

9. ホームアイコン をクリックして、メインマッピングに戻ります。
10. 新たに作成されたユーザー定義関数をライブラリペインからマッピングへドラッグします。

11. 以下のスクリーンショットに示されるような接続を作成します。



マッピングの左側では、2つのスキーマ/XML ファイルのアイテムからマッピングされる入力パラメーターが示されます：

- **ShortPO** により、入力コンポーネントの**POArtNr** に対するデータが与えられます。
- **Articles** からは、複合型の入力コンポーネントに対するデータが与えられます。Articles スキーマファイルのコンポーネントが挿入された際にArticles.xml インスタンスファイルが割り当てられました。
- XML 子ノードを伴う複合型入力コンポーネントのe complex 入力コンポーネント**Articles** には、Articles コンポーネントからデータがマッピングされます。

マッピングの右側には、以下のアイテムが表示されます：

- **"Name"** という名前の単純型出力パラメーターでは、同じArticle 番号によりフィルタリングされたアイテムが与えられ、CompletePO のName アイテムへ渡されます

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <CompletePO xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3    <LineItems>
4      <LineItem>
5        <Article>
6          <Number>3</Number>
7          <Name>Pants</Name>
8          <Amount>5</Amount>
9        </Article>
10     </LineItem>
11     <LineItem>
12       <Article>
13         <Number>1</Number>
14         <Name>T-Shirt</Name>
15         <Amount>17</Amount>
16       </Article>
17     </LineItem>
18   </LineItems>
19 </CompletePO>
```

メモ:
スキーマとユーザー定義関数のパラメーター間にて、**全てコピー**接続を作成する場合、両方のコンポーネントが同じスキーマをベースにしている必要があります！両コンポーネントが同一のルート要素を持っている必要はありません。

7.2.6 複合型のユーザー定義関数 - XML ノードを出力に使用

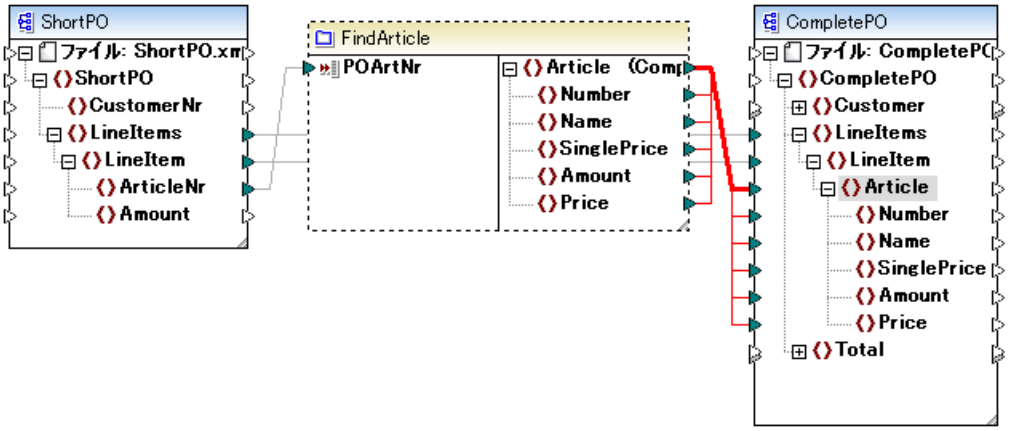
以下のサンプルは**...\ MapForceExamples** フォルダーに収められている**lookup-udf-out.mfd** ファイルにて確認することができます。このセクションでは、複合型の出力コンポーネントを持つインラインのユーザー定義関数の作成方法について説明します。
ユーザー定義関数の "FindArticle" は 2 つのパーツから構成されることに注意してください。

以下の入力パラメーターを含む左半分：

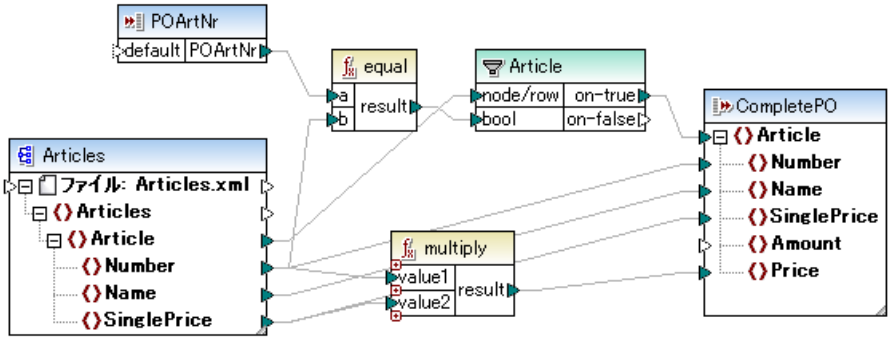
- 単純型の入力パラメーター：**POArtNr**

以下のコンポーネントを含む右半分：

- CompletePO へマッピングされた XML 子ノードを伴う複合型の出力パラメーター：**Article (CompletePO)**



ユーザー定義関数を行使絵するコンポーネントを以下のスクリーンショットに示します。左側に入力コンポーネントがあり、右側にお複合型の出力コンポーネントが表示されています。

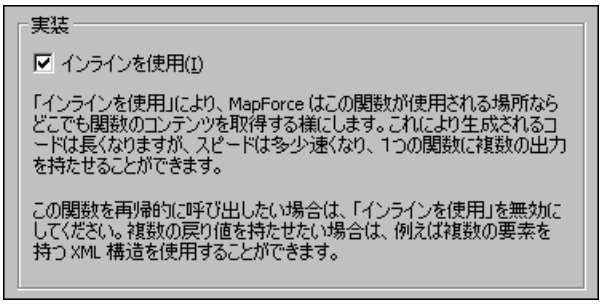


7.2.6.1 *複合型の出力コンポーネント - 定義*

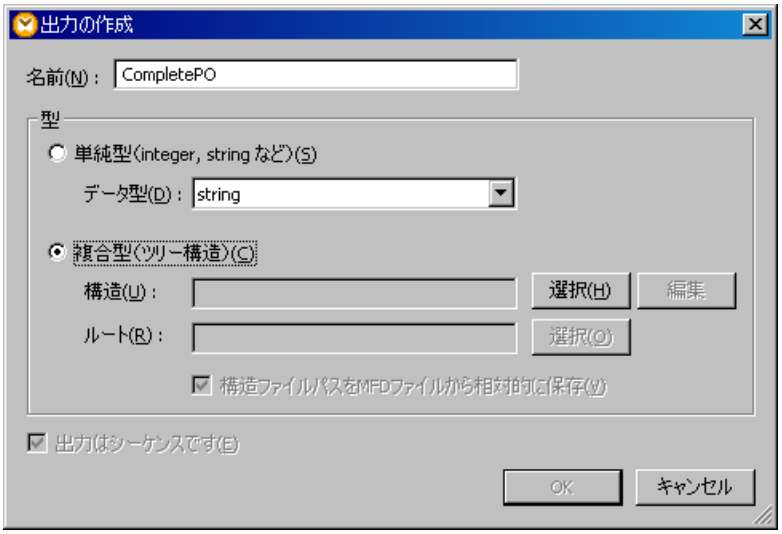
XML 構造を出力パラメーターとして返す関数の作成方法は、以下のとおりです：

1. 通常の方法でユーザー定義関数を作成します。メニューオプションから「関数 | ユーザー定義関数の作成」を選択し、FindArticle という関数名を与え、「**OK**」をクリックすることで確定します。**インライン**を使用オプションが自動

的に選択されます。



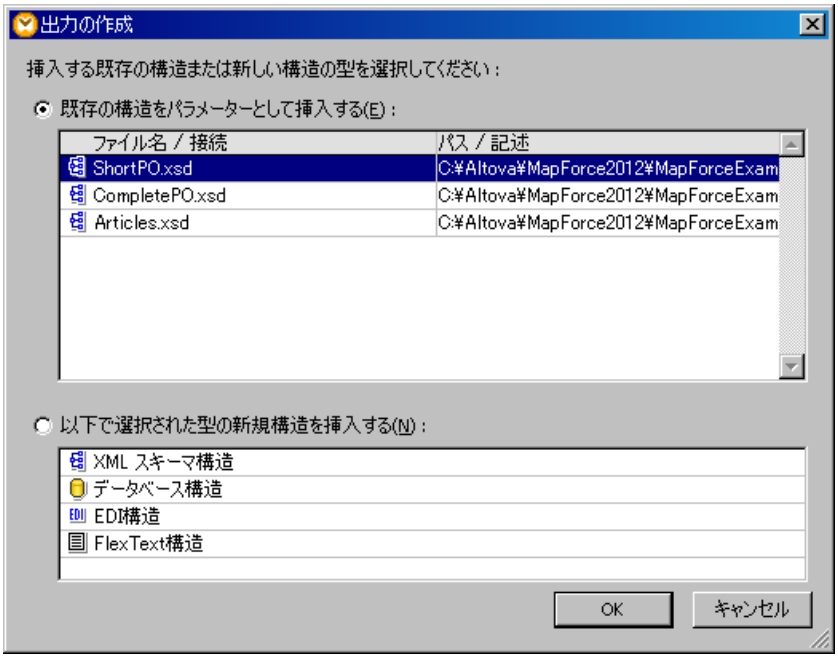
2. アイコンバーにある**出力**アイコン  をクリックして、名前を入力します (例：CompletePO)。



3. **複合型**ラジオボタンをクリックして、構造フィールドの隣にある**選択**ボタンをクリックします。
別のダイアログボックスが新たに表示されます

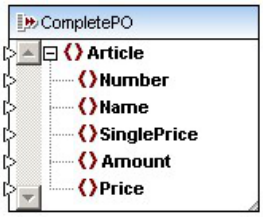
上部のリストボックスには、マッピングにおける**既存**のコンポーネントが表示されます (サンプルファイルを開いた場合、3 つのスキーマが表示されます)。このリストには、アクティブなマッピングに挿入された全てのコンポーネント (XML スキーマ ファイル) が含まれていることに注目してください。

下のリストボックスから、以下の新規の複合型データ構造を選択することができます：XML スキーマ ファイル)。

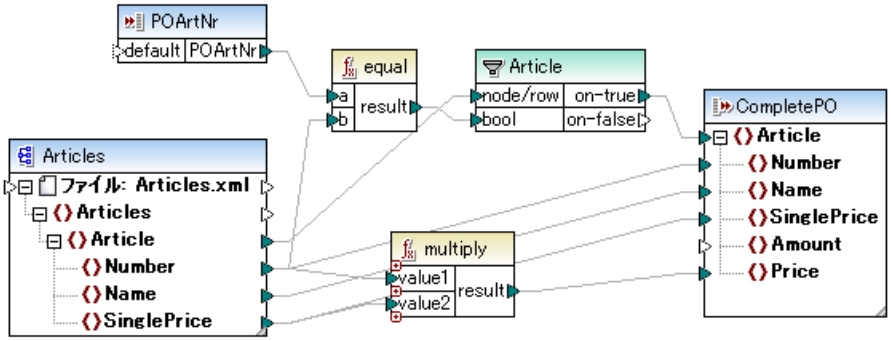


4. "**新規構造の挿入 ...**" ラジオボタンをクリックして、XML スキーマ構造のエントリを選択し、「OK」をクリックして続行します。
5. "開く"ダイアログボックスから**CompletePO.xsd** を選択します。
6. コンポーネント内でルート要素になる要素（例えば、**Article**）をクリックします。「OK」をクリックしてダイアログボックスを閉じます。

CompletePO コンポーネントはユーザー定義関数に挿入されます。出力アイコン がコンポーネント名の左横に表示されることに注意してください。これは、コンポーネントが複雑型出力コンポーネントに使用されていることを示します。



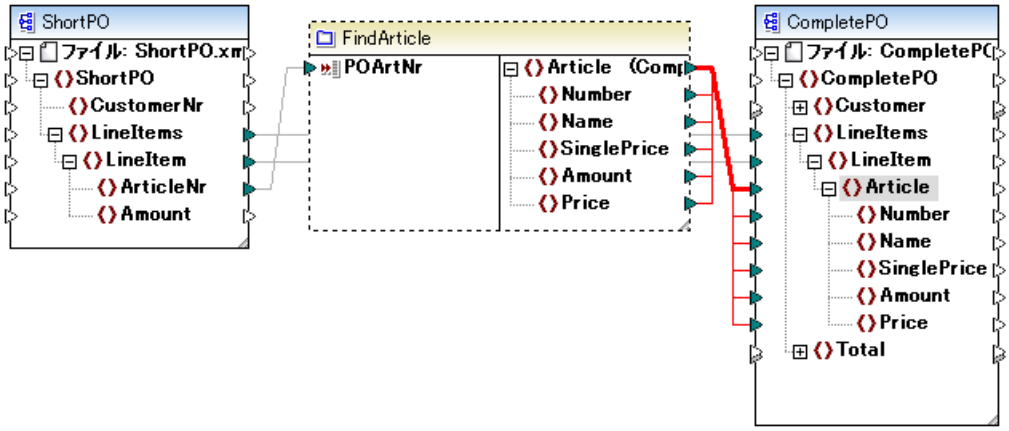
7. Article スキーマ/XML ファイルをユーザー定義関数に挿入し、**Articles.xml** をXML インスタンスとして割り当てます。
8. 残りのコンポーネントを下のスクリーンショットに表示されるように挿入します。つまり：'単純型 "入力コンポーネント (POArtNr)、フィルター、equal、とmultiply コンポーネントが以下のように接続されます。



- ユーザー定義関数内部では、**Articles** コンポーネントによりArticles.xml インスタンスファイル内にあるデータが与えられます。
- 入力コンポーネントからはPOArtNr (ArticleNr) が与えられ、Articles | Number とPrice が計算に使用されます。
- フィルターコンポーネントにより、両方の番号が等しいレコードだけがCompletePO 出力コンポーネントへ渡されます。

9. ホームアイコン をクリックして、マッピングへ戻ります。
10. ライブラリペインにて新たに作成されたユーザー定義コンポーネントをマッピングへドラッグします。

11. 以下のスクリーンショットにあるような接続を作成します。
Article (CompletePO) への接続を作成したら、接続の線を右クリックして、コンテキストメニューから「全てコピー」を選択して下さい。その他のコネクタが自動的に生成され、以下のスクリーンショットに示されるように、ハイライトされます。



メモ:
スキーマと「インライン」型のユーザー定義関数間で**全てコピー**接続を作成する場合、両方のコンポーネントが同じスキーマをベースにしている必要があります！両コンポーネントが同一のルート要素を持っている必要はありません。

関数コンポーネントの左側には入力パラメーターが表示されており、単純型のアイテムがマッピングされています。

- ShortPO から**POArtNr** 入力コンポーネントへArticleNr が渡されます。

関数コンポーネントの右側には以下のものが含まれます：

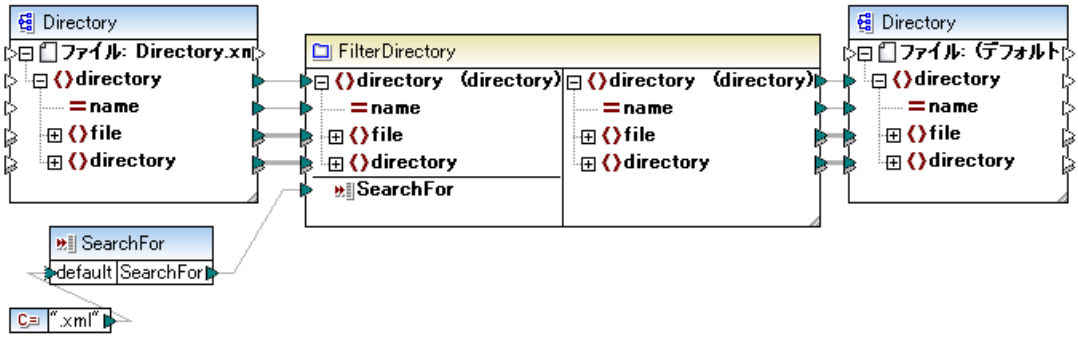
- フィルタリングされたアイテムが与えられる "**Article (CompletePO)**" という名前の複合型出力コンポーネントが、子ノードとともにCompletePO へマッピングされます。

```
<?xml version="1.0" encoding="UTF-8"?>
<CompletePO xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespace
  <LineItems>
    <LineItem>
      <Article>
        <Number>3</Number>
        <Name>Pants</Name>
        <SinglePrice>34</SinglePrice>
        <Price>102</Price>
      </Article>
    </LineItem>
    <LineItem>
      <Article>
        <Number>1</Number>
        <Name>T-Shirt</Name>
        <SinglePrice>25</SinglePrice>
        <Price>25</Price>
      </Article>
```

7.2.7 再帰的なユーザー定義マッピング

このセクションでは、**..\MapForceExamples** フォルダー内部の**RecursiveDirectoryFilter.mfd** にあるマッピングの作成方法ならびに再帰的なマッピングのデザイン方法について説明します。MapForceExamples プロジェクトフォルダーには、再帰的なマッピングを行う際に参照することのできるサンプルファイルが他にも収められています。

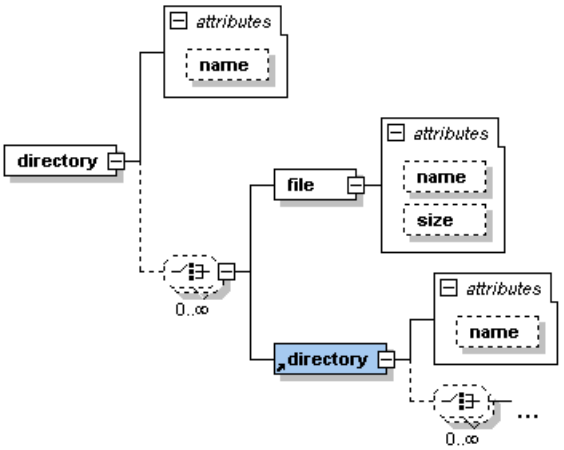
以下のスクリーンショットでは、再帰的なユーザー定義関数のFilterDirectory を含むマッピングが示されており、ソースファイル内にある xml ファイルのリストをフィルタリングします。



このマッピングに対するファイルならびにディレクトリのデータが収められている**ソースファイル**はdirectory.xml です。この XML ファイルにより、ディレクトリならびにファイルのデータが階層構造で与えられます（以下のスクリーンショットを参照）。

```
<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 name="Import">
  <file name="altova.mdb" size="266240"/>
  <file name="Data_shape.mdb" size="225280"/>
</directory>
<directory name="IndustryStandards">
  <directory name="News">
    <file name="high-tide.jpg" size="10793"/>
    <file name="Newsml-example.xml" size="5004"/>
    <file name="nitf-example.xml" size="9327"/>
  </directory>
```

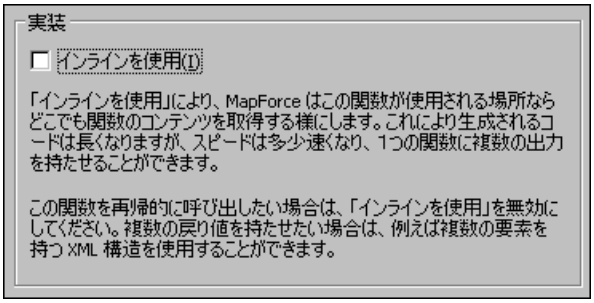
Directory.xml により参照されているXML スキーマファイルには、"directory" という名前の**再帰的**な要素が含まれており、directory 要素以下に任意の数のサブディレクトリを配置することができるようになっています。



7.2.7.1 再帰的なユーザー定義関数の定義

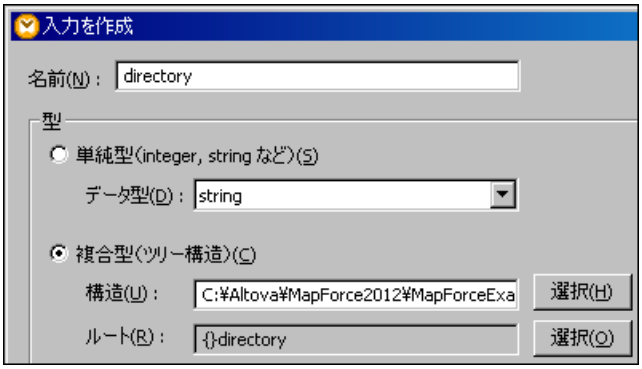
再帰的ユーザー定義関数の作成方法は、以下のとおりです：

1. メニューオプションの「関数 | ユーザー定義関数の作成」を選択して、作成する関数の名前（例えば FilterDirectory）を入力します。
2. 実装グループにある**インラインを使用**チェックボックスのチェックを外して、**「OK」**をクリックしてください。これで関数を再帰的にデザインすることができるようになります。

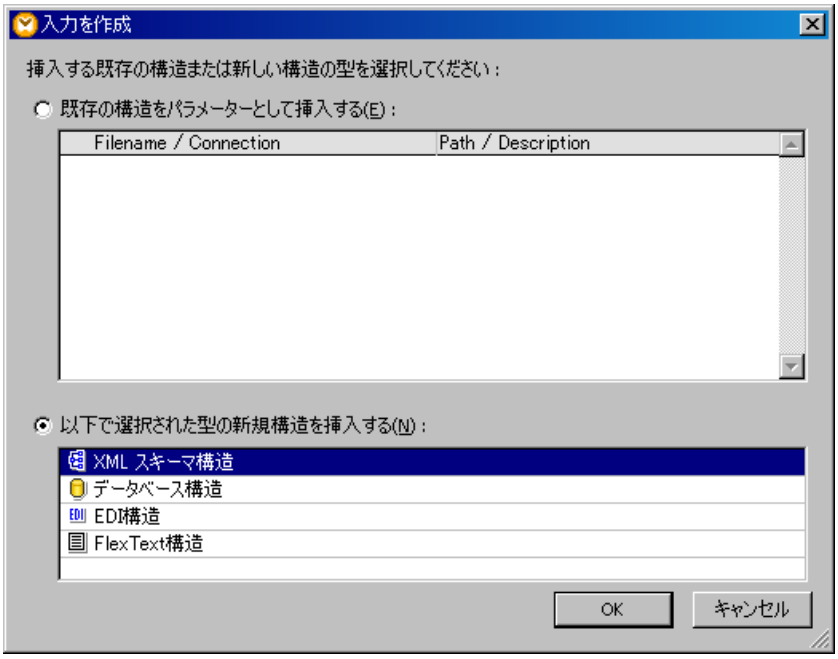


これで**FilterDirectory** ウィンドウが表示され、ユーザー定義関数の作成を始めることができます。

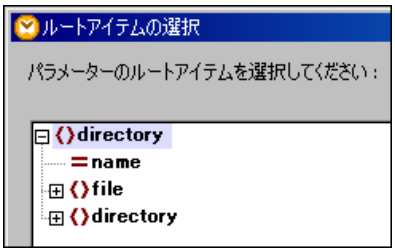
3. メニューから「関数 | 入力の挿入」を選択して、入力コンポーネントを挿入します。
4. 入力コンポーネントの名前（例えばdirectory）を入力し、**複合型**（ツリー構造）ラジオボタンをクリックします。



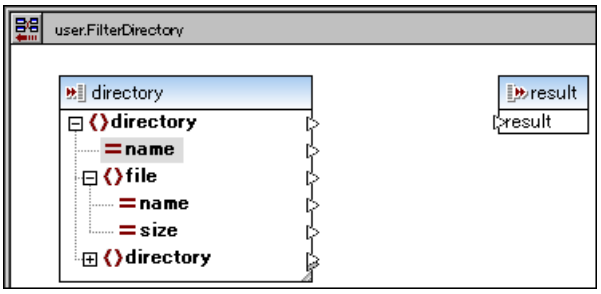
5. 選択ボタンをクリックして、表示される入力を作成ダイアログボックスにてXML スキーマ構造をクリックし、**「OK」**をクリックします。



6. ダイアログボックスにて .\MapForceExamples フォルダーの**Directory.xsd** ファイルを選択します。
7. ルートアイテムを選択するダイアログにて **「OK」** をクリックして、以下に示される様にdirectory を選択します。



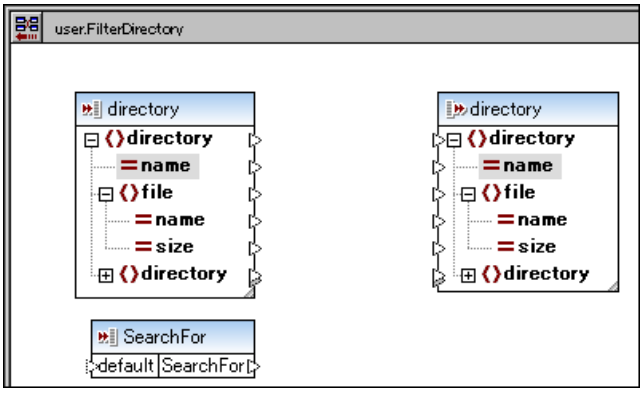
8. **「OK」** クリックして、複合型の入力パラメーターを挿入します。
ユーザー定義関数が表示されます。



9. 複合型の出力コンポーネントが必要であるため、単純型の出力コンポーネントを削除します。
10. 「関数 | 出力の挿入」を選択し、上記にある方法にて出力コンポーネントを挿入します。複合型の出力コンポーネントである "directory" を作成します。

入力側に 1つ、出力側に 1つの複合型コンポーネントが作成されました

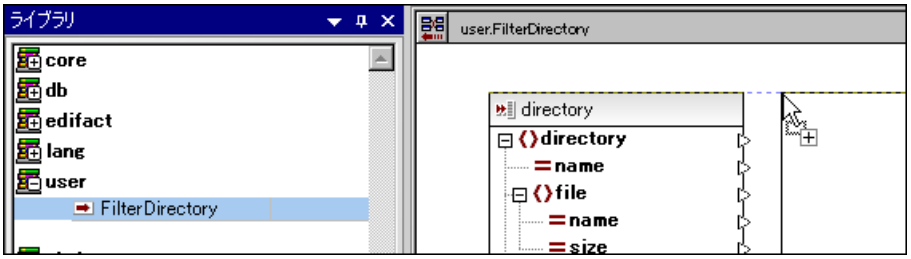
11. 「関数 | 入力の挿入」を選択して、単純型のコンポーネントである**SearchFor** を挿入します



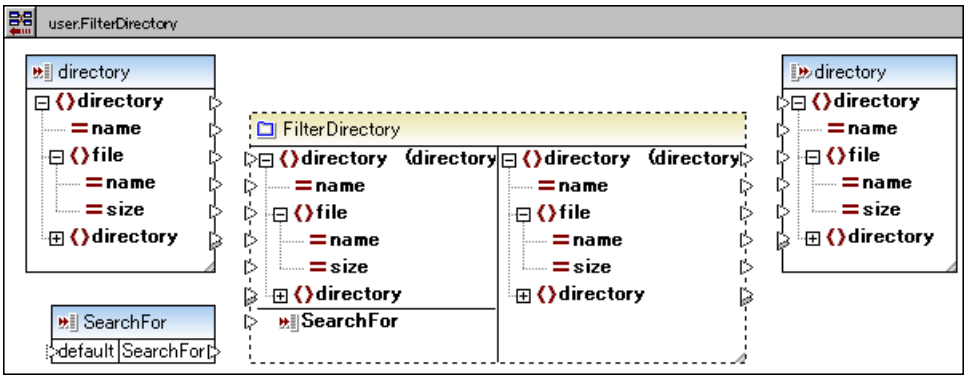
再帰的なユーザー定義関数を挿入する

この時点で必要な入力ならびに出力コンポーネントが全て定義されました。次に「未完成の」関数を、現在のユーザー定義関数ウィンドウに挿入する必要があります（この操作は始どいつでも行うことができます）。

1. **ライブラリ**ウィンドウの**user** セクションにはFilterDirectory 関数があります。
2. FilterDirectory をクリックして、これまで作業してきたFilterDirectory ウィンドウへ関数をドラッグします。

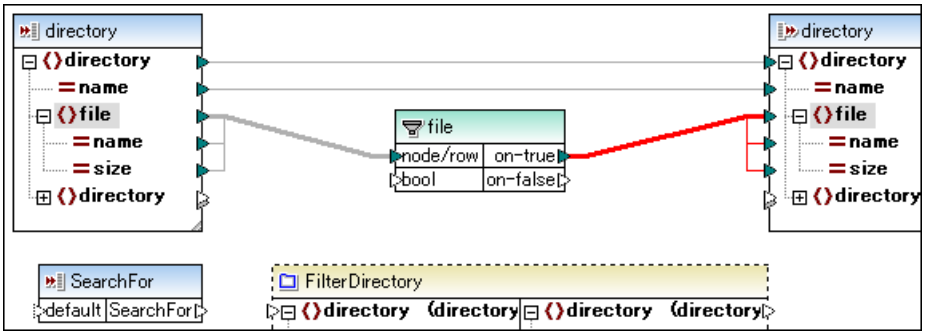


ユーザー定義関数が、再帰的なコンポーネントとしてユーザー定義関数ウィンドウに表示されます。

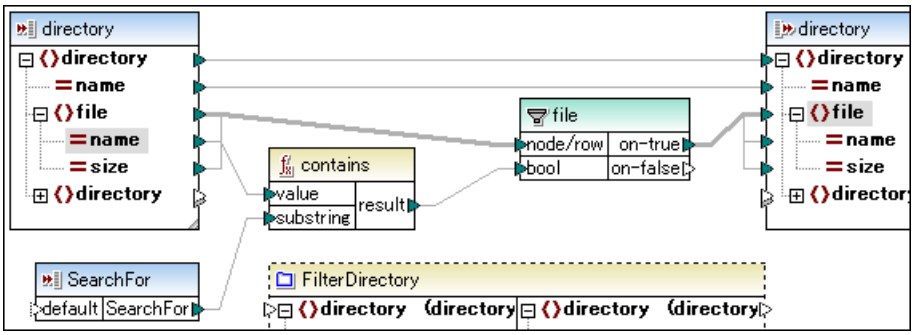


3. 入力コンポーネントの**directory**、**name** そして**ファイル**アイテムを、出力コンポーネントにある同名のアイテムへ接続します。
4. **ファイル**アイテム間で接続されているコネクタを右クリックして、ノード行フィルターを挿入を選択します。
5. on-true コネクタを右クリックして、コンテキストメニューから**全てコピー**（子要素）を選択します。

コネクタが全てコピーに変更されます。



6. **core** ライブラリの**string functions** 以下からcontains 関数を挿入します。
7. **name** から**値** への接続を行い、**SearchFor** パラメーターから**substring** へ接続した後、result をフィルターのbool アイテムへ接続します。



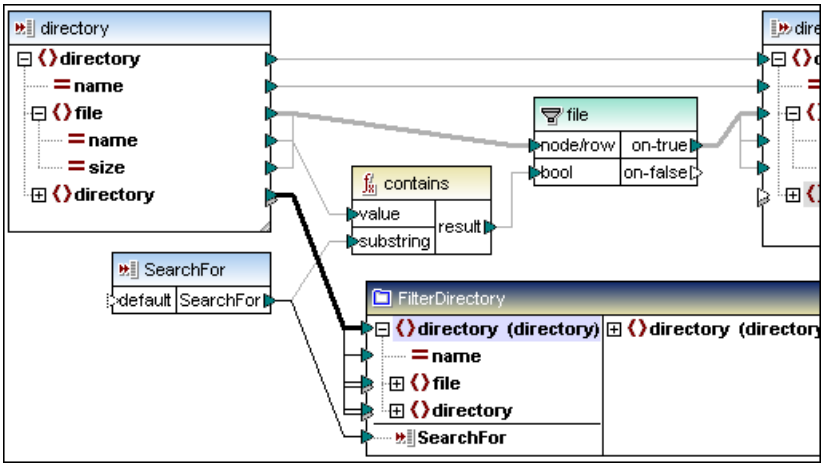
8. 入力コンポーネントのSearchFor アイテムから、ユーザー定義関数のSearchFor アイテムへ接続します。

再帰の定義を行う

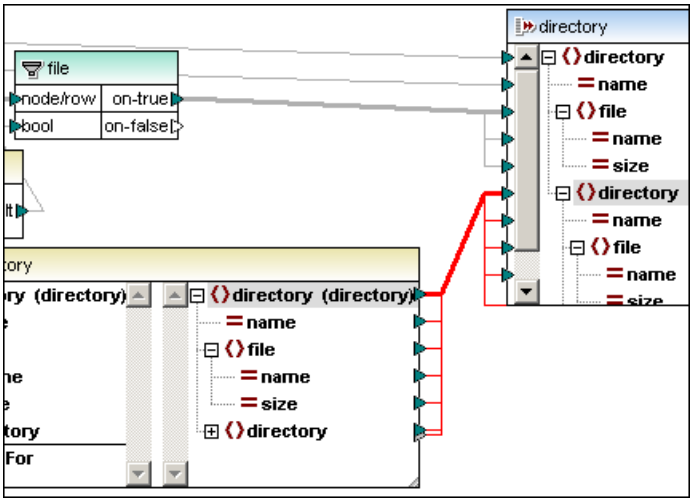
この段階で、与えられた 1 つのディレクトリ階層に対するマッピングが作成されました。次にサブディレクトリに対する処理を定義します。

アイコンバーの自動接続アイコン が有効になっていることを確認してください。

1. 入力コンポーネント下部にあるdirectory アイテムを、再帰的なユーザー定義関数の上部にあるdirectory アイテムへ接続します。



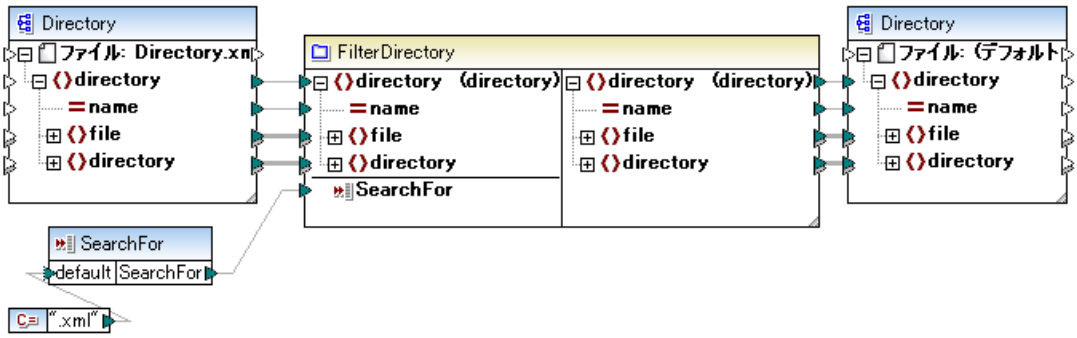
2. ユーザー定義関数の上部にあるdirectory アイテムを、出力コンポーネント以下にあるdirectory アイテムへ接続します。
3. 全てコピー接続を作成するか尋ねられた場合は、コネクタを右クリックして、コンテキストメニューから全てコピーを選択しをクリックします。



これでユーザー定義関数の定義を完了することができます。
メインマッピングウィンドウへ戻るアイコン をクリックして、マッピングの定義を継続します。

メインマッピングウィンドウ

1. ライブラリウィンドウの**user** セクションからFilterDirectory 関数を**メイン**マッピングエリアへドラッグします。
2. メニューオプションの 挿入 | XML スキーマファイル」からDirectory.xsd を挿入し、インスタンスファイルとして Directory.xml を選択します。
3. 同様の方法でDirectory.xsd を選択し、ルート要素の選択をスキップして出力コンポーネントを挿入します。
4. 定数コンポーネントを挿入し、SearchFor コンポーネントという名前の入力コンポーネントを挿入します。
5. 以下のスクリーンショットに有るような接続を作成します。
6. FilterDirectory ユーザー定義関数の入力ならびに出力側に接続されているdirectory アイテムのコネクタを右クリックして、コンテキストメニューから全てコピーを選択します。



7. 出力ボタンをクリックして、マッピングの結果を確認します。

メモ:
Directoryコンポーネントの最下部にある "directory" アイテムをダブルクリックすることで、1つ下のレベルにある再帰構造 (directory | file | directory) を確認することができます。全てコピー接続を使用することで、XML インスタンスファイル内にある全ての再帰レベルが使用されるようになり、手動で再帰レベルを展開する必要がなくなります

7.3 カスタム XSLT 1.0 または 2.0関数のインポート

XSLT 1.0 と2.0関数ライブラリをMapForce 内で使用することのできる単純型を返すカスタム関数として拡張することができます。

簡単なデータ型 (例えば、文字列) を返すカスタム関数のみがサポートされます。

XSLT ファイルから関数をインポートする:

1. **ツール** メニューから、**オプション**をクリックします。(または、ライブラリウィンドウの下のエリア内の**ライブラリの追加 削除**をクリックします。)
2. **ライブラリ**の横の**追加**をクリックし、.xsl または .xslt ファイルを参照します。

インポートされたXSLT ファイルは、ライブラリウィンドウ内のライブラリに表示され、全ての名前を持つテンプレートを関数として、ライブラリ名の下に表示されます。インポートされたライブラリが表示されない場合、XSLT が変換言語として設定されているかを確認してください (次を参照 :変換言語の選択)。

以下の点に注意してください::

- MapForce にインポートするには、関数は、XSLT ファイル内のXSLT 仕様に準拠する名前を持つテンプレートとして宣言されている必要があります。XSLT 2.0 ドキュメント内で発生する関数を`<xsl:function name="MyFunction">` の書式でインポートすることもできます。インポートされたXSLT ファイル内で、他のXSLT ファイルをインポートまたは含むことができ、これらのXSLT ファイルと関数もインポートされることができます。
- インポートされたカスタム関数のマップすることのできる入力コネクタはテンプレート呼び出し内で使用されるパラメーターの数により異なります。任意のパラメーターもサポートされています。
- 名前空間はサポートされています。
- MapForce に既にインポートされているXSLT ファイルの更新を行う場合、変更は自動的に検知され、MapForce がファイルの再ロードを促します。
- 名前付きのテンプレートに書き込む場合、テンプレート内で使用されているXPath ステートメントが正しい名前空間を持つことを確認してください。マッピングの名前空間にバインドする名前空間を確認するには、生成されたXSLT コードをプレビューしてください。.

XPath 2.0 内のデータ型

XML ドキュメントからXML スキーマが参照されており、スキーマに対してドキュメントが妥当である場合、演算にて目的のデータ型へ暗示的に変換されないデータ型を明示的に構築またはキャストする必要があります。

Altova XSLT 2.0 エンジンで使用されるXPath 2.0 データモデルでは、XML ドキュメントから**原子化**された全てのノード値にxs:untypedAtomic データ型が割り当てられます。xs:untypedAtomic 型は、以下にあるように暗示的な型変換に使用されます:

例えば、

- xs:untypedAtomic("1") +1 では、xdt:untypedAtomic 値が加算演算子によりxs:double に**暗黙的に**変換され、式の結果は 2 となります。.
- 四則演算では、オペランドが暗黙的にxs:double へ変換されます。
- 値の比較を行う場合、比較の前にオペランドがxs:string へ変換されます。

以下も参照してください:

例 :カスタムXSLT 1.0 関数の追加

XSLT 1.0 エンジンの実装
XSLT 2.0 エンジンの実装

7.3.1 例 :カスタム XSLT 1.0 関数の追加

この例は、カスタムXSLT 1.0 関数をMapForce にインポートする方法を説明しています。以下に示される単純なサンプルに必要なファイルは **...\ MapForceExamples** ディレクトリに収められています。

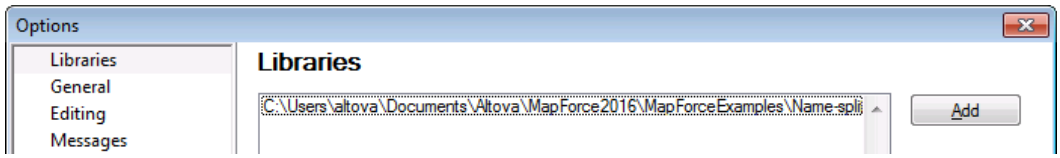
- 目的の変換を行い、その結果を出力するXSLT ファイルを作成します。
  以下に示される**Name-splitter.xslt** サンプルでは、"string" のパラメーターを伴う "**tokenize**" という名前付きテンプレートを確認することができます。テンプレートにより、空白スペースの後に来る文字が大文字になります。

```
2  <xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
3    <xsl:output method="xml" version="1.0" encoding="UTF-8" indent="yes"/>
4
5    <xsl:template match="*">
6      <xsl:for-each select=".">
7      <xsl:call-template name="tokenize">
8      <xsl:with-param name="string" select="."/>
9      </xsl:call-template>
10     </xsl:for-each>
11   </xsl:template>
12
13   <xsl:template name="tokenize">
14     <xsl:param name="string" select="."/>
15     <xsl:variable name="caps" select="translate($string, '-abcdefghijklmnopqrstuv
16     <xsl:variable name="capscount" select="string-length($caps)"/>
17     <xsl:variable name="token">
```

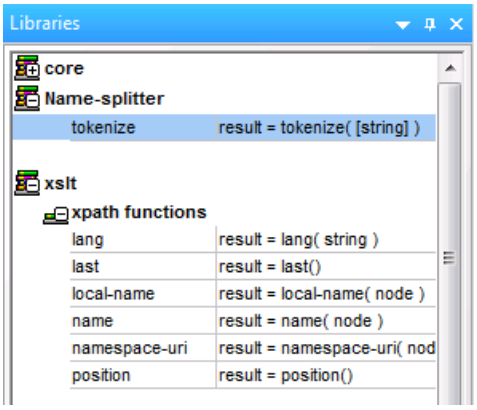
- **Name-splitter.xml** (処理されるソースXML インスタンスファイル)
- **Customers.xsd** (ソースXML スキーマ)
- **CompletePO.xsd** (ターゲットXML スキーマ)

カスタムXSLT 関数の追加 :

1. XSLT を変換言語として選択します。 (次を参照してください:変換言語の選択)
2. ライブラリウィンドウの下のエリア内の**ライブラリを追加 /削除** ボタンをクリックします。または、**ツール**メニューから**オプション**をクリックして、**ライブラリ**を選択します。
3. **追加** をクリックして、関数として使用する名前付きテンプレートを含むXSL またはXSLT ファイル (この例では **Name- splitter.xslt**) を選択します。

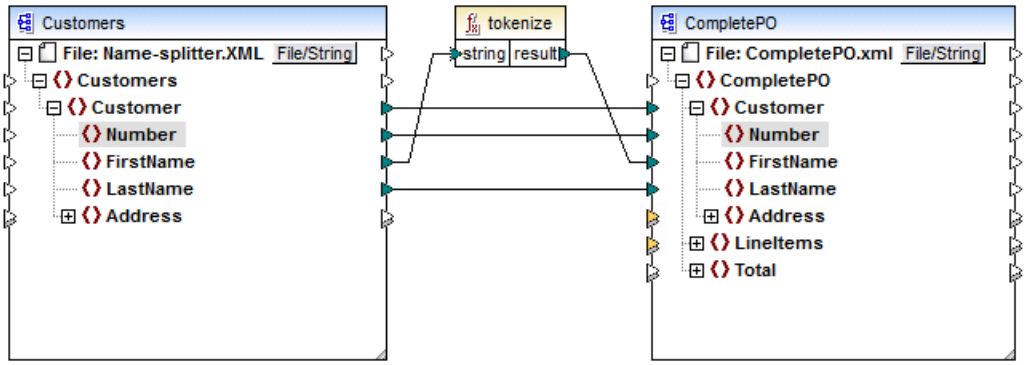


4. 「**OK**」をクリックします。名前付きテンプレートとして定義された関数とともにXSLT ファイル名がライブラリウィンドウに表示されます。この例では、**Name- splitter** というライブラリ以下に`tokenize` という関数が表示されます。



マッピング内でXSLT 関数を使用する:

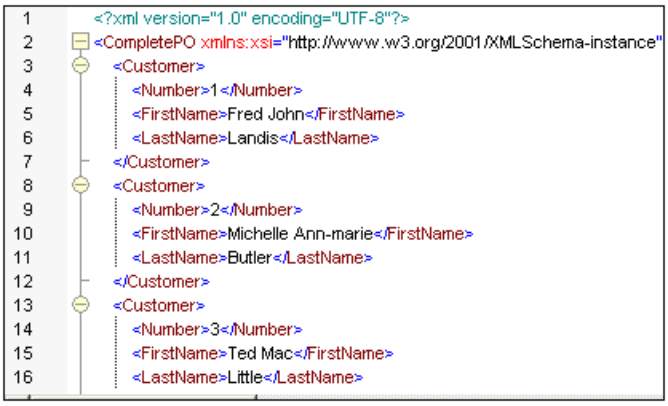
1. マッピングウィンドウへ`tokenize` 関数をドラッグして、以下のスクリーンショットに表示されているようなマッピングを作成します。



2. XSLT タブをクリックして、XSLT コードを生成します。

メモ: 名前付きテンプレートがマッピングで使用されると、名前付きテンプレートが含まれているXSLT ファイルが、生成されたXSLT コードに **xsl:include href...** により**インクルード**され、**xsl:call- template** コマンドを使用することで呼び出されます。

3. 出力タブをクリックすることで、マッピングの結果を確認できます。



XSLT ライブラリを MapForce から削除する:

1. **ライブラリを追加/削除**ボタンをクリックします。
2. ライブラリタブにて、目的のXSLT **ライブラリ名**をクリックし、削除ボタンをクリックした後

7.4 カスタム XQuery 1.0 関数をインポートする

XQuery がマッピングの変換言語として選択されている場合、MapForce は、XQuery のために使用することのできる関数ライブラリをライブラリウィンドウ内に表示します。必要であれば、このリストをXQuery 関数を使用して、カスタム XQuery 1.0 ライブラリモジュールを MapForce にインポートして拡張することができます

MapForce にインポートするためには、XQuery ファイルは以下の条件を満たす必要があります :

- XQuery 仕様に従う有効なライブラリモジュールである必要があります。すなわち、`module namespace <prefix>="<namespace name"` などのモジュール宣言と共に開始する必要があります。i
- インポートされたライブラリモジュール内で宣言された全ての関数は、動的なデータ型を返す必要があります (例えば、`xs:string`, `xs:boolean`, `xs:integer` など)。関数パラメーターも動的な型を有する必要があります。

XQuery ライブラリモジュールをインポートする :

1. 「**ツール**」メニューから「**オプション**」をクリックします (または、ライブラリウィンドウの下のエリアの「**ライブラリの追加 / 削除**」をクリックします)
2. 「**ライブラリ**」の横から、「**追加**」をクリックして、xq または xquery ライブラリファイルを参照します。

インポートされた ライブラリモジュールがサポートされていない場合、メッセージボックスがプロンプトします。それ以外の場合、インポートされた ライブラリモジュールは、ライブラリウィンドウに表示されます。特定の関数をマッピングエリアにドラッグして、他の MapForce 関数コンポーネントと同様に使用します。

インポートされたXQuery がライブラリモジュールに表示されない場合、XQuery が変換言語として選択されていることを確認してください (次を参照してください 変換言語の選択)。

以下も参照 :

XQuery エンジンの実装

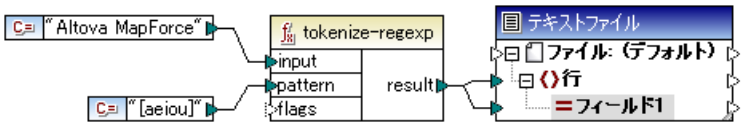
7.5 正規表現

MapForce では **tokenize-regexp** 関数 **pattern** パラメーターにて正規表現を使用して、入力パラメーターから得られた文字列の高度な絞り込みを行うことができます。

正規表現の構文ならびにセマンティクスは XSLT や XQuery で使用されているものと同一で、http://www.w3.org/TR/xmlschema-2/ にて定義されています。プログラミング言語によっては、正規表現の構文が若干違ってくることに注意してください。

用語

input 正規表現が適用される文字列
pattern 正規表現
flags オプションのパラメーターで、正規表現の解釈方法を定義する
result 関数から得られる結果



tokenize-regexp 関数により文字列のシーケンスが得られます。行アイテムへの接続により、シーケンスのアイテムごとに新たな行が作成されます。

正規表現構文

リテラル (単一の文字)
例 :"a" という文字は最も基本的な正規表現となります。この正規表現により、文字列内にある最初の "a" という文字がマッチします。

文字クラス [ ]
角括弧に含まれる全ての文字セットがマッチの対象となります。

角括弧に含まれる文字の**1 つ だけ**が文字列に対してマッチします。

pattern **[aeiou]**
小文字の母音アルファベット全てがマッチの対象となります。

pattern **[mj]ust**
"must" または "just" という単語がマッチします。

"pattern" に含まれる値は大文字と小文字で区別されるという点に注意してください。小文字の "a" は大文字の "A" にマッチしません。

文字範囲 **[a-z]**
2 つの文字間にある文字全てがマッチの対象となります。一度にマッチする文字は一文字だけである点に注意してください。

pattern **[a-z]**
a から z まで、全ての小文字アルファベットがマッチの対象となります。

否定クラス **[^]**
角括弧の開始直後にハット記号を使用することで、その文字クラスを否定したことになります。

pattern **[^a-z]**
文字クラスに含まれていない（つまりこの例では小文字以外）全ての文字にマッチします。

メタ文字 "."

ドットメタ文字
（改行を除く）**全ての文字**にマッチします。

pattern **.**
全ての単一文字にマッチします。

量指定子 ? + * {}

量指定子を使用することで、与えられた文字列にマッチする繰り返しを定義することができます。

?
0 または 1　　その前の文字列（チャンク）がオプションとなります。

+
1回以上　　その前に表示される文字が1回以上マッチします。

*****
0回以上　　その前に表示される文字が0回以上マッチします。

{}
最小（/最大）繰り返し　　与えられた回数の繰り返しにマッチします。

例 :mo{1,3} により、"mo"、"moo"、"mooo" がマッチします。

()
サブパターン
正規表現の要素をグループ化するために使用されます。

|
代替（/または）左から順にサブパターンのマッチを行います。
例：**(horse|make) sense** により、"horse sense" または "make sense" がマッチします。

フラグ

以下に示されるのはオプションのパラメーターで、正規表現の解釈方法を指定するために使用されます。個々の文字によりオプションを指定することができます。文字はどの順序でも使用することができ、繰り返して使用することもできます。

s
このオプションが指定された場合、ドットの適用範囲が拡張されます。

メタ文字の "."が全ての文字にマッチするようになります。**s** フラグがセットされていれば、入力文字列に "hello" と "world" が 2 つの異なる行にある場合でも、"hello.*world" という正規表現によりマッチが行われます。

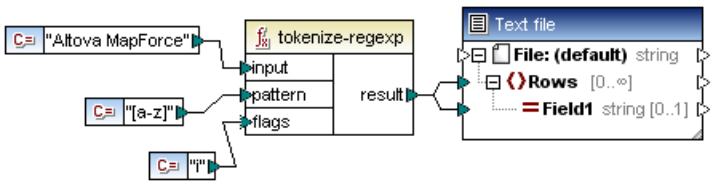
m
このオプションが指定された場合、複数行モードによるマッチが行われます。

複数行ノードでは、ハット記号 ^ が**全て**の行頭（文字列全体の開始位置と改行文字の直後に来る文字）に対してマッチするようになります。

更にドル記号 **$** が**全て**の行末（文字列全体の終了位置と改行文字の直前に来る文字）に対してマッチするようになります。

改行に使用される文字は #x0A となります

i
このオプションが指定された場合、マッチングにて大文字と小文字の違いが無視されます。
i フラグを指定すると、[a-z] によりa-z の文字とA-Z の文字がマッチするようになります。



x
このオプションが指定されている場合、マッチング処理が行われる前に正規表現文字列内にある空白スペースが削除されます。空白スペースの文字は #x09、#x0A、#x0D、そして #x20 になります。

例外：
文字クラス表現内の空白文字（例：[#x20]）は削除されません。

メモ：
コードを生成する場合、言語によっては正規表現構文の正確な機能や振る舞いが異なる可能性もあります。ご利用になる言語の正規表現がどのように動作するかを確認するようにしてください。

7.6 関数ライブラリレファレンス

このレファレンスチャプターでは、ライブラリ別に整理された『ライブラリ』ペイン内で使用することのできるMapForce 内蔵の関数に関して説明されています。

『ライブラリ』ペイン内の関数ライブラリは、選択された変換言語により異なります (変換言語の選択 を参照してください)。

XPath 2.0 制約: シーケンスに関連するXPath 2.0 関数の一部は現在使用することができません。

7.6.1 core | QName functions

QName 関数は、XML ドキュメント内の修飾名 (QName) を操作する方法を提供します。

7.6.2 core | aggregate functions

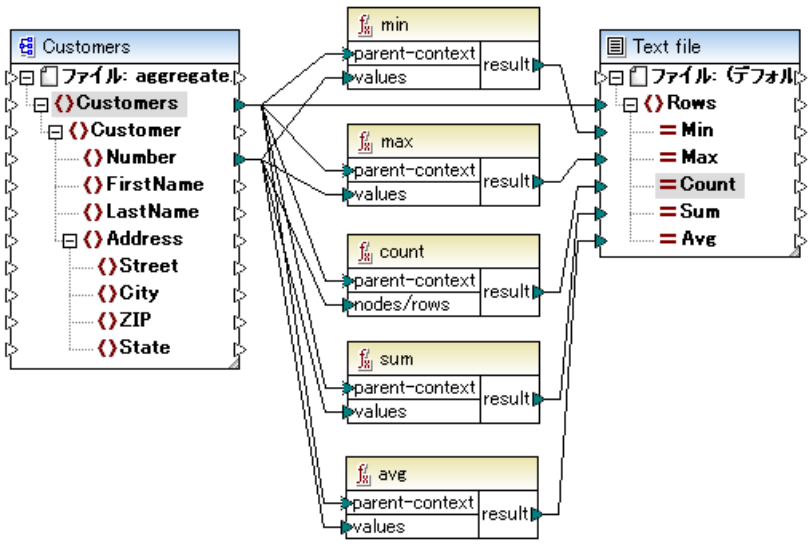
aggregate 関数は入力値のシーケンスやセットに対して使用されます。min、max、sum、そして avg の入力データが **decimal** データ型に変換され、処理されます。

- 入力値は関数の**values** パラメーターへ接続する必要があります。
- コンテキストノード (アイテム) を**parent-context** パラメーターへ接続して、入力シーケンスからデフォルトのコンテキストをオーバーライドすることができます。parent-context パラメーターはオプションになります！
- 関数の**result** を目的のターゲットアイテムへ接続します。

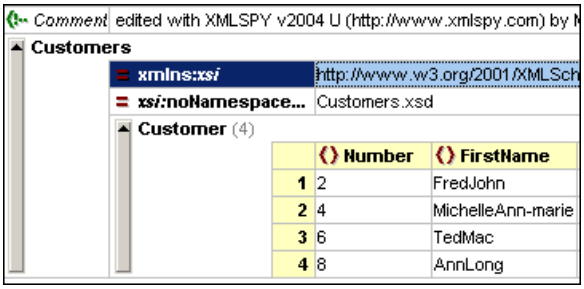
以下に示されるマッピングは、.\Tutorial フォルダーに収められている**Aggregates.mfd** から確認することができます。

集計関数には 2 つの入力アイテムが備わっています。

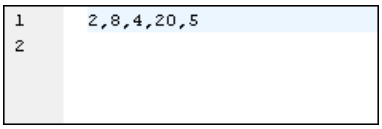
- **values** (node/row) にはデータを取得するためのソースアイテムが接続されます (この場合はNumber)。
- **parent-context** にはコンテキストとなるアイテムが接続されます。このパラメーターはオプションです。



ここでの入力インスタンスはXML ファイルで、以下のデータが含まれています：



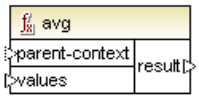
- ソースデータにある値は 2、4、6、8 という数値のシーケンスになります。
- ここでの出力コンポーネントはテキストファイルとなります。
  出力ボタンをクリックすることで以下の結果を得ることができます：



min=2、max=8、count=4、sum=20 とavg=5。

7.6.2.1 avg

入力シーケンスから得られた値の平均値を返します。入力が空セットの場合、結果は空セットになります。XSLT1 では利用できません。

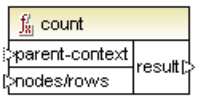


引数	説明
parent-context	オプションの引数。 親コンテキストを提供します。マッピングコンテキストのオーバーライド参照。
values	この引数は、実際のデータを提供するソースアイテムに接続されている必要があります。 与えられる引数の値は数値である必要があります。

使用方法の例に関しては、**<Documents>\Altova\MapForce2017\MapForceExamples** ディレクトリ内のマッピング **GroupTemperaturesByYear.mfd** を参照してください。

7.6.2.2 count

入力シーケンスから得られた値の数を返します。入力が空セットの場合、出力は 0 になります。XSLT1 では機能が制限されます。

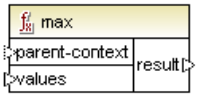


引数	説明
parent-	オプションの引数。 親コンテキストを提供します。マッピングコンテキストのオ

引数	説明
context	オーバーライド参照。
nodes/ rows	この引数は数えられるソースアイテムに接続されている必要があります。

7.6.2.3 max

入力シーケンスから得られた値の最大値を返します。入力が空セットの場合、結果は空セットになります。XSLT1 では利用できません。

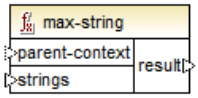


引数	説明
parent-context	オプションの引数。 親コンテキストを提供します。マッピングコンテキストのオーバーライド参照。
values	この引数は、実際のデータを提供するソースアイテムに接続されている必要があります。与えられる引数の値は数値である必要があります。文字列のシーケンスから最大値を取得するためには、`max-string` 関数を使用します。

使用方法の例に関しては、**<Documents>\Altova\MapForce2017\MapForceExamples** ディレクトリ内のマッピング **GroupTemperaturesByYear.mfd** を参照してください。

7.6.2.4 max-string

入力シーケンス内の全ての文字列の値の最大の値を返します。例 :`max-string("a","b","c")` は、 "c" を返します。この関数は、XSLT1 では利用できません。

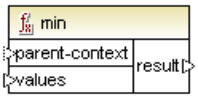


引数	説明
parent-context	オプションの引数。 親コンテキストを提供します。マッピングコンテキストのオーバーライドド参照。
strings	この引数は、実際のデータを提供するソースアイテムに接続されている必要があります。与えられた引数の値は `xs:string` のシーケンス(ゼロまたは複数)である必要があります。

strings 引数が空のセットの場合、関数は空のセットを返すことに注意してください。

7.6.2.5 min

入力シーケンスから得られた値の最小値を返します。入力が空セットの場合、結果は空セットになります。XSLT1 では利用できません。

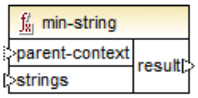


引数	説明
parent-context	オプションの引数。 親コンテキストを提供します。マッピングコンテキストのオーバーライド参照。
values	この引数は、実際のデータを与えるソースアイテムに接続されている必要があります。与えられる引数の値は数値である必要があります。文字列のシーケンスから最小値を取得するためには、`min-string` 関数を使用します。

使用方法の例に関しては、**<Documents>\Altova\MapForce2017\MapForceExamples** ディレクトリ内のマッピング **GroupTemperaturesByYear.mfd** を参照してください。

7.6.2.6 min-string

入力シーケンス内の全ての文字列の値の最小の値を返します。例 :`min-string("a", "b", "c")` は、 "a" を返します。この関数は、XSLT1 では利用できません。

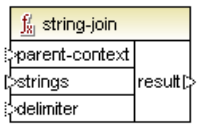


引数	説明
parent-context	オプションの引数。親コンテキストを提供します。マッピングコンテキストのオーバーライド参照。
strings	この引数は、実際のデータを与えるソースアイテムに接続されている必要があります。与えられた引数の値は `xs:string` のシーケンス(ゼロまたは複数)である必要があります。

文字列の引数が空のセットの場合、関数は空のセットを返すことに注意してください。

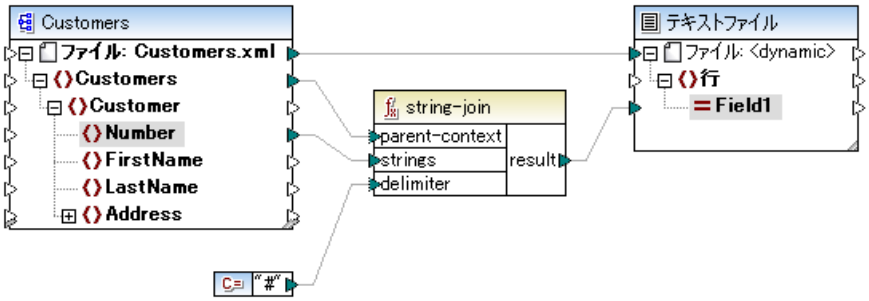
7.6.2.7 string-join

delimiter にて与えられた文字を区切り文字として、入力シーケンスから与えられた値を連結します。入力が空セットの場合、結果は空セットになります。XSLT1 では利用できません。



以下に示されるサンプルでは、Number に4つの値 (2、4、6、8)が与えられています。定数コンポーネントからは、ハッシュを表す "#"がdelimiter として与えられています。

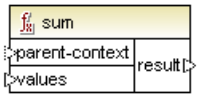
結果 = 2#4#6#8



delimiter (区切り記号)が与えられていない場合、デフォルトの区切り文字である空文字が使用され、区切り文字が表示されることはありません (上の例の場合、結果は 2468 となります)。

7.6.2.8 sum

入力シーケンスから得られた値の合計値を返します。入力セットが空の場合、結果は 0になります。XSLT1 では利用できません。



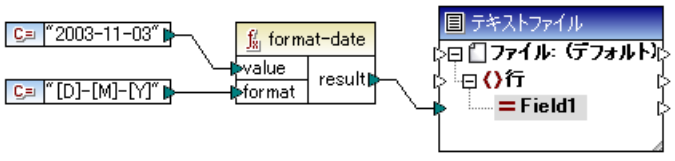
引数	説明
parent-context	オプションの引数。親コンテキストを提供します。マッピングコンテキストのオーバーライド参照。
values	この引数は、実際のデータを与えるソースアイテムに接続されている必要があります。与えられる引数の値は数値である必要があります。

以下を参照 :例 : ノードの値の加算。

7.6.3 core | conversion functions

明示的なデータ型変換をサポートするには、**conversion** ライブラリ内で複数の変換関数を使用することができます。多くの場合、MapForce は、必要な変換を自動的に作成し、これらの関数は特別な場合のみに使用される必要があります。

入力ノードが異なる型の場合、例えば、整数と文字列、変換関数を使用して文字列または数値の比較を行うことができます。



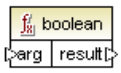
上のサンプルでは、最初の定数は型文字列で、文字列 "4"を含んでいます。 2番目の定数は、定数 12 を含んでいます。 2つの値を明示的に比較するには型が一致しなくてはなりません。

number 関数を最初の定数に追加するには、文字列の定数を数値 4に変換します。比較の結果は "true" になります。

数値関数が使用されない場合、例えば、 4がパラメーターに直接接続されている場合、文字列の比較が生じ、結果は false になります。

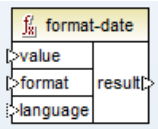
7.6.3.1 boolean

入力により得られた数値型の値をブール型へ変換します。入力が 0の場合は "false" が返され、1の場合は "true" になり、(equal や greater といった) logical 関数や、フィルター、またはif-else 関数の入力として使用することができます。



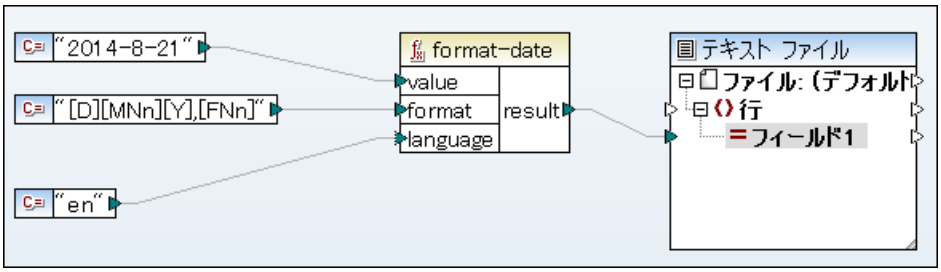
7.6.3.2 format-date

`xs:date` 入力の値を、特定のオプションに従い文字列にフォーマットし変換します。



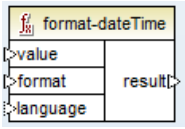
引数	説明
value	フォーマットされる日付。
format	日付がフォーマットされる方法を識別する文字列をフォーマットします。この引数は、**`format-dateTime`** 関数内の `format` 引数 と同じ方法で使用されます。
language	オプションの引数。与えられると、つきの名前と週の曜日が特定の言語で返されます。有効な値： **en (デフォルト)** 英語 **es** スペイン語 **de** ドイツ語 **ja** 日本語

下のサンプルでは、出力の結果は、以下の通りです："2014年、8月、14日、木曜日"。この値をスペイン語に翻訳するには、引数の値を`es` に設定します。



7.6.3.3 format-dateTime

日時の値 (xs:dateTime) を文字列に変換します。日時の文字列の表示は format 引数の値に従いフォーマットされます。



引数	説明
value	変換する xs:dateTime 型の値です。
format	**値**がフォーマットされる方法を識別するフォーマット文字列。
language	オプションの引数。与えられると、つきの名前と週の曜日が特定の言語で返されます。有効な値： **en (デフォルト)** 英語 **es** スペイン語 **de** ドイツ語 **ja** 日本語

メモ:
値はターゲットの型へキャストされるため、関数の出力 (result) が string 型以外のノードへ接続されている場合、書式が失われる可能性があります。ターゲットコンポーネントのコンポーネント設定にて**値をターゲットの方にキャスト**チェックボックスのチェックを外すことで、自動キャストを無効にすることができます(次を参照してください：コンポーネント設定の変更)。

format 引数は、角括弧に囲まれた変数マーカーと呼ばれる文字列により構成されます。角括弧の外にある文字は、定数として結果に反映されます。角括弧を定数文字として使用する場合、括弧を重ねあわせて記述する必要があります。

各変数マーカーは、表示する日付や時刻を表す識別子、オプションの書式修飾子、プレゼンテーション修飾子、幅修飾子により構成され、これらオプションの修飾子はコンマにより分割されます。

```
format := (literal | argument)*
argument := [component(format)?(presentation)?(width)?]
width := , min-width ("-" max-width)?
```

コンポーネントは以下のとおりです：

識別子	説明	デフォルトの表示
Y	年 (絶対値)	4 桁の数値 (2010)
M	その年の月	1-12
D	その月の日付	1-31
d	その年の日付	1-366
F	その週の曜日	曜日の名前 (言語に依存)

識別子	説明	デフォルトの表示
W	その年の週	1-53
w	その月の週	1-5
H	時（24時間）	0-23
h	時（12時間）	1-12
P	A.M. または P.M.	アルファベット(言語に依存)
m	その時間の分	00-59
s	その分の秒	00-59
f	小数点以下の秒	数値、小数点1つ
Z	UTC からのタイムゾーンオフセット	+08:00
z	GMT からのタイムゾーンオフセット	GMT+n

書式修飾子は以下であることができます：

文字	説明	使用例
1	最初に 0を伴わない数値書式：1,2,3,...	1, 2, 3
01	2桁の数値書式：01、02、03、...	01, 02, 03
N	コンポーネント名（大文字）	MONDAY, TUESDAY [1)]
n	コンポーネント名（小文字）	monday, tuesday [1)]
Nn	コンポーネント(頭文字が大文字)	Monday, Tuesday [1)]

メモ： N、n、Nn 修飾子は以下のコンポーネントでのみサポートされます：M、d、D。

width 修飾子がある場合、コンマと一緒に表示され、以下のような形式になります：

```
, min-width ("-" max-width)?
```

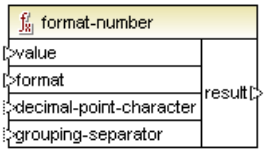
`format-dateTime` 関数を使用して、下のテーブルは `xs:dateTime` 値のフォーマットのサンプルを表示しています。"Value" 列は、`value` 引数に与えられる値を指定します。"Format" 列は、`format` 引数の値を指定します。"Result" 列は、関数により返される内容を表示しています。

日時	書式文字列	結果
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]`	2003-11-03
2003-11-03T00:00:00	`[Y]-[M,2]-[D,2] [H,2]:[m]:[s]`	2003-11-03 00:00:00

日時	書式文字列	結果
2010-06-02T08:02	`[Y] [MNn] [D01] [F,3-3] [d] [H]:[m]:[s].[f]`	2010 June 02 Wed 153 8:02:12.054
2010-06-02T08:02	`[Y] [MNn] [D01] [F,3-3] [d] [H]:[m]:[s].[f] [z]`	2010 June 02 Wed 153 8:02:12.054 GMT+02:00
2010-06-02T08:02	`[Y] [MNn] [D1] [F] [H]:[m]:[s].[f] [Z]`	2010 June 2 Wednesday 8:02:12.054 +02:00
2010-06-02T08:02	`[Y] [MNn] [D] [F,3-3] [H01]:[m]:[s]`	2010 June 2 Wed 08:02:12

7.6.3.4 format-number

XSLT 1.0 や XSLT 2.0、Java、C#、C++ や内蔵の実行エンジンで利用することができます。



引数	説明
`value`	必須の引数。フォーマットする値。
`format`	必須の引数。数値を整形（フォーマット）するためのフォーマット文字列。 この引数は**`format-dateTime`** 関数内の `format` 引数と同じ方法で使用されます。
`decimal-point-format`	オプションの引数。小数点に使用される文字。デフォルトでは "." になります
`grouping-separator`	オプションの引数。数値グループを分割するのに使用されるセパレーター /デリミタ。デフォルトでは "," になります。

メモ: 値はターゲットの型へキャストされるため、関数の出力 (esult) が string 型以外のノードへ接続されている場合、書式が失われる可能性があります。ターゲットコンポーネントのコンポーネント設定にて**値をターゲットの方にキャスト**チェックボックスのチェックを外すことで、自動キャストを無効にすることができます。

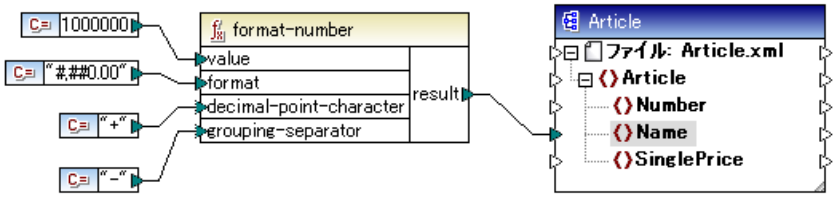
フォーマット:

```
format := subformat (;subformat)?
subformat := (prefix)? integer (.fraction)? (suffix)?
prefix := 特別文字以外の文字
suffix := 特別文字以外の文字
integer := (#)* (0)* ( allowing ',' to appear)
fraction := (0)* (#)* (allowing ',' to appear)
```

最初のsubformat が正の数のフォーマットに使用され、２番目のsubformat が負の数に対して使用されます。subformat が１つだけ指定された場合、同じsubformat が正の数と負の数に使用され、負の数の場合はprefix の前にマイナス記号が挿入されます。

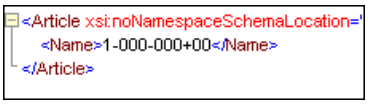
特別な文字	デフォルト	説明
zero-digit	0	result にてこの場所に常に数値が表示されます。
digit	#	最終的に 0になる場合を除き、result の文字列にてこの場所に常に数値が表示されます。
decimal-point	.	数値の整数部と小数点以下を分けるために使用されます。
grouping-seperator	,	数値のグループ分けを行うのに使用されます。
percent-sign	%	数値を100倍して、パーセンテージとして表示します。
per-mille	‰	数値を1000倍して、パーミルとして表示します。

decimal-point-character とgrouping-separator に使用される文字は、それぞれ常に "."と ","になります。しかしこれらのノードに定数をマッピングすることで、出力されるフォーマットを変更することができます。



上に示されるformat-number 関数のマッピング結果を以下に示します：

- decimal-point-character が "+"に変更されました。
- grouping-separator が "-"に変更されました。



四捨五入

端数処理には四捨五入が使用されます。つまり端数が 0.5以上であれば切り上げが行われ、0.5未満であれば切り捨てが行われます。このメソッドは、生成されたコードとビルトイン実行エンジンでのみ使用することができます。

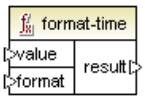
XSLT 1.0 では四捨五入のモードが定義されておらず、XSLT 2.0 では偶数丸めが行われます。

数値	書式文字列	結果
1234.5	#,##0.00	1,234.50
123.456	#,##0.00	123.46
1000000	#,##0.00	1,000,000.00
- 59	#,##0.00	- 59.00
1234	###0.0###	1234.0
1234.5	###0.0###	1234.5
.00025	###0.0###	0.0003
.00035	###0.0###	0.0004
0.25	#00%	25%

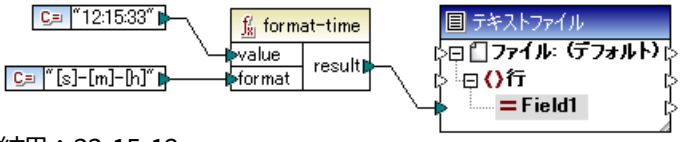
数値	書式文字列	結果
0.736	#00%	74%
1	#00%	100%
- 42	#00%	- 4200%
- 3.12	#.00;(#.00)	(3.12)
- 3.12	#.00;#.00CR	3.12CR

7.6.3.5 format-time

xs:time 入力値を文字列へ変換します。`format` 引数は **`format-dateTime`** 関数内の `format` 引数と同じ方法で使用されます。



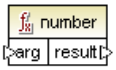
例



結果：33-15-12

7.6.3.6 number

入力値の文字列を数値に変換します。また、ブール型入力も数値に変換します。



7.6.3.7 string

入力値を文字列型の出力値に変換します。関数はノードのテキストコンテンツとして使用することもできます。

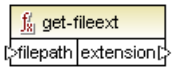
XML 複合型が入力ノードとなる場合、全ての子要素も単一の string として出力されます。

7.6.4 core | file path functions

ファイルパス (file path) 関数を使用することで、ファイルパスのデータやファイルの拡張子へアクセスまたは操作することができ、更なるマッピング処理に使用することができます。これらの関数は MapForce にて使用可能な全ての言語で使用することができます。

7.6.4.1 get-fileext

ドット "."文字を含むファイルパスの拡張子を返します。



例 : c:\data\Sample.mfd' からは 'mfd' が返されます。

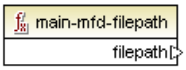
7.6.4.2 get-folder

ファイルパスから、フォルダーを分けるスラッシュやバックスラッシュ (日本語環境では円マーク) を含むフォルダー名を返します。

例 : c:/data/Sample.mfd' からは c:/data/' が返されます。

7.6.4.3 main-mfd-filepath

メインマッピングを含んでいるmfd ファイルのフルパスを返します。mfd ファイルが保存されていない場合、空の文字列が返されます。



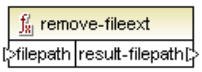
7.6.4.4 mfd-filepath

関数がメインマッピング内で呼び出された場合、main-mfd-filepath と同一の値 (メインマッピングが含まれているmfd ファイルのフルパス) が返されます。mfd ファイルが保存されていない場合、空の文字列が返されます。

mfd ファイルへインポートされた**ユーザー定義関数**から関数が呼び出された場合、そのユーザー定義関数の**定義**を含んでいるインポートされたmfd ファイルのフルパスが返されます。

7.6.4.5 remove-fileext

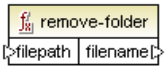
ドット文字を含むファイルパスの拡張子が削除されます。



例 : c:/data/Sample.mfd' からは c:/data/Sample' が返されます。

7.6.4.6 remove-folder

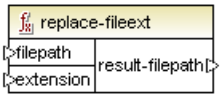
ファイルパスから、スラッシュやバックスラッシュ (日本語環境では円マーク) を含むフォルダー名が削除されます。



例 : 'c:/data/Sample.mfd' からは 'Sample.mfd' が返されます。

7.6.4.7 replace-fileext

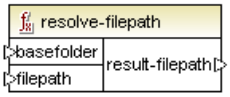
filepath パラメーターから得られたファイルパスの拡張子を、extension パラメーターから得られたファイルパスの拡張子で置き換えます。



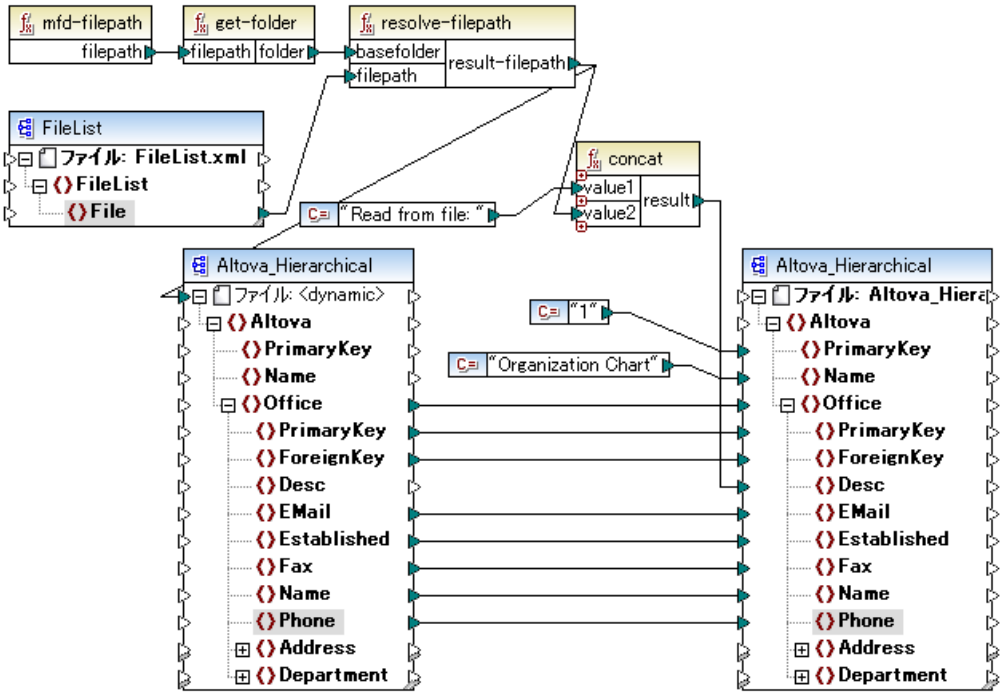
例 :filepath 入力パラメーターに'c:/data/Sample.mfd' が与えられ、extension 入力パラメーターに '.mfp' が与えられた場合、'c:/data/Sample.mfp' が返されます。

7.6.4.8 resolve-filepath

相対ファイルパスから相対または絶対ベースフォルダーを解決します。関数では "." (カレントディレクトリ) と ".." (親ディレクトリ) を使用することができます。



..\MapForceExamples フォルダーに含まれている**MergeMultipleFiles_List.mfd** マッピングからサンプルを確認することができます。

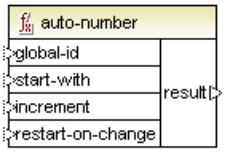


7.6.5 core | generator functions

core / generator 関数ライブラリは値を生成する関数を含みます。

7.6.5.1 auto-number

auto-number 関数により、指定されたパラメーターに従うかたちでコンポーネントのターゲットノードが統合されます。関数の結果は**start_with** から始まる値で、**インクリメント**で増加します。デフォルトの値 :開始=1、=1 づつ増加。双方のパラメータは負である可能性があります。

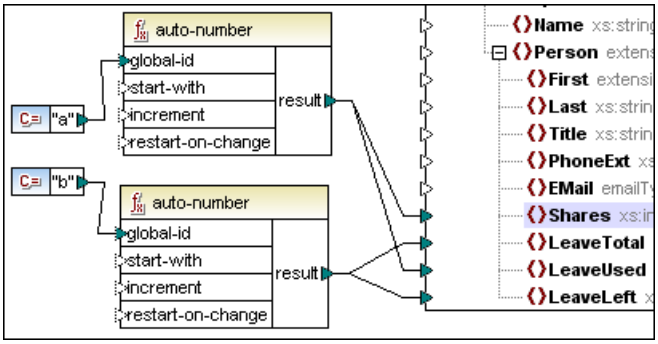


(auto-number 関数の) result コネクタがターゲットノードに**直接**接続されていることを確認してください。生成されたマッピングコードから関数が呼び出される正確な順番は定義されていません。MapForce ではキャッシュされた計算結果を再利用するか、任意の順序で条件式を再評価するかを選択することができます。従って、auto-number 関数を使用する際には特に注意する必要があります。

global-id
このパラメーターにより、独立した2つのauto-number 関数内に蓄えられているシーケンス出力の内容を同期することができます。

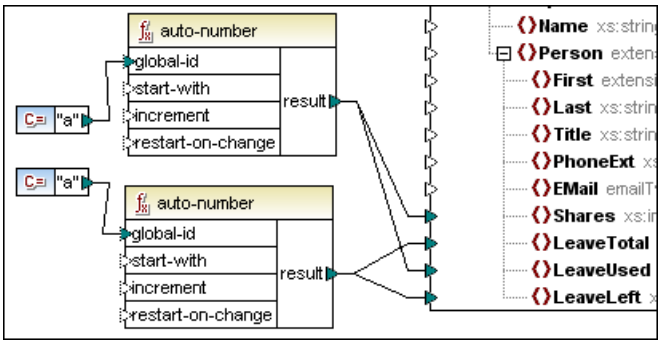
2 つのauto-number 関数が同一のglobal-id を**持たない**場合、各関数により別々の番号がターゲットアイテムにて追加されます。以下にあるサンプルでは、各関数に異なるgloba-id のa とb が与えられています。

マッピングの出力は 1、1、2、2 となります。上部にあるauto-number 関数により最初の 1が、下部のauto-number により2番目の 1が与えられます。



2 つの関数が**同一**のglobal-id である**a** を持っている場合、現在のauto-number の値が各関数により「共有」され、値が同期されることになります。

以下に示されるマッピングの出力は、1,2,3,4 となります。上部にある関数から最初のアイテムに 1が与えられ、2番目のアイテムには 2が与えられます。

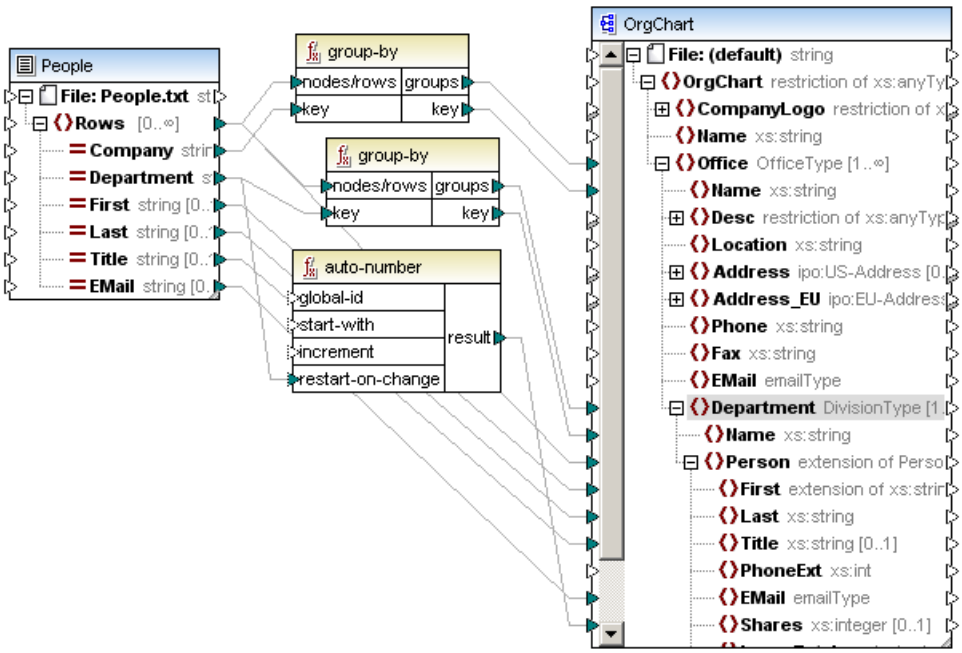


start-with
自動連番シーケンスの初期値として使用される値です。デフォルトでは 1が使用されます。

increment
シーケンスのインクリメントに使用される値です。デフォルトでは 1が使用されます。

restart on change
接続されているアイテムのコンテンツが変更された場合、自動連番のカウンターを "start-with" で指定された値にリセットします。

以下にあるサンプルでは、start-with とincrement の両方にデフォルトの 1が指定されています。Departmento のコンテンツが変更されると（つまり部門名が変更されると）連番に使用されているカウンターの値が 1にリセットされ、新たな部門に対して使用されるようになります。



7.6.6 core | logical functions

logical 関数は入力されたデータを比較して、"true" または "false" というboolean 型の出力を生成します。通常、フィルターを使用してサブセットを渡す前段階で、データを評価するために使用されます。

入力パラメーター = **a** | **b** または **value1** | **value2**
出力パラメーター = 結果

2 つの入力ノードの評価結果は、入力の値と比較に使用されたデータ型により異なります。

例 整数の値 4と 12の 'less than' 比較は、4が 12より小さいため、boolean 値 "true" を返します。 2つの入力文字列が '4'と '12'を含む場合、 '4'はアルファベットの順序でオペランド (12)の最初の文字 '1'よりも大きいため、構文的分析はの出力値 "false" を返します。

すべての入力データ型が同じ型の場合、 例 全ての入力ノードは数値型、または、文字列型であり、共通の型のための比較が行われます。

入力ノードが異なる型の場合、 例 整数と文字列 または 文字列と日付 そして、使用されるデータ型は、2つの入力型のうち **最も一般的な (最も制限のない)入力データ型です。**

2つの値を比較する前に、すべての入力の値は、共通のデータ型に変換されます。データ型 "文字列 " が "整数 "より制限のない前のサンプルを使用ます。整数の値 4を文字列 '12'と比較し、整数の値 4を文字列 '4'に変換し、文字列 '12'と比較します。

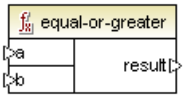
メモ: Logical 関数をnull 値の存在をテストするために使用することはできません。null 値をlogical 関数の引数として与えると、null 値を返します。null 値の扱い方に関しての詳しい情報は、Nil 値 /Nillable を参照してください

7.6.6.1 equal

a=b の場合、結果はtrue となり、それ以外の場合 false となります。

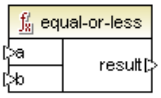
7.6.6.2 equal-or-greater

入力 a がb 以上の場合 true を返し、それ以外の場合にfalse を返します。



7.6.6.3 equal-or-less

入力 a がb 以下の場合 true を返し、それ以外の場合にfalse を返します。



7.6.6.4 greater

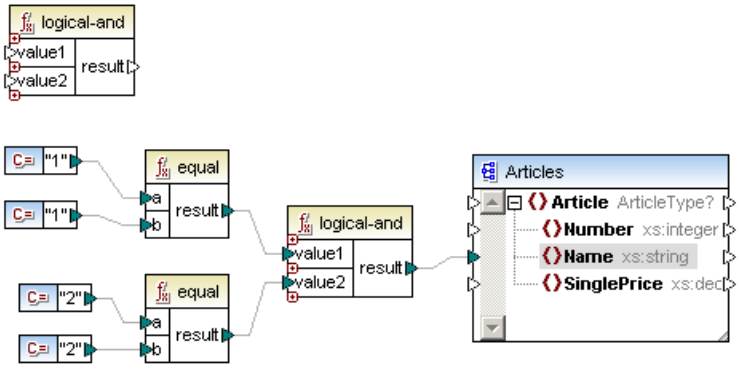
入力 a がb よりも大きい場合 true を返し、それ以外の場合にfalse を返します。

7.6.6.5 less

入力 a がb よりも少ない場合 true を返し、それ以外の場合にfalse を返します。

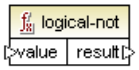
7.6.6.6 logical-and

value1 とvalue2 の論理積が求められ、両方の値がtrue の時に、関数の結果もtrue となります。どちらかでもfalse の場合には結果もfalse となります。

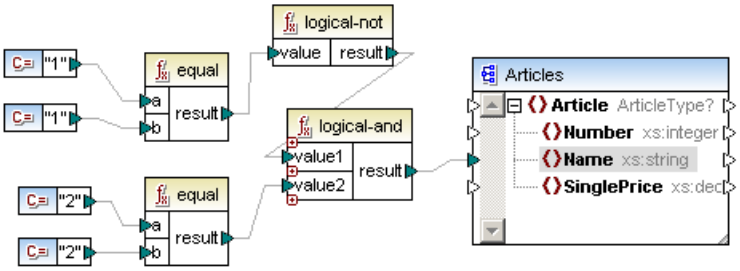


7.6.6.7 *logical-not*

論理状態 結果を反転します。入力がtrue であればlogical-not の出力はfalse になり、入力がfalse であれば出力がtrue になります。

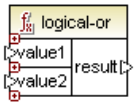


以下に示されるlogical-not 関数により、equal 関数の結果が反転されます。これにより、2 つあるequal 関数のどちらか一方がfalse で、もう一方がtrue になっている状態でしかlogical-and 関数からtrue が得られなくなります。



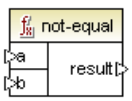
7.6.6.8 *logical-or*

両方の入力関数をboolean 型に揃える必要があります。value1 とvalue2 の少なくともどちらか一方がtrue となっている場合、関数の結果もtrue になります。両方の値がfalse の場合、結果もfalse になります。



7.6.6.9 *not-equal*

入力 a とb が等しく無い場合にtrue が返されます。

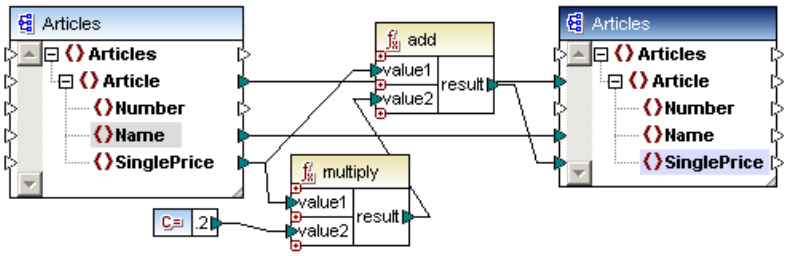


7.6.7 core | math functions

データを使った初歩的な計算を行うためにmath 関数を利用することができます。これら関数を使用して期間や日時の計算はできないという点に注意してください。

入力パラメーター = **value1** | **value2**
出力パラメーター = **result**

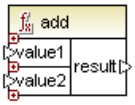
入力値は数値へ自動的に変換され、処理が行われます。



上に示されるサンプルマッピングでは、ターゲットコンポーネントにマッピングされた各製品の価格に対して20%の消費税が上乗せされます。

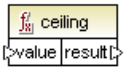
7.6.7.1 add

value1 と**value2** が加えられた数値が結果として返されます。



7.6.7.2 ceiling

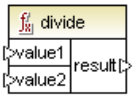
value にて与えられた値よりも大きな最小の整数値が返されます。つまり**value** にて与えられた数値の次に大きな整数が結果として返されます。



例えば、division 関数の結果が 11.2だった場合、ceiling 関数を使用することで、結果は (次に大きな整数値の) 12となります。

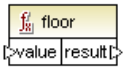
7.6.7.3 divide

value1 を**value2** で割った数が返されます。除算結果の精度はターゲット言語により変化します。round-precision 関数を使用することで、結果の精度を定義することができます。



7.6.7.4 floor

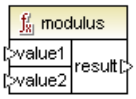
value にて与えられた値よりも小さな最大の整数値が返されます。つまり**value** にて与えられた数値の次に小さな整数が結果として返されます



例えば、division 関数の結果が 11.2 だった場合、floor 関数を使用することで、結果は（次に小さな整数値の）11 となります。

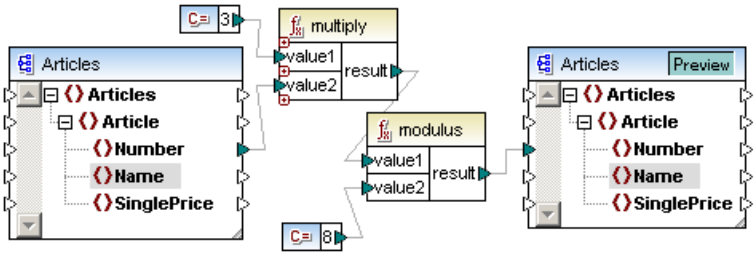
7.6.7.5 modulus

value1 を**value2** で割った値の余りが返されます。



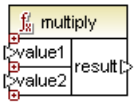
以下に示されるマッピングでは、Numbers の値が 3 倍されて modulus 関数の value1 へ渡されています。入力値は 3、6、9、12 となります。

modulus 関数の value2 入力に 8 を与えると、余りとして 3、6、1、4 が返されます



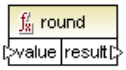
7.6.7.6 multiply

value1 と**value2** をかけ合わせた（乗算）値が返されます。



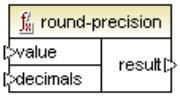
7.6.7.7 round

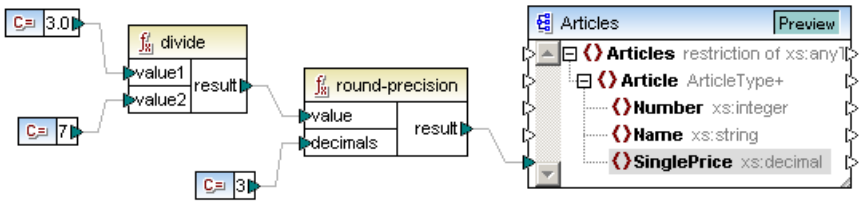
value により与えられた値の四捨五入結果が整数値で返されます。値が 2 つの整数の間の場合、"正の無限大への丸め" アルゴリズムが使用されます。例えば、"10.5" の値は "11" に四捨五入され、"-10.5" の値は "-10" に四捨五入されます。



7.6.7.8 round-precision

"decimals" により得られた値を小数点以下の有効数字として value 値の四捨五入を行います。

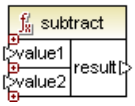




上のマッピングにより 0.429 という値が得られます。XML ファイルにて結果を正しく表示するには、xs:decimal 型の要素へマッピングするようにしてください。

7.6.7.9 subtract

value1 から**value2** を引いた数値を返します。



7.6.8 core | node functions

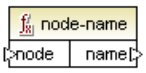
ノード関数を使用してノードにアクセスしたり、ノードを特定の方法で処理することができます。

7.6.8.1 is-xsi-nil

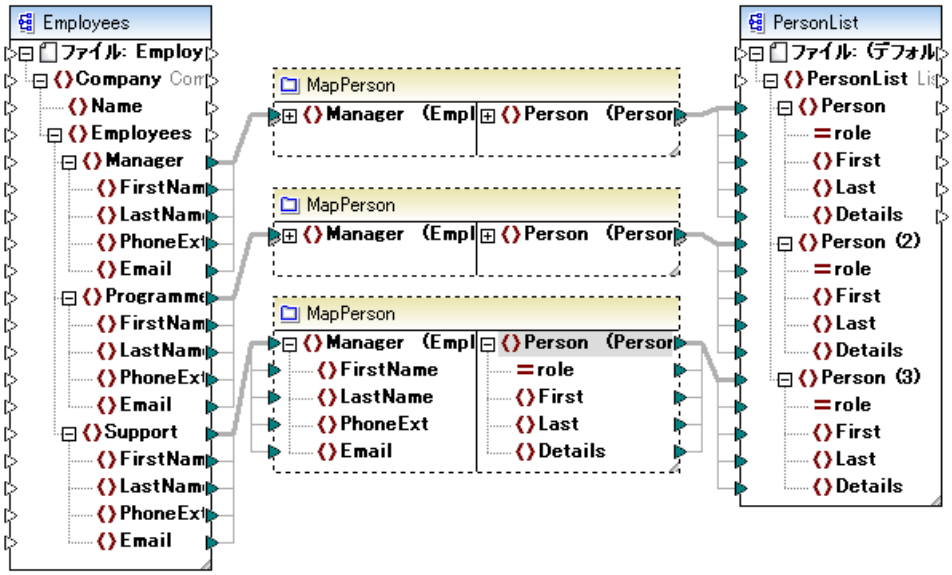
ソースコンポーネントの要素ノードに xsi:nil 属性が含まれており、属性の値が "true" になっている場合、true (<OrderID>true</ OrderID>) を返します

7.6.8.2 node-name

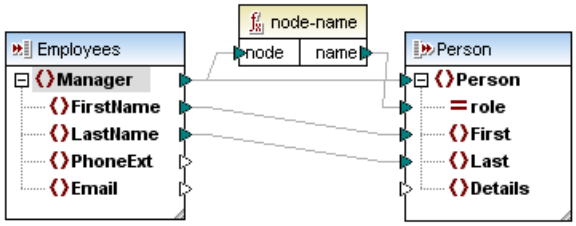
接続されているノードの修飾名 (QName) を返します。もしノードが XML text() の場合、空の QName が返されます。この関数は名前付きのノードに対してしか使用することはできません。(fn:node-name を呼び出す) XSLT をターゲット言語としている場合、関数は名前を持っていないノードに対して空のシーケンスを返します。



- データベースのテーブル/フィールドからの名前取得はサポートされていません。
- XBRL とExcel はサポートされていません。
- ファイル入力ノードの名前取得はサポートされていません。
- Web サービスのノードは、以下の例外を除き XML ノードと同じような動作をします：
  - "part" からのノード名はサポートされていません。
  - ルートノード ("Output" または "Input") からのノード名はサポートされていません。



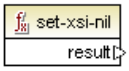
MapPerson のユーザー定義関数は**node-name** を使用することで入力**ノード**の名前を取得しており、その情報を role 属性に渡しています。ユーザー定義関数内にあるEmployees.xsd のルートノードは "Manager" として定義されています。



Manager はユーザー定義関数の**外側**からデータを取得します。データは "Manager"、"Programmer"、"Support" のどれかになります。このデータがPersonList のrole 属性に渡されます。

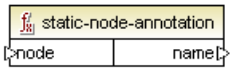
7.6.8.3 set-xsi-nil

ターゲットノードにxsi:nil をセットします。



7.6.8.4 static-node-annotation

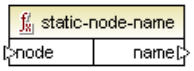
接続されたノードの注釈と共に文字列を返します。入力は以下である必要があります:(i) ソースコンポーネント ノード、または (ii) 呼び出しマッピング内のノードに直接接続されているパラメーターに直接接続されているインライン関数。



接続は直接である必要があります。フィルター、または非インラインユーザー定義関数をパススルーすることはできません。これは、接続されたノードから取得されたテキストの生成時に置き換えられた擬似関数で、全ての言語で使用することができます。

7.6.8.5 static-node-name

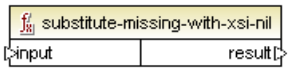
接続されたノードの名前と共に文字列を返します。入力は以下である必要があります:(i) ソースコンポーネント ノード、または (ii) an呼び出しマッピング内のノードに直接接続されているパラメーターに直接接続されているインライン関数。



接続は直接である必要があります。フィルター、または非インラインユーザー定義関数をパススルーすることはできません。これは、接続されたノードから取得されたテキストの生成時に置き換えられた擬似関数で、全ての言語で使用することができます。

7.6.8.6 substitute-missing-with-xsi-nil

ソースコンポーネントにて見つからない値 (または null 値) をxsi:nil 属性に置き換えます



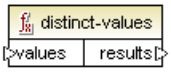
7.6.9 core | sequence functions

MapForce では入力シーケンスとそのコンテンツを処理するために用意されているsequence 関数を使用することができます。**key** 入力パラメーターの値 (コンテンツ) をノード行に接続することで、そのシーケンスをグループとして扱うことができます。

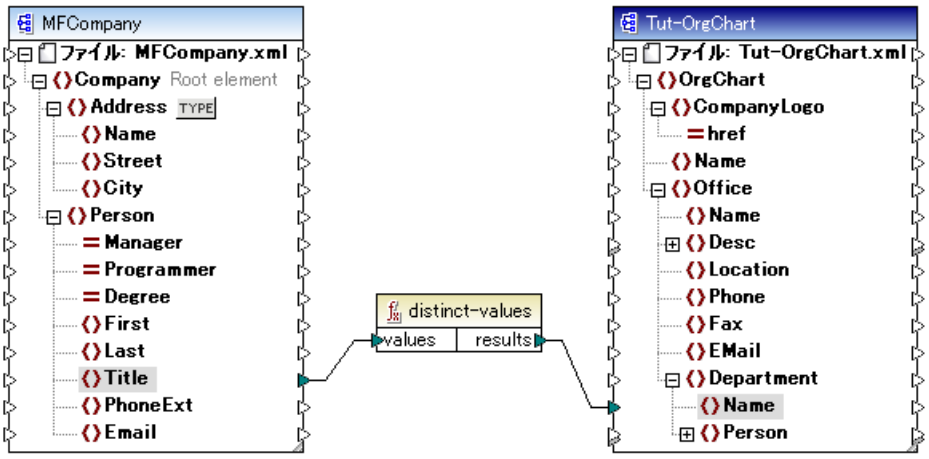
- 任意のデータ型に接続された入力パラメーター**key** の値は**group-adjacent** や**group-by** で使用できるようにstring 型へ変換されます。
- Boolean 型の**bool** 入力パラメーターは**group-starting-with** ならびに**group-ending-with** にて使用されます。
- 出力パラメーターの**key** は、現在のグループキーになります。

7.6.9.1 distinct-values

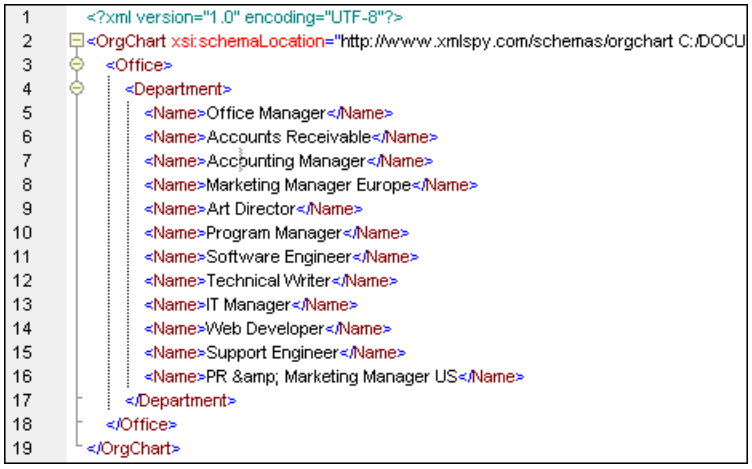
シーケンスにて重複した値を取り除き、ユニーク (一意) なアイテムをターゲットコンポーネントへ渡します。



以下の例では、ソースコンポーネントの "Title" アイテムにあるコンテンツがスキャンされ、一意 (ユニーク) な役職だけが、ターゲットコンポーネントの Department / Name アイテムへマッピングされます。

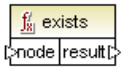


ソースコンポーネント内にある Title アイテムの順番はターゲットコンポーネントでも保持されていることに注目してください。



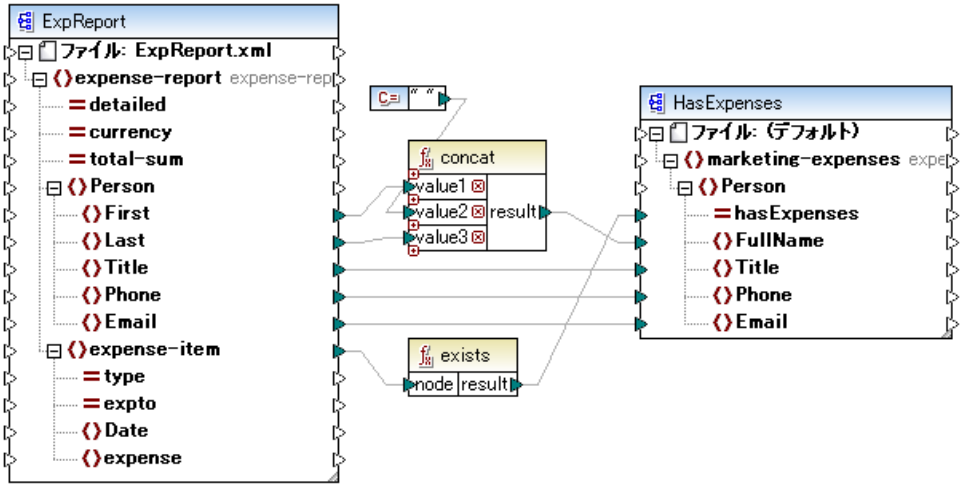
7.6.9.2 exists

ノードが存在する場合に true を返し、存在しない場合は false を返します。



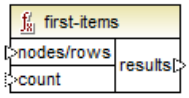
...\MapForceExamples フォルダー内部の **"HasMarketingExpenses.mfd"** ファイルには、以下に示されるサンプルが収められています。

expense-item がソース XML にて存在する場合、ターゲット XML/スキーマファイルの "hasExpenses" 属性値が true となります。



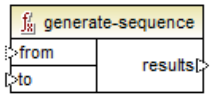
7.6.9.3 first-items

"count" パラメーターにより数値がX 提供される、入力シーケンスの最初の "X" アイテムを返します。例 :値 3がcount パラメーターとnodes/row パラメーターの親ノードにマップされ、最初の3つのアイテムが出力としてリストされます。



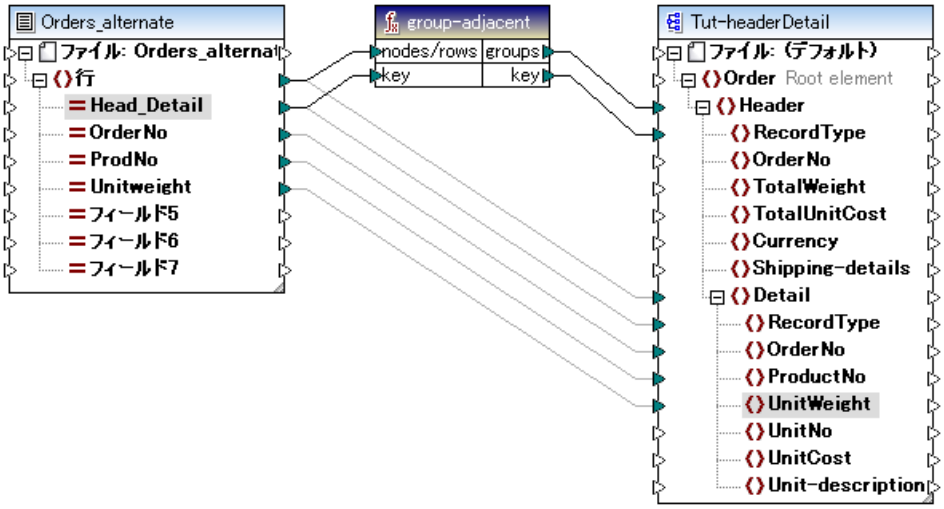
7.6.9.4 generate-sequence

"from" と "to" パラメーターを境界として使用し、整数のシーケンスを作成します。



7.6.9.5 group-adjacent

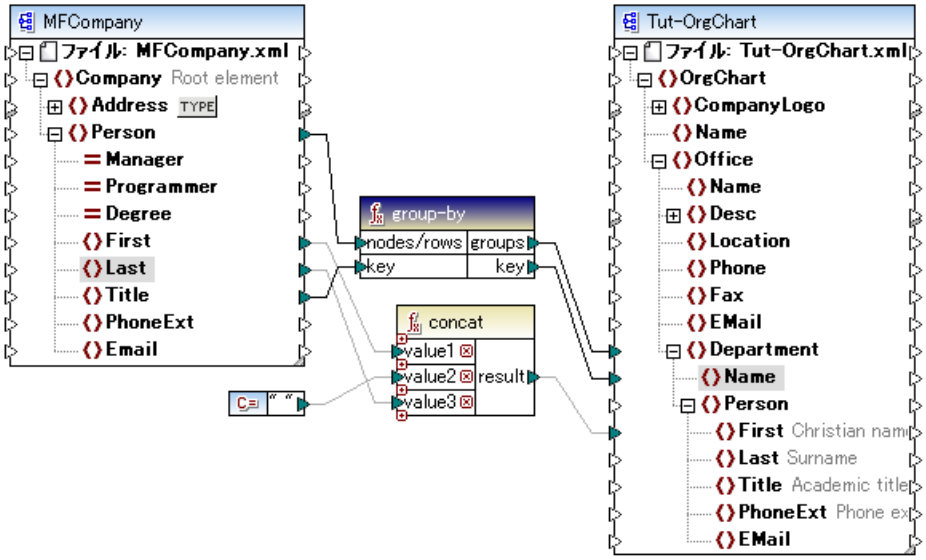
入力シーケンスのノード行を、同じ**キー**を持つ隣接したグループへ変換します。group-adjacent はグループ化のキーとしてノード/アイテムの**コンテンツ**を使用する点に注意してください



7.6.9.6 group-by

入力シーケンスのノード(行)を同じキーを持つグループへ変換します(グループが隣接しているとは限りません)。入力シーケンスにて出現する**キー**の順序がグループの出力に使用されます:

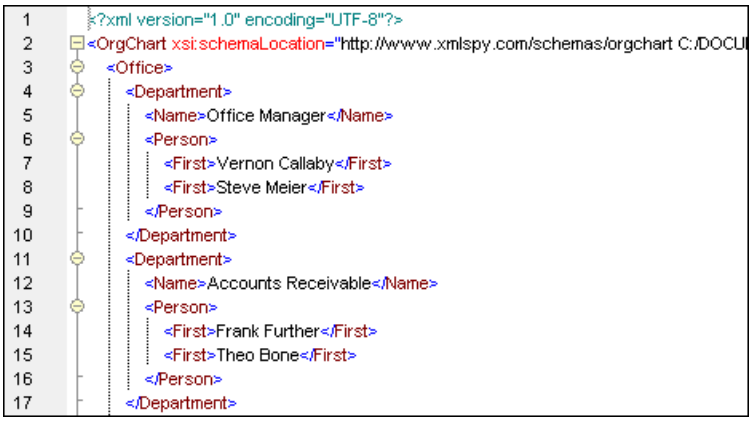
- ソースコンポーネントからグループ化を行うためのキーはTitle アイテムとなります。このキーにより、company の従業員 (Person) がグループ化されます。
- ターゲットコンポーネントのDepartment/Name にグループ名が出力されるとともに、従業員の苗字 (Last) と名前 (First) が連結され、Person/First 子アイテムへ出力されます



group-by ではノード/アイテムの**コンテンツ**をグループのキーとして使用します! Title フィールドのコンテンツが従業員のグループ化、そしてターゲットにあるDepartment/Name アイテムへのマッピングにて使用されます。

メモ: group-by 関数は、サンプルとそのマッピング結果からも分かる通り、ソースドキュメントからターゲットドキュメントへの**暗黙的なフィルタリング**としても機能します。ターゲットドキュメントでは、Person アイテムが、Title キーにより**グループ化**された各 Department アイテムに含まれています。

出力ボタンをクリックすることで、グループ化処理の結果を確認することができます。

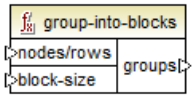


7.6.9.7 group-ending-with

この関数により入力シーケンスのノード行から、bool 入力がtrue の時の入力が最後の要素となるグループが作成されます。

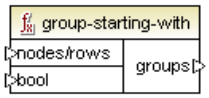
7.6.9.8 group-into-blocks

block-size パラメーターにより与えられた数値により定義された同じ値を持つ、入力シーケンス**nodes/rows** をブロックにグループ化します。

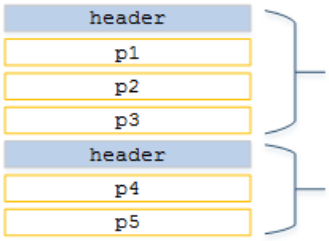


7.6.9.9 group-starting-with

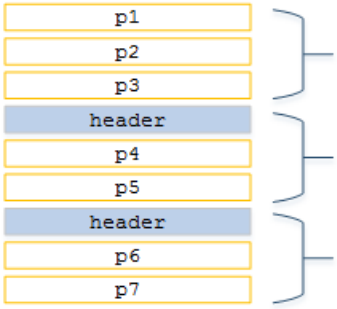
この関数により入力シーケンスのノード行から、**bool** 入力がtrue の時の入力が最初の要素となるグループが作成されます。



次のサンプルは、ノードのシーケンスがノードの "header" が発生し、**ブール値の**`true` を返す箇所を示しています。このノードのシーケンスに**`group-starting-with`** 関数を適用すると、結果として発生する2つのグループは、以下のとおりです。

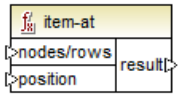


シーケンス内の最初のノードは、**ブール**の値に関係なく、新しいグループで開始することにご注意してください。すなわち、下のようなシーケンスは3つのグループを作成します。



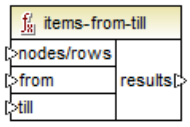
7.6.9.10 item-at

position パラメーターにより与えられたポジションにある**nodes/rows** を返します。を返します。最初のアイテムは、ポジション "1"にあります。



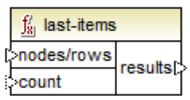
7.6.9.11 items-from-till

"from" と "till" パラメーターを境界として使用し、**nodes/rows** のシーケンスを返します。最初のアイテムは、ポジション "1" にあります。



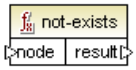
7.6.9.12 last-items

"count" パラメーターにより与えられた数値が X である場所のシーケンスの最後の "X" **nodes/rows** を返します。最初のアイテムは、ポジション "1"にあります。



7.6.9.13 not-exists

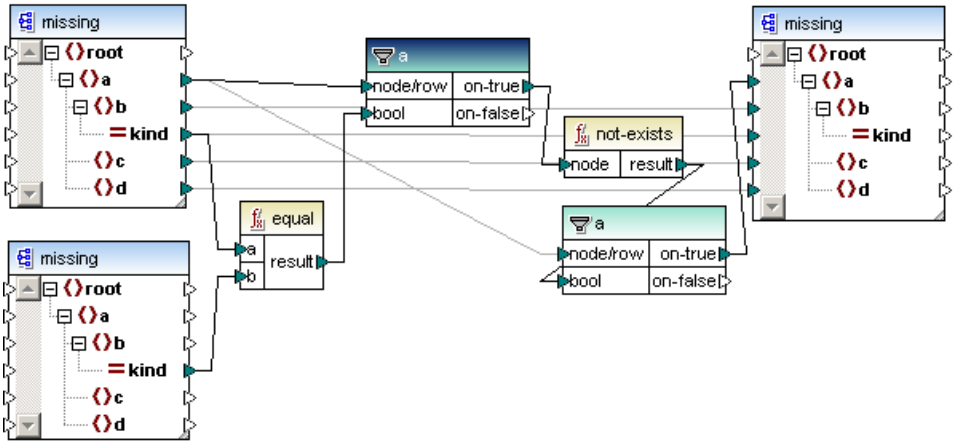
ノードが存在する場合 false を返します。



以下の例では、not-exists 関数を使用することでソースファイルには存在しないノードのマッピングを行います。

このマッピングでは、以下の処理を行います

- 2 つのソースXML ファイルにあるノードを比較します。
- 2 番目のソースXML ファイルに存在しないノードを、1番目のソースXML ファイルからフィルタリングします。
- 見つからないノードとそのコンテンツだけをターゲットファイルへマッピングします。



以下に 2 つのXML インスタンスを示します。以下の点が互いに異なっています：

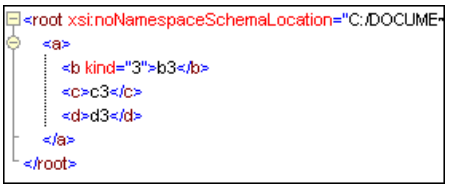
- 左側にある**a.xml** では、b.xml には無い <b **kind="3"**> が含まれています。
- 右側にある**b.xml** では、a.xml には無い <b **kind="4"**> が含まれています。

- equal 関数により、両方のXML ファイルにあるkind 属性が比較され、結果がフィルターコンポーネントへ渡されます。
- non-exists 関数が最初のフィルターの後に配置されており、各ソースファイルで見つからないノードを選択しま

す。

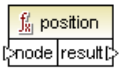
- 2 番目のフィルターは、a.xml から得られた見つからないノードとそのデータだけを渡すためだけに使用されます。

マッピングにより、b.xml には存在しないノード <b kind="3"> がターゲットコンポーネントへ渡されます。



7.6.9.14 position

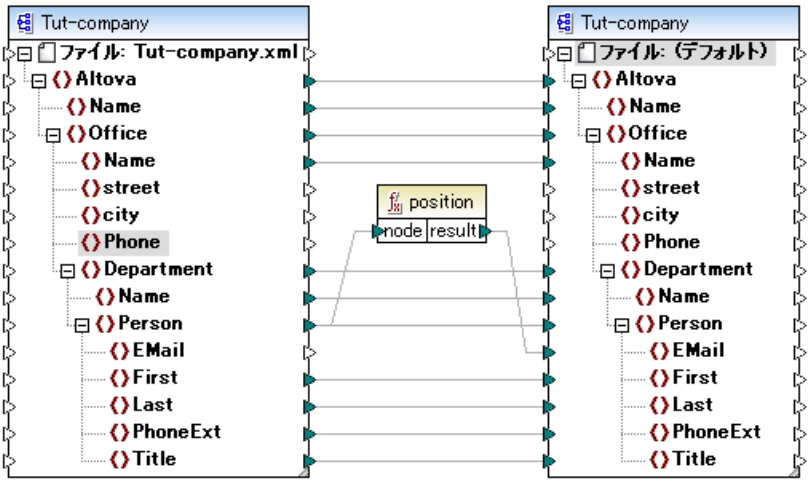
シーケンス内にあるノードの位置を返します。



position 関数を使用することで、シーケンス内部にある特定ノードの位置を決定したり、位置を元にしたフィルタリングを行うことができるようになります。

position 関数の node パラメーターへアイテムを接続することで、コンテキストアイテムが定義されます（以下の例では Person ノード）。

以下のマッピングでは、各部署（Department）の従業員（Person）に対して、（ソース XML にある要素の）位置番号が追加されます。



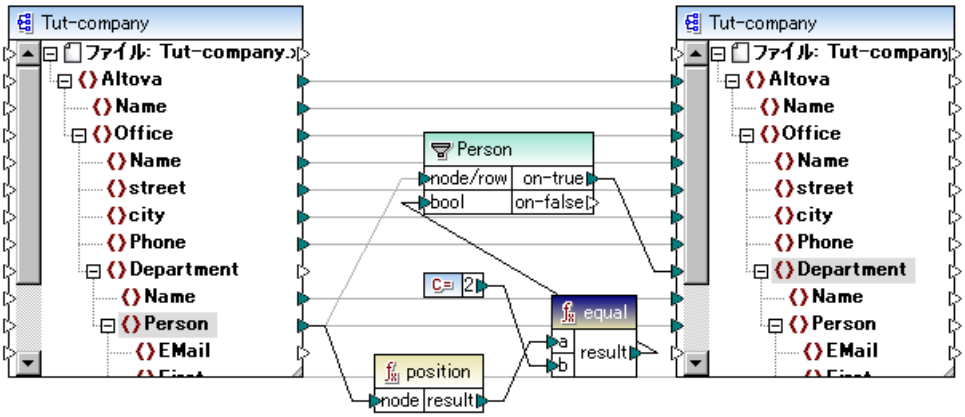
位置番号は、各部署ならびに各オフィスごとにリセットされます。

```
<Office>
  <Name>Microtech, Inc.</Name>
  <Department>
    <Name>Admin</Name>
    <Person>
      <EMail>1</EMail>
      <First>Albert</First>
      <Last>Aldrich</Last>
      <PhoneExt>582</PhoneExt>
      <Title>Manager</Title>
    </Person>
    <Person>
      <EMail>2</EMail>
      <First>Bert</First>
      <Last>Bander</Last>
      <PhoneExt>471</PhoneExt>
      <Title>Accounts Receivable</Title>
    </Person>
```

ポジション関数は特定のノードをフィルターするために使用されます。

ポジション関数はフィルターを使用して、ソースコンポーネント内に特定のポジションを持つ特定のノードをフィルターと共にマップすることができます。

フィルター "node/row" パラメーターとポジション "ノード"は、そのシーケンスの特定のポジションをフィルターアウトするために、同じソースコンポーネントのアイテムに接続されている必要があります。



このマッピングは以下を出力します：

- 各Department 内の第 2番目のPerson
- Altova 内の各オフィス.

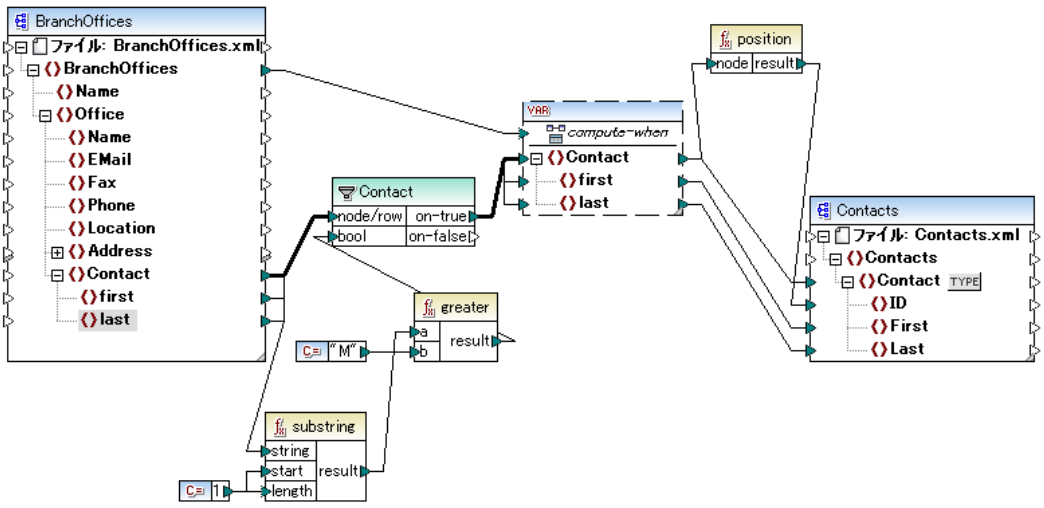
```
<Office>
  <Name>Microtech, Inc.</Name>
  <Department>
    <Name>Admin</Name>
    <Person>
      <EMail>b.bander@microtech.com</EMail>
      <First>Bert</First>
      <Last>Bander</Last>
      <Title>Accounts Receivable</Title>
    </Person>
  </Department>
  <Department>
    <Name>Sales and Marketing</Name>
    <Person>
      <EMail>e.ellas@microtech.com</EMail>
      <First>Eve</First>
      <Last>Ellas</Last>
      <Title>Art Director</Title>
    </Person>
  </Department>
  <Department>
    <Name>Manufacturing</Name>
    <Person>
      <EMail>g.gundall@microtech.com</EMail>
```

フィルターされたシーケンス内のアイテムの位置を検索する:

*フィルターコンポーネント*はシーケンス関数ではないため、*フィルターされたアイテム*のポジションを検索する関数と共には position 関数と共に直接使用することはできませんこれを行うには、"変数" コンポーネントを使用します。

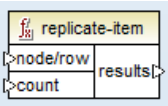
変数コンポーネントの結果は常に*シーケンス*です。例 :*シーケンス*の作成のために使用することのできる区切られた値のリスト。

- 苗字 (Last) の頭文字がM より後 (つまりN 以降) の文字で始まるContact のデータが*フィルタリング*され、変数コンポーネントへ集められます。
- これらのContact 情報が 変数コンポーネントから) ターゲットコンポーネントへ渡されます。
- position 関数により、*フィルタリング*された順にデータが番号付けされます。



7.6.9.15 replicate-item

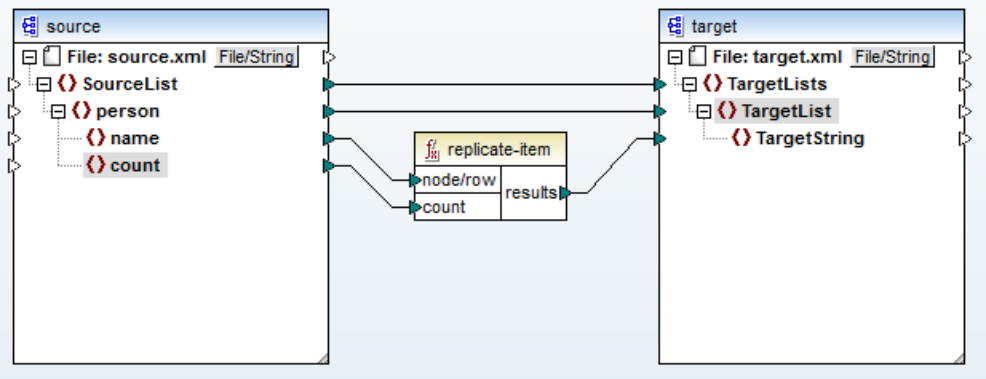
count 引数で指定された回数入力シーケンス内のアイテムを繰り返します。単一のアイテムを node/row 引数に接続すると、関数は、N が count 引数の値である箇所の N アイテムを返します。アイテムのシーケンスを node/row 引数に接続すると、1度に1つのアイテムを処理し、関数は、シーケンス内のそれぞれのアイテムを count 回繰り返します。例：count が 2 の場合、シーケンス (1,2,3) は、(1,1,2,2,3,3) を生成します。



各アイテムのために異なった count 値を与えることができる点に注意してください。各アイテムのために異なった値を与えることができることに注意してください。例：以下の構造を持つソース XML ファイルがあると仮定します：

```
<?xml version="1.0" encoding="UTF-8"?>
<SourceList xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="source.xsd">
   <person>
      <name>Michelle</name>
      <count>2</count>
   </person>
   <person>
      <name>Ted</name>
      <count>4</count>
   </person>
   <person>
      <name>Ann</name>
      <count>3</count>
   </person>
</SourceList>
```

replicate-item 関数を使用して、ターゲットコンポーネント内で何度も個人を異なった回数繰り返し使用することができます。これを達成するには、**replicate-item** 関数の count 入力に各個人の <count> ノードを接続します：



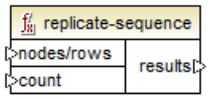
出力は、以下のとおりです：

```
<?xml version="1.0" encoding="UTF-8"?>
<TargetLists xsi:noNamespaceSchemaLocation="target.xsd"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
   <TargetList>
      <TargetString>Michelle</TargetString>
      <TargetString>Michelle</TargetString>
```

```
   </TargetList>
   <TargetList>
      <TargetString>Ted</TargetString>
      <TargetString>Ted</TargetString>
      <TargetString>Ted</TargetString>
      <TargetString>Ted</TargetString>
   </TargetList>
   <TargetList>
      <TargetString>Ann</TargetString>
      <TargetString>Ann</TargetString>
      <TargetString>Ann</TargetString>
   </TargetList>
</TargetLists>
```

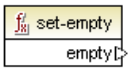
7.6.9.16 replicate-sequence

入力シーケンス内の全てのアイテムを count 引数内で指定された回数複製します。例 :count が 2 の場合、シーケンス (1,2,3) は (1,2,3,1,2,3) を作成します。



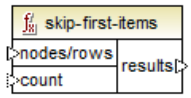
7.6.9.17 set-empty

空のシーケンスを返します。



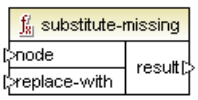
7.6.9.18 skip-first-items

X が "カウント" パラメーターにより提供される数量である、入力シーケンスの最初の "X" アイテム/ノードをスキップし、残りのシーケンスを返します。



7.6.9.19 substitute-missing

この関数は exists と if-else 条件の組み合わせにより構成されます。XML ソースファイルにてノードが存在する場合、現在のフィールドコンテンツがマッピングされ、それ以外の場合はマッピングされた "replace-with" パラメーターの内容が使用されます。

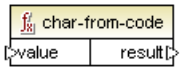


7.6.10 core | string functions

string 関数では、データの一部分を取得、文字列の検証、または文字列から必要な情報を検索するといった、様々な種類のソースデータを処理する為の文字列関数を使用することができます。

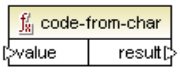
7.6.10.1 char-from-code

value により得られた整数値のUnicode 値を文字に変換した結果が返されます。



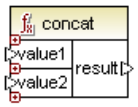
7.6.10.2 code-from-char

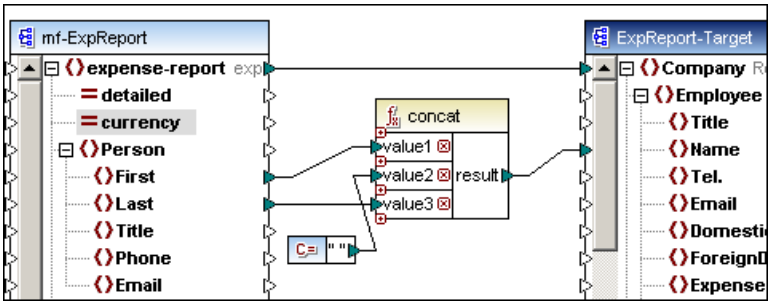
value から得られた最初の文字を整数値のUnicode へ変換した結果が返されます。



7.6.10.3 concat

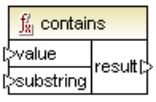
2 個以上の値を連結 (追加) して、result のパラメーターから出力します。全ての入力値は自動的に文字列型へ変換されます。





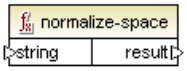
7.6.10.4 contains

substring パラメーターから得られた値が、value パラメーターの値に含まれる場合、true を返します。



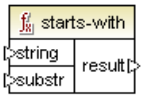
7.6.10.5 normalize-space

正規化された入力文字列が返されます。先頭と末尾にあるスペースが削除され、連続したスペースが 1 つのスペース文字に置き換えられます。Unicode 文字の空白スペースには (U+0020) が使用されます。



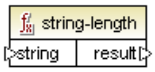
7.6.10.6 starts-with

string 入力パラメーターから得られた文字列が、**substr** パラメーターから得られた文字列で始まる場合、true を返し、それ以外の場合には false を返します。



7.6.10.7 string-length

string パラメーターにて得られた文字列の文字数を返します。



7.6.10.8 substring

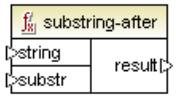
string パラメーターから得られた文字列の一部分を返します。"start" パラメーターにより先頭文字の位置が指定され、"length" パラメーターにより、文字の長さが指定されます。

length パラメーターが指定されていない場合、start パラメーターにより指定された位置から、string パラメーターにより得られた文字列の最後までが返されます。インデックスは 1 から開始されます。

例：substring("56789",2,3) の結果は 678 となります。

7.6.10.9 substring-after

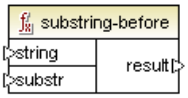
"**string**" パラメーターから得られた値の先頭から、**substr** パラメーターの値が、最初に出現するまでの箇所が削除されます。残りの文字列が関数の戻り値となります。**substr** が **string** 内で出現しなかった場合、空の文字列が返されます。



例 :substring-after("2009/01/04","/") の結果は、最初に出現する "/" より後の "01/04" となります。

7.6.10.10 substring-before

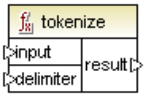
"**string**" パラメーターの先頭から、substr により指定された文字が出現するまでのフラグメントが返されます。**substr** が **string** 内で出現しなかった場合、空の文字列が返されます。



例：substring-after("2009/01/04","/") の結果は、最初に出現する "/" より後の "01/04" となります。

7.6.10.11 tokenize

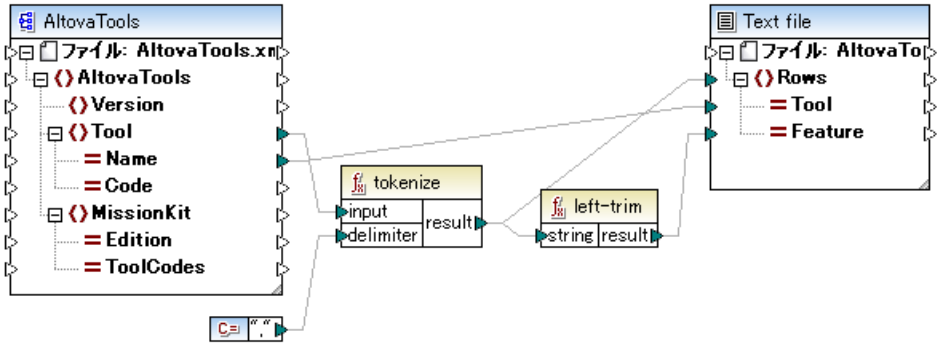
input パラメーターから入力文字列を delimiter パラメーターにより定義されたセグメントのシーケンスに分割します。関数の結果は更なる処理にて使用されます。



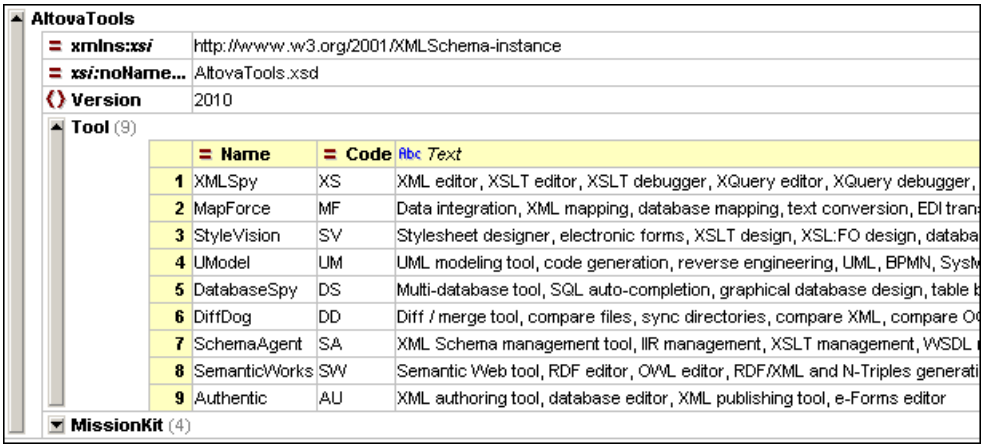
例：input の文字列が "A,B,C" で、delimiter から ","が与えられている場合、A B C がシーケンスとして出力されます。

サンプル

..\MapForceExamples フォルダーに収められている**TokenizeString1.mfd** では tokenize 関数の使用方法を確認することができます。



XML ソースファイルを以下に示します。**Tool** 要素には Name と Code という2つの属性が含まれており、Tool 要素のコンテンツにはコンマにより分けられたデータが与えられています。



AltovaTools

xmlns:xsi	http://www.w3.org/2001/XMLSchema-instance
xsi:noName...	AltovaTools.xsd
Version	2010

Tool (9)

	Name	Code	Text
1	XMLSpy	XS	XML editor, XSLT editor, XSLT debugger, XQuery editor, XQuery debugger,
2	MapForce	MF	Data integration, XML mapping, database mapping, text conversion, EDI tran
3	StyleVision	SV	Stylesheet designer, electronic forms, XSLT design, XSL:FO design, databa
4	UModel	UM	UML modeling tool, code generation, reverse engineering, UML, BPMN, Sysl
5	DatabaseSpy	DS	Multi-database tool, SQL auto-completion, graphical database design, table b
6	DiffDog	DD	Diff / merge tool, compare files, sync directories, compare XML, compare O
7	SchemaAgent	SA	XML Schema management tool, IIR management, XSLT management, WSDL
8	SemanticWorks	SW	Semantic Web tool, RDF editor, OWL editor, RDF/XML and N-Triples generati
9	Authentic	AU	XML authoring tool, database editor, XML publishing tool, e-Forms editor

MissionKit (4)

マッピングの動作：

- tokenize 関数により **Tool** 要素 *アイテム*からデータが与えられ、コンマ ","デリミタにより 入力パラメーターにて与えられた文字列が分割されます 最初の要素は "XML editor" になります)。
- result パラメーターがターゲットコンポーネントの**Row** アイテムへマッピングされているため、分割された各要素ごとに新たな**行**が作成されます。
- result パラメーターは **left-trim** 関数にもマッピングされており、各要素にある先頭の空白スペースが取り除かれます。
- **left-trim** 関数のresult パラメーターは、ターゲットコンポーネントのFeature アイテムへマッピングされます。
- ターゲットコンポーネントとなっている出力ファイルはCSV ファイル (AltovaToolFeatures.csv) として定義されており、各フィールドはセミコロンにより区切られます (コンポーネントをダブルクリックすることで設定を確認することができます)。

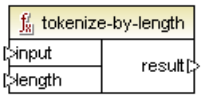
マッピングの結果：

- ソースファイルの各 Tool 要素のため行われます。
- (Tool) 名がターゲットコンポーネント内のTool アイテムにマップされます。
- Tool コンテンツのトークン化された各チャンクが (Tool Name) Feature アイテムに追加さ
- 例 :最初のツールであるXMLSpy には、分割された最初の要素である "XML editor" が加えられます。
- このような動作が現在のTool と全てのTools に対して、繰り返されます。
- 出力タブをクリックすることで、以下にある結果を確認することができます。

```
 1  Tool;Feature
 2  XMLSpy;XML editor
 3  XMLSpy;XSLT editor
 4  XMLSpy;XSLT debugger
 5  XMLSpy;XQuery editor
 6  XMLSpy;XQuery debugger
 7  XMLSpy;XML Schema / DTD editor
 8  XMLSpy;WSDL editor
 9  XMLSpy;SOAP debugger
10  MapForce;Data integration
11  MapForce;XML mapping
12  MapForce;database mapping
```

7.6.10.12 tokenize-by-length

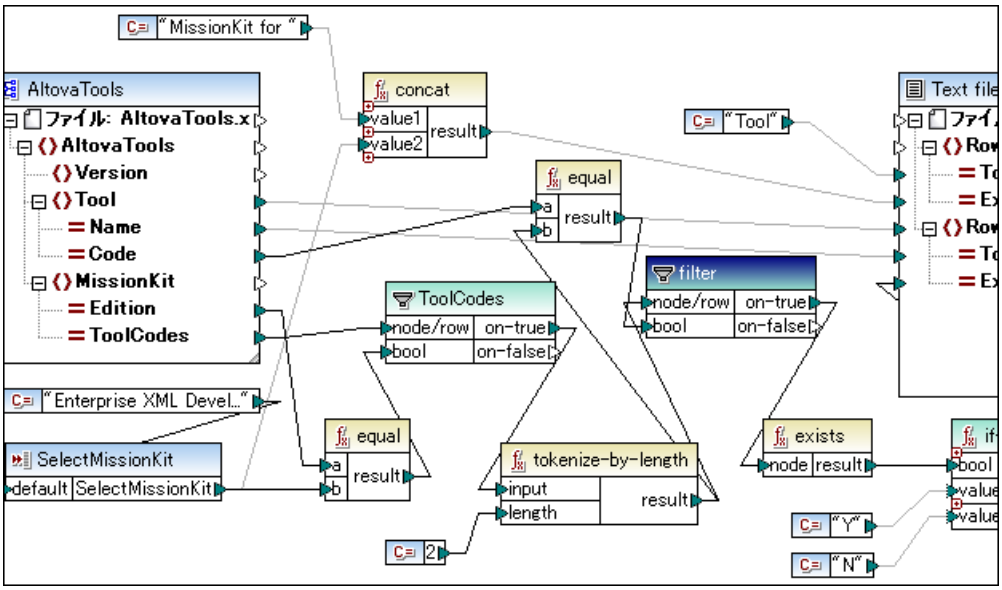
input パラメーターから得られた文字列をlength パラメーターにより定義されたセグメントのシーケンスに分割します。関数の結果は更なる処理にて使用されます。



例 :input の文字列が "ABCDEF" で、length の値が 2 となっている場合、AB CD EF が出力されます。

サンプル

..\MapForceExamples フォルダーに収められている**TokenizeString2.mfd** にて、**tokenize-by-length** 関数の使用方法を確認することができます。



tokenize 関数の例で使用したものと同じXML ソースファイルを以下に示します。**MissionKit** 要素にも**Edition** と **ToolCodes** という 2 つの属性が与えられていますが、要素のコンテンツは無い点に注目してください。

Tool (9)

	Name	Code	Text
1	XMLSpy	XS	XML editor, XSLT editor, XSLT debugger, XQuery editor, XQuery debugger, XML S
2	MapForce	MF	Data integration, XML mapping, database mapping, text conversion, EDI translator,
3	StyleVision	SV	Stylesheet designer, electronic forms, XSLT design, XSL:FO design, database rep
4	UModel	UM	UML modeling tool, code generation, reverse engineering, UML, BPMN, SysML, prc
5	DatabaseSpy	DS	Multi-database tool, SQL auto-completion, graphical database design, table brows
6	DiffDog	DD	Diff / merge tool, compare files, sync directories, compare XML, compare OOXML,
7	SchemaAgent	SA	XML Schema management tool, IIR management, XSLT management, WSDL manag
8	SemanticWorks	SW	Semantic Web tool, RDF editor, OWL editor, RDF/XML and N-Triples generation and
9	Authentic	AU	XML authoring tool, database editor, XML publishing tool, e-Forms editor

MissionKit (4)

	Edition	ToolCodes
1	Enterprise Software Architects	XSMFSVUMDSDDSASW
2	Professional Software Architects	XSMFSVUMDS
3	Enterprise XML Developers	XSMFSVDDSASW
4	Professional XML Developers	XSMFSV

このマッピングの目的：
どのAltova 製品が、MissionKit の各エディションに含まれているかを表すリストを生成します。

マッピングの動作：

- SelectMissionKit **入力**コンポーネントには定数コンポーネントからデフォルト値（この場合は "Enterprise XML Developers"）が与えられます。
- **equal** 関数により、入力値と**Edition** コンポーネントの値が比較され、その結果がToolCodes フィルターの**bool** パラメーターへ渡されます。
- ToolCodes フィルターの**node/row** 入力がソースコンポーネントの**ToolCodes** アイテムから接続されています。例えばEnterprise XML Developers エディションの値はXSMFSVDDSASW となります。
- XSMFSVDDSASW という値が**on-true** パラメーターから、**tokenize-by-length** 関数の**input** パラメーターへ渡されます。

tokenize-by-length 関数の動作：

- ToolCodes から得られたXSMFSVDDSASW という**入力**値が、**length** パラメーターに接続されている 2 という値から、2文字ごとに複数の要素へ分割されます。従って合計 6個の要素が生成されます。
- 各要素はequal 関数のb パラメーターへ渡され、ソースファイルのCode に記述されている 2文字の**Code** 値と比較されます（Code には合計 9個のエントリー*アイテム*が与えられています）。
- 比較の結果がフィルター（filter）の**bool** パラメーターへ渡されます。
- tokenize-by-length 関数から得られた**全て**の要素がフィルター（filter）の**node/row** パラメーターへ渡されている点に注意してください。

- **exists** 関数により、フィルターコンポーネントのon-true パラメーターから得られた値が存在する/存在しないかがチェックされます。

  存在するノードとは、Code の値と**ToolCodes** から得られた要素間でマッチするノードのことです。

  存在しないノードとはCode の値にマッチする**ToolCodes** の要素間でマッチしなかったノードのことです。

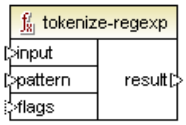
- **exists** 関数の結果は**if-else** 関数へ渡され、ノードが存在する場合にはY が、存在しない場合にはN がターゲットへ渡されます。

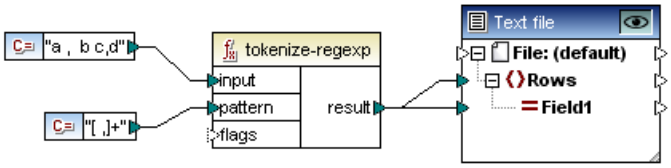
マッピング結果を以下に示します：

```
1   Tool;MissionKit for Enterprise XML Developers
2   XMLSpy;Y
3   MapForce;Y
4   StyleVision;Y
5   UModel;N
6   DatabaseSpy;N
7   DiffDog;Y
8   SchemaAgent;Y
9   SemanticWorks;Y
10  Authentic;N
11
```

7.6.10.13 tokenize-regexp

input から得られた文字列を、文字列（string）のシーケンスに変換します。**pattern** パラメーターにて得られた正規表現のパターンにマッチする文字列がセパレーターとして使用されます。セパレーターとして使用された文字列は関数の出力に表示されません。オプションのflags を使用することもできます。





上に示される例では：
空白スペースとコンマで区切られた文字列（a , b, c,d) が**input** パラメーターから与えられています。

pattern に接続されている正規表現は[,] (["space""comma"]) となっており、空白スペースとコンマのどちらか一方がマッチする文字がセパレーターとして使用されます。

+ 文字を使用することで、「1つ以上の」連続した文字クラス文字列の出現にマッチすることができます。

返される結果を以下に示します：

1	a
2	b
3	c
4	d
5	

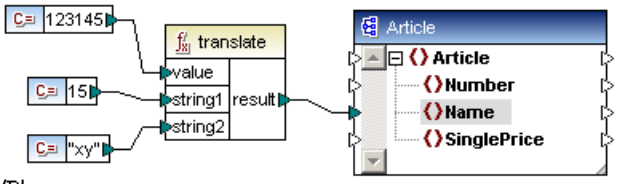
言語によって正規表現には細かな違いがある点に注意してください。C++ のTokenize-regexp はVisual Studio 2008 SP1 以降でしか扱えない点に注意してください。

正規表現に関する更に詳しい情報については正規表現のセクションを参照ください。

7.6.10.14 translate

string1（検索文字列）により得られた文字と同じ位置にある**string2**（置き換え文字列）の内容を使って "**value**" パラメーターの文字列を置き換えます。

string2 に対応する文字が無い場合、その文字は削除されることになります。



例：
入力文字列が "123145"で、
検索文字列）string1 ：15
置換文字列）string2 ：xy

となっている場合、
value パラメーターから得られた文字列にある "**1**" が**x** に置き換えられ、
value パラメーターから得られた文字列にある "**5**" が**y** に置き換えられます。

結果として得られる文字列は**x23x4y** となります。

string2 が空の場合（string1 よりも短い文字列の場合）、対応する文字列は削除されます。

例 2：
value から得られた入力文字列が "aabaacbca" で、
検索文字列）string1 は、 "a" です。
置換文字列）string2 は、 "" （空の文字列）です。

結果として "bcbc" が得られます。

例 3：
入力文字列が "aabaacbca" で、
string1 は、 "ac" です。
string2 は、 "ca" です。

結果として "cbccabac" が得られます。

7.6.11 xpath2 | accessors

XPath2 関数は XSLT2 または XQuery 言語が選択されている状態で利用することができます。

7.6.11.1 base-uri

base-uri 関数はパラメーターにノードを受け取り、そのノードが含まれる XML リソースの URI を返します。出力の型は xs:string になります。入力ノードが与えられていない場合、MapForce によりエラーが表示されます。

7.6.11.2 node-name

node-name 関数は入力パラメーターとしてノードを受け取り、その QName を返します。QName が文字列として表示され、ノードにプレフィックスがある場合は prefix:localname 形式の入力を受け取り、ノードにプレフィックスが無い場合は、localname 形式の入力が受け取られます。ノードの名前空間 URI を取得するには、（name-に関連した functions 以下にある）namespace-URI-from-QName 関数を使用してください。

7.6.11.3 string

`string` 関数は、`xs:string` のように作動し、入力された値を `xs:string` 型に変換します。

入力が（例えば `xs:decimal` の）原子型である場合、原子型の値が `xs:string` 型の値に変換されます。入力パラメーターがノードの場合、ノードの文字列値が抽出されます（ノードの文字列値とはノードの子孫ノードから得られた値を連結したものになります）。

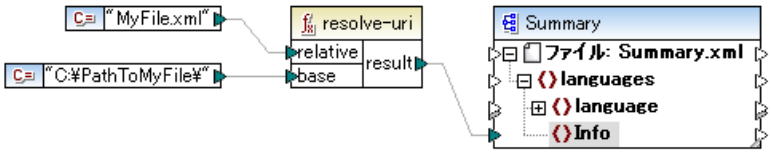
7.6.12 xpath2 | anyURI functions

XPath2 関数は XSLT2 または XQuery 言語が選択されている状態で利用することができます。

7.6.12.1 resolve-uri

revolve-uri 関数は最初のパラメーター（relative）として URI（データ型：xs:string）を受け取り、2番目のパラメーター（base）から得られたベース URI（データ型：xs:string）に対して URI を解決します。

結果 (データ型 xs:string) は結合されたURI です。この方法で、相対 URI (最初の引数)は、ベースURI に対して解決されることで、絶対 URI に変換されます。



上のスクリーンショットでは、最初のパラメーターにより相対 URI が与えられ、 2番目のパラメーターによりベースURI が与えられます。解決されたURI はベースURI と相対 URI が連結され、C:\PathToMyFile\MyFile.xml が得られます。

メモ: パラメーターは両方ともxs:string 型で、両方の入力を文字列として扱うことで処理は行われます。そのため、解決されたURI のリソースが実際に存在するかどうかはチェックされません。 2番目のパラメーターが与えられていない場合、MapForce によりエラーが表示されます。

7.6.13 xpath2 | boolean functions

XPath2 関数はXSLT2 またはXQuery 言語が選択されている状態で利用することができます。
boolean 関数として収められているtrue ならびにfalse 関数はパラメーターを受け取ること無く、boolean の定数値であるtrue とfalse をそれぞれ返します。定数のboolean 値が必要な場所で使用することができます。

7.6.13.1 false

boolean 値の "false" を挿入します。

7.6.13.2 true

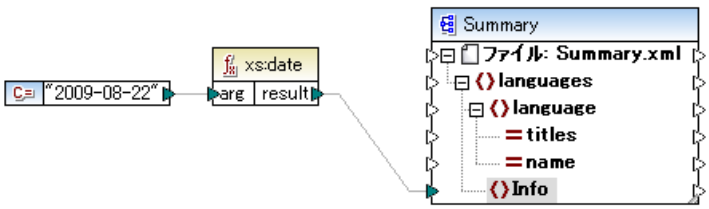
boolean 値の "true" を挿入します。

7.6.14 xpath2 | constructors

XPath2 関数はXSLT2 またはXQuery 言語が選択されている状態で利用することができます。

XPath 2.0 関数ライブラリのconstructors 以下にある関数を使用することで、入力テキストから特定のデータ型を構築することができます。通常、入力されたテキストは構築されるデータ型に対応した書式で記述されている必要があり、そうでない場合は変換に失敗します。

例えば、xs:date データ型を構築したい場合、constructors 以下にあるxs:date 関数を使用します。入力テキストはxs:date データ型の書式であるYYYY-MM-DD の形式で記述されている必要があります (以下のスクリーンショットを参照ください)。



上のスクリーンショットでは、文字列定数 (2009-08-22) が関数への入力パラメーターとして使用されています。入力はソースドキュメント内のノードからも得ることができます。

xs:date 関数により、xs:string 型で与えられた入力テキストを値とする (関数により指定されている) xs:date データ型が出力されます。

関数ボックスの入力パラメーター上にマウスを移動させると、パラメーターのデータ型がポップアップで表示されます。

7.6.15 xpath2 | context functions

XPath2 関数は XSLT2 または XQuery 言語が選択されている状態で利用することができます。

Context 関数ライブラリには現在の日付や時刻、プロセッサーにより使用されているデフォルトの照合、現在のシーケンスの大きさや現在のノードの位置といった情報を得るための関数が用意されています。

7.6.15.1 current-date

システムクロックから得られた現在の日付 (`xs:date`) を返します。

7.6.15.2 current-dateTime

システムクロックから得られた現在の日時 (`xs:dateTime`) を返します。

7.6.15.3 current-time

システムクロックから得られた現在の時刻 (`xs:time`) を返します。

7.6.15.4 default-collation

default-collation 関数は入力パラメーターを取らず、デフォルトの照合、つまり指定することができる関数に対して特定の照合が指定されなかった場合に使用される照合を返します。

Altova XSLT 2.0 エンジンでは Unicode のコードポイント照合だけがサポートされます。fn:max や fn:min 関数といった比較機能は、この照合をベースに行われます。

7.6.15.5 implicit-timezone

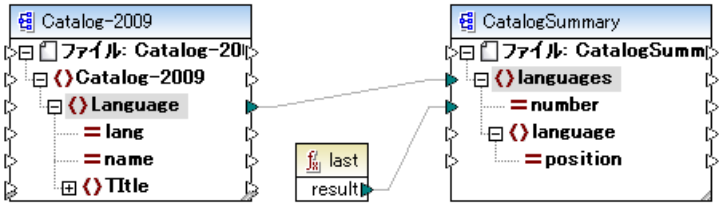
評価コンテキストからの "implicit timezone" プロパティの値を返します。

7.6.15.6 last

last ならびに position 関数は入力パラメーターを取りません。last 関数により、コンテキストノードセット内の最後にあるノードの場所が返されます。position 関数は、ノードセット内で現在処理されているノードの位置を返します。

関数が接続されているノードのコンテキストノードセットが、関数が適用されるノードセットとなります。以下にあるスクリーン

ショットでは、language 要素が last ならびに position 関数のコンテキストノードセットとなります。



上の例では、last 関数によりコンテキストノードセット (Language 要素のノードセット) における最後のノード位置が、number 属性の値へ返されます。この値はそのままノードセット内にあるノードの数となり、ノードセットの大きさを表す値にもなります。

position 関数により、処理されている Language ノードの位置が返されます。各 Language 要素ノードに対して、Language 要素のノードセット内における位置が language/@position 属性ノードへ出力されます。

core 関数にある**position** ならびに**count** 関数の使用が推奨されます。

7.6.16 xpath2 | durations, date and time functions

XPath2 関数は XSLT2 または XQuery 言語が選択されている状態で利用することができます。

XPath 2 duration, date, and time functions 内にある関数を使用することで、タイムゾーンの日時を調整したり、date-time データから特定のコンポーネントを抽出、複数ある日時の値から期間を取得することができます。

'Adjust-to-Timezone' 関数

これら関数はそれぞれ date、time、または dateTime データ型の値を最初の入力パラメーターとして受け取り、2番目のパラメーターから得られた値により、タイムゾーンの削除や編集を行います。

最初のパラメーターにタイムゾーン情報が含まれていない (例えば 2009-01 という日付や 14:00:00 という時刻) 場合、adjust-to-timezone 関数を使った以下のような場合が考えられます：

- (2番目の入力パラメーターである) timezone パラメーターが与えられている場合、2番目のパラメーターから得られた値がタイムゾーンとして使用されます。2番目のパラメーターから得られたタイムゾーンが追加されます。
- timezone パラメーターが与えられていない場合、システムのタイムゾーンである暗黙的なタイムゾーンが結果に含まれます。システムのタイムゾーンが含まれます。
- timezone パラメーターが空の場合、結果にタイムゾーンは含まれません。

最初のパラメーターにタイムゾーン情報 (例えば 2009-01-01+01:00 という日付や 14:00:00+01:00 という時刻) が含まれている場合、adjust-to-timezone 関数を使った以下のような場合が考えられます：

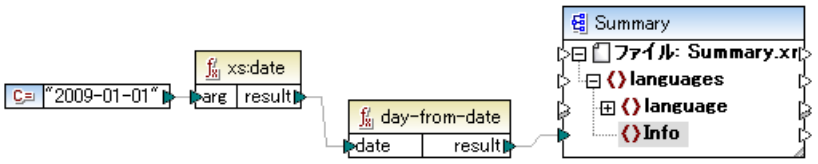
- (2番目の入力パラメーターである) timezone パラメーターが与えられている場合、2番目のパラメーターから得られた値がタイムゾーンとして使用されます。オリジナルのタイムゾーンが、2番目のパラメーターから得られたタイムゾーンに置き換えられます。
- timezone パラメーターが与えられていない場合、システムのタイムゾーンである暗黙的なタイムゾーンが結果に含まれます。オリジナルのタイムゾーンがシステムのタイムゾーンに置き換えられます。
- timezone パラメーターが空の場合、結果にタイムゾーンは含まれません。

'From' 関数

これらfrom 関数は、(i) 日付や時刻のデータ または (ii) 期間を表すデータから特定のコンポーネントを抽出するのに使用されます。xs:decimal データ型の結果が返されます。

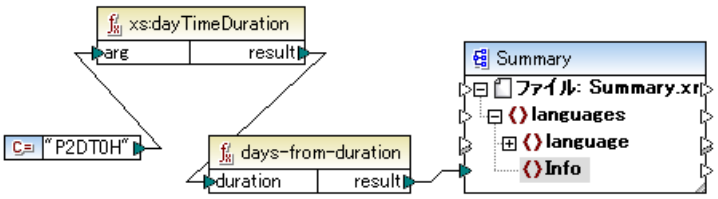
日付または時刻のデータから、コンポーネントを抽出する例として、以下のスクリーンショットにあるようなマッピングを使用しま

す。



入力パラメーターにはxs:date 型の日付 (2009-01-01)が与えられています。day-from-date 関数により、xs:decimal データ型の日付コンポーネントが抽出されます。

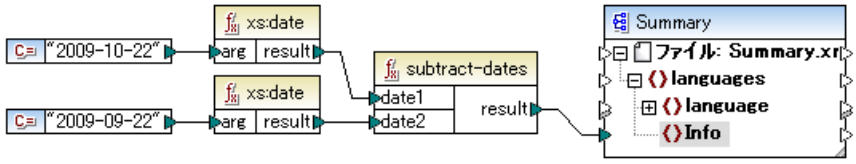
期間型のデータから時刻コンポーネントを抽出する場合、 年と月の抽出するには、期間をxs:yearMonthDuration または 日、時、分、秒の抽出には、 xs:dayTimeDuration として指定する必要があります。xs:decimal 型の結果が得られます。P2DT0H というxs:dayTimeDuration 型の入力をdays-from-duration 関数へ与えているマッピング例を以下のスクリーンショットに示します。結果はxs:decimal 型の 2 となります。



'Subtract' 関数

用意されている 3個のsubtraction 関数により、時間に関する値を他の値から差し引き、期間の値を得ることができます。subtract-dates、subtract-times、そしてsubtract-dateTimes という関数が用意されています。

subtract-dates 関数を使用することで、 2つの日付を差し引いた期間 (2009-10-22 - 2009-09-22) を求める例を、以下のスクリーンショットにて示します。xs:dayTimeDuration 型のP30D が結果として得られます。

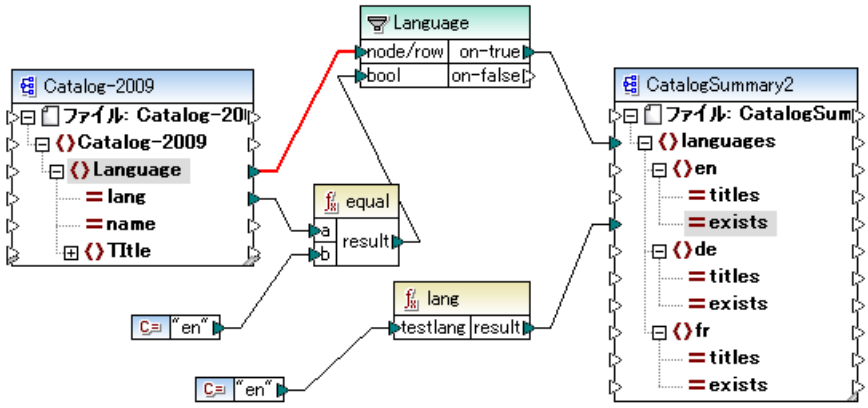


7.6.17 xpath2 | node functions

以下のnode 関数が用意されています：

lang

lang 関数は、例えばen といった言語コードが書かれた文字列を受け取ります。コンテキストノードにxml:lang 属性が存在しており、属性値が関数のパラメーターで指定された値にマッチするかがチェックされ、true または false が返されます。



上のスクリーンショットについて、以下の点に注意してください：

1. ソーススキーマ内にあるLanguage 要素以下には、xml:lang 属性があります。
2. Language 要素がフィルタリングされ、en という値を持った xml:lang 型の要素だけが処理されます（フィルタリングを行うための条件は equal 関数により与えられます）。
3. Language ノードがコンテキストノードとなり、出力ドキュメントではen 要素が作成されます。
4. lang 関数の出力（true または false）が、出力コンポーネントのen/@exists 属性へ渡されます。関数のパラメーターにはen という定数の文字列が与えられます。lang 関数により、コンテキストノード（Language 要素）に e n という（関数のパラメーターから得られた）値を持ったxml:lang 属性があるかどうかがチェックされます。もしそのような属性値が存在する場合、true が返され、それ以外の場合はfalse が返されます。

local-name, name, namespace-uri

local-name、name、namespace-uri 関数により、それぞれローカル名、名前、そして入力ノードの名前空間 URI が返されます。例えば、altova:Products というノードがある場合、Products がローカル名になり、名前が altova:Products、そして名前空間 URI は、altova: というプレフィックスに対応した名前空間のURI （例：http://www.altova.com/examples）となります。

これらの関数には、それぞれ 2 つの派生があります：

- 入力パラメーターを持たない場合：コンテキストノードに対して関数が適用されます（コンテキストノードのサンプルについては、上にあるlang 関数の例を参照ください）。
- node 入力パラメーターを持つ場合：与えられたノードに対して関数が適用されます。

これら関数の出力はstring 型の文字列となります。

number

入力文字列を数値へ変換します。boolean 型の値も数値へ変換されます。

number 関数はノードを入力として受け取り、ノードを原子化（コンテンツの抽出）して、その値を数値へ変換し、変換された値を返します。変換することのできる値のデータ型はboolean、string、その他の数値型だけになります。数値型ではない値（例えば数値では無い文字列）が入力値の場合、NaN (Not a Number) という値が返されます。

number 関数には二種類あります：

- 入力パラメーターを持たない場合：コンテキストノードに対して関数が適用されます（コンテキストノードのサンプルについては、上にあるlang 関数の例を参照ください）。
- node 入力パラメーターを持つ場合：与えられたノードに対して関数が適用されます。

7.6.18 xpath2 | numeric functions

以下のnumeric 関数が用意されています：

abs

abs 関数は数値型の値を受け取り、その値の絶対値を返します。例えば、入力パラメーターから -2 または +2 が与えられた場合、2 という値が返されます。

round-half-to-even

round-half-to-even 関数により、最初のパラメーターから与えられた値を（2番目のパラメーターから与えられた）オプションの数値精度で四捨五入します。例えば、最初のパラメーターに 2.141567 という値が与えられており、2番目パラメーターに 3 という数値が与えられている場合、最初のパラメーター値が小数点第 3位で四捨五入され、2.141 が返されます。精度（2番目パラメーター）が指定されていない場合、精度に 0 が指定され、与えられた値は整数（nteger）になります。

関数に éven' という名前が付けられている通り、値がちょうど中間に位置している場合、偶数が採用されます。例えば、round-half-to-even(3.475, 2) は 3.48 という値を返します。

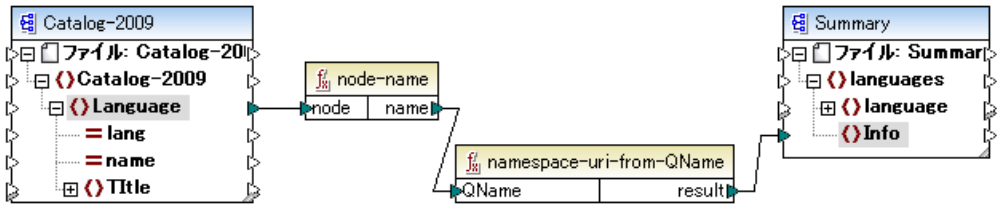
7.6.19 xpath2 | qname- related functions

QName に関連した関数には、local-name-from-QName とnamespace-uri-from-QName という関数が用意されています。

関数は両方とも（文字列形式の）拡張 QName を入力パラメーターとして受け取り、拡張 QName のローカル名と名前空間 URI がそれぞれ返されます。

両方の関数において入力は文字列として処理されるため、これらの関数の入力パラメーターに対して、ノードを直接接続することはできない点に注意してください。

ノードをnode-name 関数へまず接続し、拡張 QName の出力を得ます。node-name 関数の出力から得られた拡張 QName を、これら関数の入力パラメーターへ与えます（以下のスクリーンショットを参照ください）。



これら関数の出力結果はいずれも文字列型になります。

7.6.20 xpath2 | string functions

以下にあるXPath string 関数が用意されています：

compare

compre 関数は、入力パラメーターから与えられた 2つの値がアルファベットとして同一の値かを比較します。string1 の値がアルファベットとしてstring2 のそれよりも小さい場合（例えばA とB という文字列が与えられた場合）、-1 という値が関数から返されます。2つの文字列が同一の場合（例えばA とA）、0 が返されます。string1 の値がstring2 よりも大きい場合（例えばB とA）、+1 が返されます。

この関数の派生型を使用すると、文字列の比較に照合 (collation) を使用することができます。collation パラメーターに値が接続されていない場合、デフォルトの照合である Unicode のコードポイント照合が使用されます。Altova エンジンでは Unicode コードポイント照合だけが利用可能です。

ends-with

ends-with 関数は、string の終端が substring と等しいかをチェックして、等しい場合は true を、そうでない場合は false を返します。

この関数の派生型を使用すると、文字列の比較に照合 (collation) を使用することができます。collation パラメーターに値が接続されていない場合、デフォルトの照合である Unicode のコードポイント照合が使用されます。Altova エンジンでは Unicode コードポイント照合だけが利用可能です。

escape-uri

`escape-uri` 関数は、URI を最初の文字列の引数を入力として取り、文字列に対しての RFC 2396 の URI エスケープ規約に適用します。URI 内で予約済みの文字がエスケープされる場合、第 2 のブール引数 (`escape-reserved`) は `true()` に設定されます。 (例 "+" または "/")。

例 :

`escape-uri("My A+B.doc", true())` would give `My%20A%2B.doc`
`escape-uri("My A+B.doc", false())` would give `My%20A+B.doc`

lower-case

matches 関数により、(input パラメーターにより) 与えられた文字列が、(pattern パラメーターから得られる) 正規表現にマッチするかがチェックされます。正規表現の構文は、XML スキーマの pattern ファセットにて定義する必要があります。文字列が正規表現にマッチすれば true が返され、それ以外の場合は false が返されます。

matches

matches 関数により、(input パラメーターにより) 与えられた文字列が、(pattern パラメーターから得られる) 正規表現にマッチするかがチェックされます。正規表現の構文は、XML スキーマの pattern ファセットにて定義する必要があります。文字列が正規表現にマッチすれば true が返され、それ以外の場合は false が返されます。

オプションの flags パラメーターを関数に与えることもできます。 4 つのフラグ (i、m、s、x) を指定することができます。例えば imx のように、複数のフラグを使用することもできます。フラグが使用されていない場合、 4 つのフラグのデフォルト値が使用されます。

4種類あるフラグの説明を以下に示します :

- `i` 大文字と小文字の違いを無視します。デフォルトでは大文字と小文字は異なるものとして扱われます。
- `m` 複数行モードを使用して、改行文字 (0xa) により複数行に分けられている入力文字列を、1 つの文字列として扱います。メタ文字の ^ と $ が、それぞれ行頭と行末を表します。デフォルトは文字列モードになっており、個々の文字列が ^ から始まり $ で終わります。
- `s` ドット (.) の適用範囲が拡張されます。メタ文字のドット "." はデフォルトで改行文字を除く全ての文字にマッチしますが、このオプションにより、改行文字もマッチするようになります
- `x` 空白スペースを無視します。デフォルトでは、空白スペースは無視されません。

normalize-unicode
normalize-unicode 関数を使用することで、string パラメーターから得られた入力文字列を、normalizationForm パラメーターから得られたルールに従うかたちで正規化します。正規化のフォームとしてNFC、NFD、NFKC、そしてNFKDがサポートされています。

replace
replace 関数では、input パラメーターから得られた文字列に対して、pattern パラメーターで指定された正規表現を適用し、マッチした箇所をreplacement パラメーターの値により置き換えます。

マッチングに使用されるルールについては、上にあるmatches 関数の記述を参照ください。この関数にはオプションとしてflag パラメーターを与えることもできます。flag にて指定できる内容については、matches 関数の記述を参照ください。

starts-with
stars-with 関数は、string の始端がsubstring と等しいかをチェックして、等しい場合はtrue を、そうでない場合はfalse を返します。

この関数の派生型を使用すると、文字列の比較に照合 (collation) を使用することができます。collation パラメーターに値が接続されていない場合、デフォルトの照合であるUnicode のコードポイント照合が使用されます。Altova エンジンではUnicodeコードポイント照合だけが利用可能です。

substring-after
substring-after 関数では、（2番目のパラメーターである）arg2 にて得られた値が（最初のパラメーターである）arg1 から得られた文字列に対してマッチされ、マッチした部分より後ろの文字列が返されます。オプションである 3番目のパラメーターを使用することで、文字列の比較に使用する照合を指定することができます。collation パラメーターに値が接続されていない場合、デフォルトの照合であるUnicode のコードポイント照合が使用されます。Altova エンジンではUnicodeコードポイント照合だけが利用可能です。

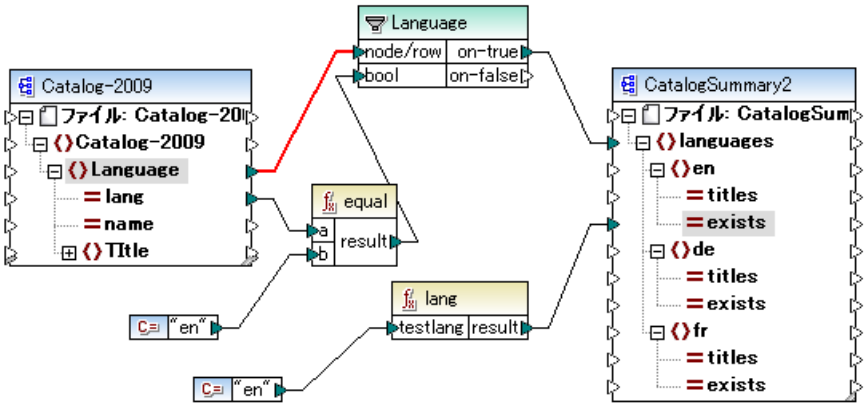
substring-before
substring-before 関数では、（2番目のパラメーターである）arg2 にて得られた値が、（最初のパラメーターである）arg1 から得られた文字列に対してマッチされ、マッチした部分までの文字列が返されます。オプションである 3番目のパラメーターを使用することで、文字列の比較に使用する照合を指定することができます。collation パラメーターに値が接続されていない場合、デフォルトの照合であるUnicode のコードポイント照合が使用されます。Altova エンジンではUnicodeコードポイント照合だけが利用可能です。

upper-case
upper-case 関数は、入力パラメーターから受け取った文字列の中にある小文字を、大文字に変換します。

7.6.21 xslt | xpath functions

xpath functions ライブラリに収められている関数は、XPath 1.0 ノードセットの関数です。各関数は、ノードやノードセットをコンテキストとして受け取り、そのノードやノードセットに関する情報を返します。これらの関数は、通常以下のような要素を備えています：

- コンテキストノード：以下のスクリーンショットでは、ソーススキーマにあるLanguage 要素がlang 関数のコンテキストノードとして与えられています。
- 入力パラメーター：以下のスクリーンショットでは、文字列定数のen がlang 関数の入力パラメーターとして与えられています。last ならびにposition 関数は入力パラメーターを取りません。



lang

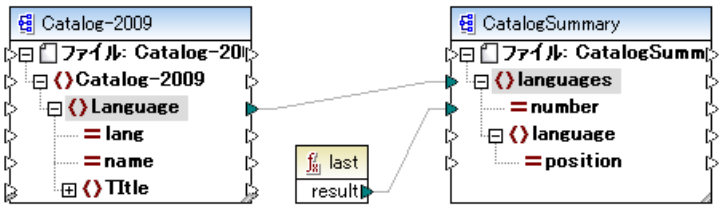
lang 関数は (例えば en という) 言語コードが書かれた文字列を受け取ります。コンテキストノードに xml:lang 属性が存在しており、属性値が関数のパラメーターで指定された値にマッチするかがチェックされ、true または false が返されます。上のスクリーンショットにおいて、以下の点に注目してください:

1. ソーススキーマにある Language 要素には xml:lang 属性が含まれています。
2. `Language` ノードはフィルタリングされており、`en` という値の `xml:lang` 属性を持った `Language` ノードだけが処理されます (フィルタリングの条件が `equal` 関数により与えられます)。
3. Language ノードがコンテキストノードとなり、出力ドキュメントにて en 要素が作成されます。
4. lang 関数の出力 (true または false) が出力ファイルの en/@exists 属性ノードへ渡されます。定数の en が関数のパラメーターへ渡されています。lang 関数はコンテキストノード (Language 要素) に xml:lang 属性があり、その値が (関数のパラメーターにより与えられた) en であるかチェックして、そうである場合は true を、そうでない場合は false を返します。

last, position

last ならびに position 関数は入力パラメーターを受け取りません。last 関数では、コンテキストノードセット内にある最後のノードの位置が返されます。position 関数により、処理中のノードセット内にある現在のノード位置が返されます。

関数が接続されているノードにあるコンテキストノードセットは、関数が適用されるノードセットとなります。以下のスクリーンショットにおいて、last 関数のコンテキストノードセットは Language 要素になります。



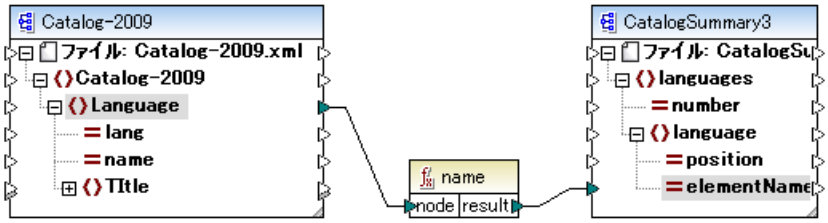
上の例では last 関数により、コンテキストノードセット (Language 要素のノードセット) にある最後のノードの位置が、number 属性の値へ返されます。この関数はノードセット内にあるノードの数を返すため、ノードセットの大きさを示す値が返されます。

position 関数は、Language ノードにおいて現在処理されているノードの位置を返します。Language 要素のノードセット内にて処理されているノードの位置が language/@position 属性ノードへ返されます。

name, local-name, namespace-uri

これらの関数は同じように使用され、それぞれ入力ノードの名前、ローカル名、名前空間 URI を返します。以下にあるスクリーンショットにて、これら関数の使用方法を確認することができます。コンテキストノードが指定されていない点に注目してください。

name 関数は、Language ノードの名前をlanguage/@elementname 属性に対して返しています。これら関数のパラメーターが、単一のノードではなく、ノードセットの場合、そのノードセットにある最初のノードの名前（またはローカル名や名前空間 URI）が返されます。



name 関数は、ノードの QName を返します。local-name 関数により、ノード QName のローカル名部分が返されます。例えば、ノードの QName が altova:MyNode となっている場合、MyNode がローカル名になります。

名前空間 URI とは、ノードが属している名前空間の URI のことです。例えば以下のような方法で、altova: というプレフィックスを、名前空間 URI へマッピングすることができます：xmlns:altova="http://www.altova.com/namespaces"。

メモ： その他の XPath 1.0 関数を core 関数 ライブラリ以下で確認することもできます。

7.6.22 xslt | xslt functions

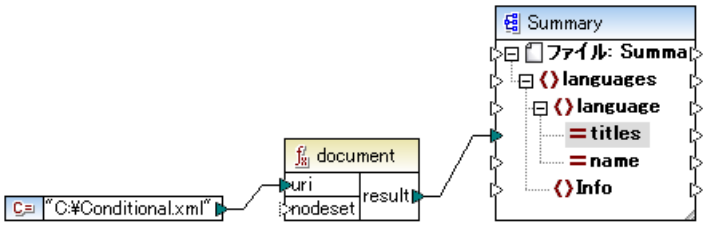
xslt functions ライブラリ内には、以下に記される XSLT 1.0 関数が収められています。

7.6.22.1 currrent

current 関数は入力パラメーターを取らず、現在のノードを返します。

7.6.22.2 document

document 関数により、uri パラメーターに接続した外部 XML ドキュメントを開くことができます（以下のスクリーンショットを参照）。uri パラメーターで与えられた URI が相対 URI の場合、URI を解決するためのベース URI を、オプションの nodeset パラメーターから指定することができます。関数の結果は、出力ドキュメント内にあるノードへ返されます。

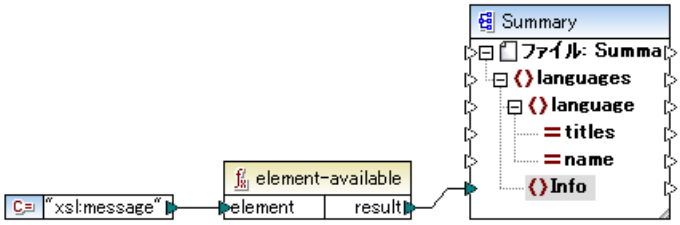


uri パラメーターに与えられる文字列は絶対ファイルパスでなければならない点に注意してください。

7.6.22.3 element-available

element-available 関数により、入力された要素が XSLT の処理系によりサポートされているかチェックすることができます。

パラメーターはQName として評価されます。そのため、XSLT 要素にはxsl: プレフィックスを、XML スキーマ要素にはxs: プレフィックスを付与する必要があります。マッピングで生成されるXSLT 内で使用される名前空間にこれらのプレフィックスが対応しています。



関数からはブール値が返されます。

7.6.22.4 function-available

function-available 関数はelement-available 関数と同様に、パラメーターにより与えられた名前の関数が、XSLT の処理系によりサポートされているかチェックするために使用することができます。

入力文字列はQName として評価されます。関数からはブール値が返されます。

7.6.22.5 generate-id

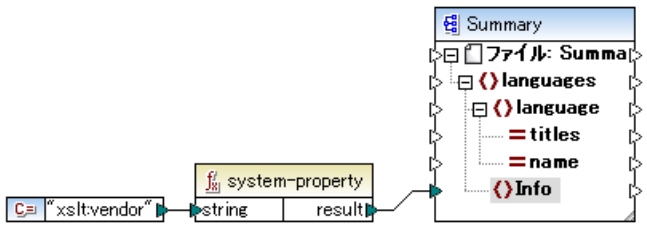
generate-id 関数を使用することで、オプションの入力パラメーターにより与えられたノードセット内における最初のノードを識別するユニークな文字列が生成されます。

入力パラメーターが与えられない場合、コンテキストノード上のID が生成されます。出力ドキュメントにある任意のノードに対して出力結果を接続することができます。

7.6.22.6 system-property

system-property 関数からはXSLT 処理系（システム）のプロパティが返されます。XSLT の処理系にはXSLT 名前空間として書かれたシステムプロパティを指定する必要があります。名前空間はxsl:version、xsl:vendor、そしてxsl:vendor-url となります。

入力された文字列はQName として評価されるため、XSLT スタイルシート内でXSLT 名前空間に関連付けられているxsl:prefix を使用する必要があります。



7.6.22.7 unparsed-enity-uri

DTD を使用している場合、パースされていないエンティティを宣言することができます。パースされていないエンティティ（例えば画像イメージ）は、そのエンティティがある場所を指し示すURI を持つことになります。

関数への入力文字列はDTD 内部にて宣言された（パースされていない）エンティティーにマッチする必要があります。パースされていないエンティティのURI が関数により返され、出力ドキュメント内にあるノード（例えばhref ノード）へ接続することができます。

Chapter 8

MapForce とマッピングの自動化

8 MapForce とマッピングの自動化

このセクションは、のコマンドラインインターフェイスおよびRaptor XML Serverを使用してマッピングを自動化する方法について説明します。

8.1 RaptorXML Server を使用して自動化する

RaptorXML Server (今後は、略してRaptorXML) は、Altova 第3世代の超高速なXML とXBRL のためのプロセッサーです。最新の標準とパラレル コンピューティング環境のために最適化されています。クロスプラットフォームに対応可能で、エンジンは今日のマルチコアコンピューティングを有効に利用し、XML とXBRL データの高速処理を提供します。

RaptorXML には、Altova ダウンロードからダウンロードしてインストールすることのできる複数のエディションがあります :

- RaptorXML Server は、XML、XML スキーマ、XSLT、XPath、XQuery、などへのサポートを搭載した高速なXML 処理エンジンです。このエディションは、FlowForce Server インストール パッケージの一部です。
- RaptorXML+XBRL Server は、XBRL の一連の標準を処理および検証する追加機能を使用して、RaptorXML Server の全ての機能をサポートします。

制約事項 :

- XML 署名はサポートされていません
- COM インターフェイスを介してグローバル リソースはサポートされていません
- ODBC とADO データベース接続はWindows のみでサポートされています。他のオペレーティングシステムでは、JDBC を使用する必要があります。

XSLT 1.0 または 2.0、でコード生成するには、MapForce は、生成の際に選択する出力フォルダーに保存される `DoTransform.bat` と呼ばれるバッチファイルを作成します。バッチファイルの実行は、RaptorXML Server を呼び出し、サーバー上でXSLT/XQuery 変換を実行します。

サーバー上の他の出力のために、MapForce マッピングを実行または自動化するには、Altova MapForce Server と FlowForce Server を参照してください。

メモ: 内蔵のエンジンを使用して、XSLT コードをプレビューすることができます。

8.2 MapForce コマンドラインインターフェイス

一般的なコマンドライン構文 :

MapForce.exe *filename* [*/target* [*outputdir*] *オプション*

- 角かっこ**[...]** 任意のパラメーターを表します。
- 中かっこ**{...}** 複数の選択肢を含むパラメータのグループを表します。
- ***パイプ***シンボル**|** は、OR を表します。例 : /XSLT または/JAVA

コマンドラインの実行に成功した場合、MapForce.exe は終了コード 0 を返します。他の値は失敗を意味します。バッチファイル内のIF ERRORLEVEL コマンドを使用してこのコードをチェックすることができます。

filename
ロードするMFD またはMFP ファイル。**パスまたは、ファイル名にスペースが含まれると、パス /ファイル名を引用符で囲んでください。すなわち、 "c:\ Program Files\ ...\ Filename"**

target

/XSLT	XSLT 1.0 コードを生成します
/XSLT2	XSLT 2.0 コードを生成します

/ **GLOBALRESOURCEFILE** *globalresourcefilename*	指定されたグローバルリソースファイル内で定義されたグローバルリソースを使用します
/ **GLOBALRESOURCECONFIG** *configurationname*	指定されたグローバルリソースの構成を使用します

outputdir
outputdir 内に生成されたマッピングコードが置かれるディレクトリは任意です。出力パスが与えられない場合、作業 /現在のディレクトリが使用されます。指定されていない場合、相対的なファイル名 は、作業中 /現在の ディレクトリに対して相対的です。相対的とはドライブの文字から始まる全部のパスではないファイル名が与えられていることを意味します)。

オプション
複数のオプションの指定 :

/**LOG** *logfilename*	*logfilename* と呼ばれるログファイルを生成します。*Logfilename* はフルパス名であることができます。すなわち、ログファイルのディレクトリとファイル名です。しかし、フルパスが与えられる場合、ディレクトリはログファイルを生成するために存在しなければなりません。 ファイル名を指定すると、ファイルは *outputdir* ディレクトリに置かれます。

例 :

MapForce.exe *filename* は、MapForce を開始し、*filename* により定義されたファイルを開きます。

I) すべてのXSLT ファイル ログファイルを出力します。
MapForce.exe *filename* **/XSLT** *outputdir* **/LOG** *logfilename*

II) すべてのXSLT ファイルを生成し、指定された構成のためのグローバルリソースファイルの全てのグローバルリソースを使用します。

Mapforce.exe *filename* **/XSLT** *outputdir* **/ GLOBALRESOURCEFILE** *globalresourcefilename* **/ GLOBALRESOURCECONFIG** *configurationname*

コマンドラインを実行すると、複数のファイルが作成されます：

- 生成されたXSLT ファイル
- XML ファイルを変換するためにRaptorXML Server を使用する**DoTransform.bat** と呼ばれるバッチファイル。
- コマンドライン内で指定されているログファイル。

生成されたXSLT を使用して XML ファイルを変換する：

1. RaptorXML Server エンジンをRaptorXML ダウンロードページからダウンロードしてインストールします。
2. 既に指定された出力フォルダー内にあるDoTransform.bat バッチファイルを実行します。
   これにより現在のフォルダーに出力ファイルを生成します。

メモ：
RaptorXML がインストールされている場所を環境変数の変数パスに追加する必要があるかもしれません。

Chapter 9

Map のカスタム化

9 Map のカスタム化

このセクションではAltova グローバルリソースとの作業、 およびカタログファイルとの作業方法について説明します。

9.1 MapForce オプションの変更

MapForce 内の一般および他の優先順位を変更する方法は、以下の通りです：

- **「ツール」**メニューから **「オプション」**をクリックします。

使用することのできるオプションは、以下に表示されているとおりです。

ライブラリ

- MapForce にカスタム関数 ライブラリを追加または削除するには以下を参照してください（カスタム XSLT 1.0 または 2.0 関数のインポート）。

全般

- MapForce の開始時にロゴを表示する、または、非表示にするかを指定します。
- マッピング ペイン内の背景のグラデーションでの表示を有効化または無効化します。
- **注釈の表示 ...**は、注釈をサポートするコンポーネントに適用されます（例えば、XML スキーマ、EDI）。注釈のテキストが複数の行を含む場合、このオプションを有効化することにより、コンポーネントの最初の N 行のみが表示されます。N は指定される値です。この設定はコンポーネント内で表示される SELECT ステートメントにも適用されます。
- 新しいコンポーネントのためのデフォルトの文字エンコードを定義します。この設定は各コンポーネントのために個別に変更されることもできます（コンポーネントの設定の変更 を参照）。
- 出力ペイン内でマッピングの結果をプレビューする際の実行のタイムアウトを定義します。
- 出力ペイン内でマッピングの結果をプレビューする際、（デフォルトのオプションである）マッピングの出力が一時的ファイルに書き込まれるか、または、最終ファイルに直接書き込まれるかを指定します。

  警告: "最終出力ファイルに直接書き込む"オプションが有効化されている場合、確認を必要とすることなく、出力ファイルは上書きされます。

- 大きな XML とテキストファイルを生成するマッピングをプレビューする場合、出力ペイン内のテキストの表示の最大値を指定します（出力のプレビュー を参照）。

編集

- マウスでドラッグする際にコンポーネントまたは関数が他のコンポーネントと共に整列するかを指定します（コンポーネントの整列 を参照）。
- 有効化されると、**スマートコンポーネントの削除** オプションは、削除された接続を "記憶" します（コンポーネントの削除後、接続を保持する を参照）。

メッセージ

- "このメッセージを表示しない"オプションにより以前に無効化されたメッセージの通知を再有効化します。

9.2 Altova グローバルリソース

グローバルリソースはAltova 製品間において生産性を向上させるための機能で、現在以下のAltova 製品にて提供されています：:XMLSpy、MapForce、StyleVision、DatabaseSpy。Altova MissionKit を使用中のユーザーも、該当する製品でこの機能へアクセスすることができます。

一般的な利用方法：

- データ処理を行う様々なAltova アプリケーションを使用するワークフローの定義
- あるアプリケーションから別のアプリケーションを呼び出し、データ処理を行ったあと、元のアプリケーションへデータを戻すといった作業
- 入出力データ ファイル の場所をグローバルリソースとして定義
- ランタイムにおいてグローバルリソースを切り替えることで、開発リソースとデプロイリソースを切り替え
- 品質管理の試験における「もしも」シナリオ

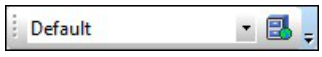
グローバルリソースの定義ファイルはGlobalResources.xml で、C:\Documents and Settings\UserName\My Documents\Altova\ 以下に収められています。これはグローバルリソースを使用する全てのAltova アプリケーションで共通したデフォルトの場所となります。そのため、グローバルリソースに対して行われた変更は、自動的に他のアプリケーションでも利用可能になります。ファイル名や位置を変更することもできます。詳細については グローバルリソース - プロパティ を参照ください。

一般的なメカニズム：

- グローバルリソースはアプリケーション内部にて **定義され**、自動的に保存されます。
- グローバルリソースはコンポーネントに対して **割り当てられ**、内部のデータを動的に扱うことができます。
- グローバルリソースはアプリケーション内部にて **呼び出され**、ランタイムにてリソースの切り替えを行うことができます。

このセクションでは…\MapForceExamples\Tutorial\ フォルダーに収められている既存のマッピングにてグローバルリソースを定義ならびに使用する方法を説明します。

グローバルリソースツールバーを利用可能にする：
メニューオプションの「ツール| カスタマイズ」を選択し、ツールバータブをクリックした後、グローバルリソースの管理のチェックボックスにチェックを入れます。この操作によりグローバルリソースツールバーが表示されます。



コンボボックスによりグローバルリソースの切り替えを行うことができます。"Default" エントリーは常に選択することができます。グローバルリソースアイコン をクリック (または「ツール| グローバルリソース」を選択) することで、グローバルリソースダイアログボックスが表示されます。

9.2.1 グローバルリソース - ファイル

MapForce のグローバルリソースでは：

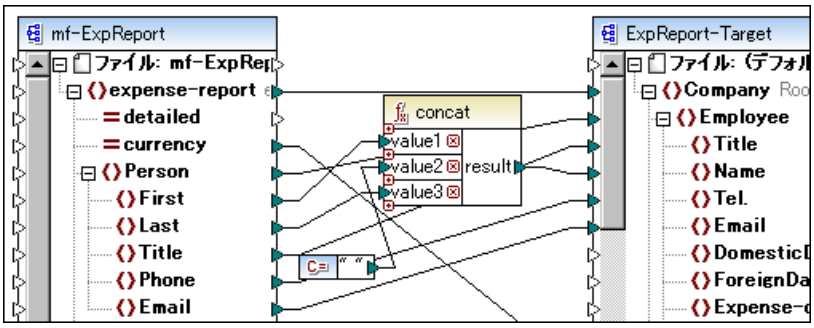
- XML、XML スキーマ ファイルといった、任意の入力/出力コンポーネントファイルをグローバルリソースとして定義することができます。

このセクションの目的：

- ソースコンポーネント入力ファイルmf-ExpReport をグローバルリソースとして定義する
- ランタイムにて **入力データ**となる 2 つのXML ファイルを **切り替え**、出力プレビュータブに表示されるXML 出力を確認する。

  このセクションでは…\MapForceExamples\Tutorial\ フォルダーに収められているTut-ExpReport.mfd ファ

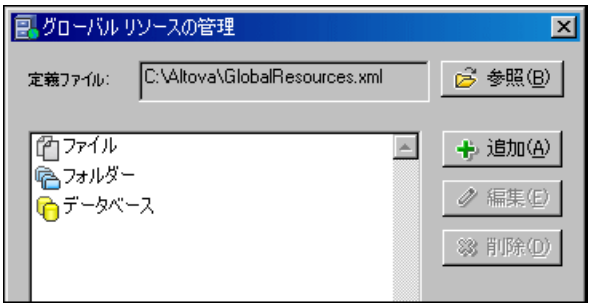
イルを使用します。



9.2.1.1 グローバルリソースの定義 /追加

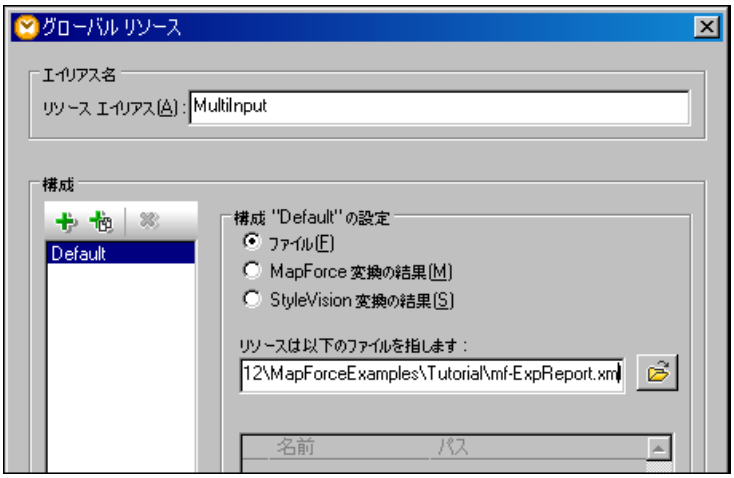
グローバルリソースファイルの定義 / 追加：

1. グローバルリソースアイコン をクリックして、ダイアログボックスを開きます。

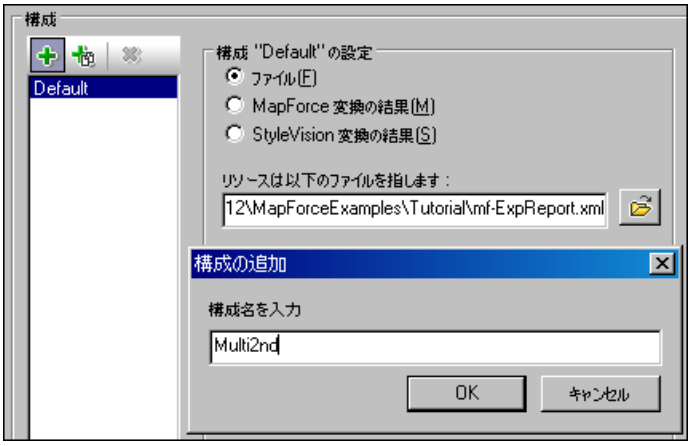


これがグローバルリソースが空の状態です。

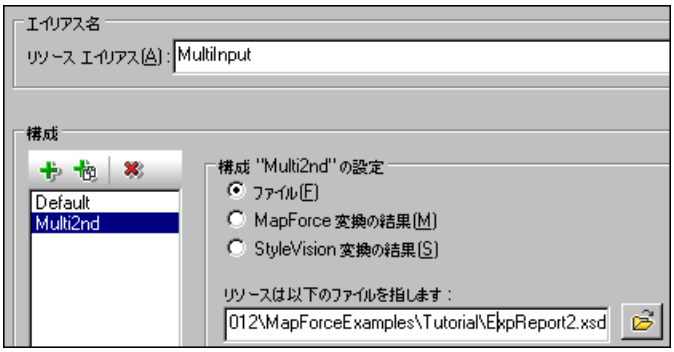
2. **「追加」**ボタンをクリックして、ポップアップから**ファイル**を選択します。
3. リソースエイリアスの名前（例：**MultiInput**）を入力します。
4. フォルダーを開くアイコンをクリックし、Default（デフォルト）の入力ファイルとなるXML ファイル（例：**mf-ExpReport.xml**）を選択します。



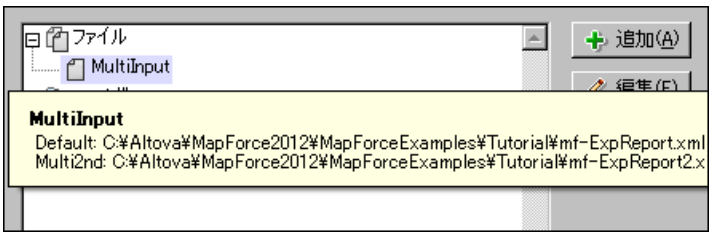
5. **構成**グループにある追加ボタン をクリックして、現在のエイリアスに対して新たな構成を追加します。構成のコピーアイコン を使用することで選択された構成をコピーして、新しい名前で保存することもできます。



6. 構成の名前（例えば **Multi2nd** ）を入力し、「OK」をクリックすることで確定します。
Multi2nd が構成リストに追加されます。
7. フォルダーを開くアイコンを再びクリックし、Multi2nd 構成にて入力ファイルとなるXML ファイル（例 **mf-ExpReport2.xml**）を選択します。



8. 「**OK**」をクリックしてリソースの定義を完了します。
MultiInput エイリアスがグローバルリソースのファイルセクションに追加されました。
マウスカーソルをエイリアスエントリーへマウスオーバーすることで、その定義がツールチップにて表示されます。



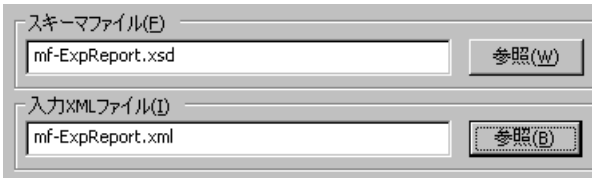
9. 「**OK**」をクリックして確定します。
この操作によりグローバルリソースの定義が完了します。次のステップでは、 グローバルリソースの割り当て について説明します。

9.2.1.2 グローバル リソースの割り当て

グローバルリソースをコンポーネントへ割り当てる
グローバルリソースが定義できたので、グローバルリソースをコンポーネントへ割り当てます。つまり、mf-ExpReport.xml ファイルがマッピングのソースファイルとして使用されます。

1. **mf-ExpReport** コンポーネントをダブルクリックして入力 XML ファイルフィールドの隣にある参照ボタンをクリックし

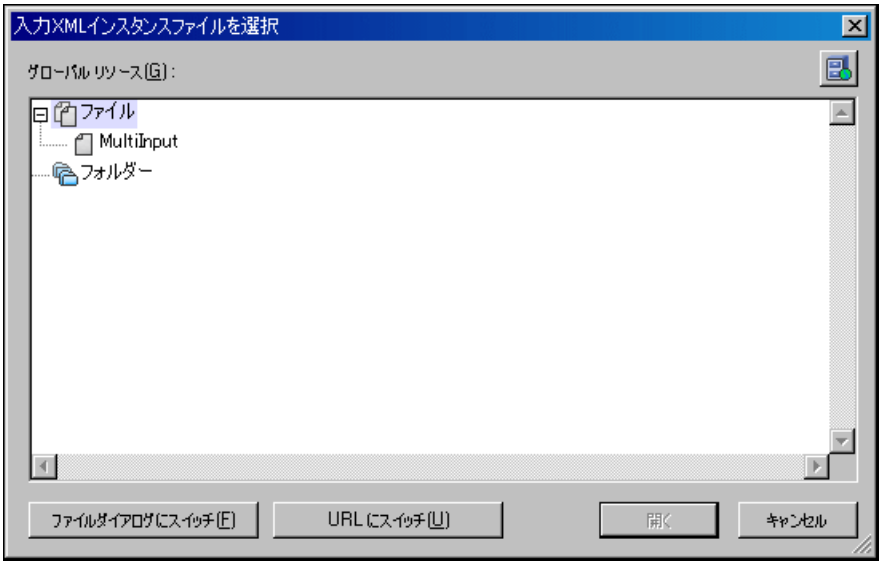
ます。



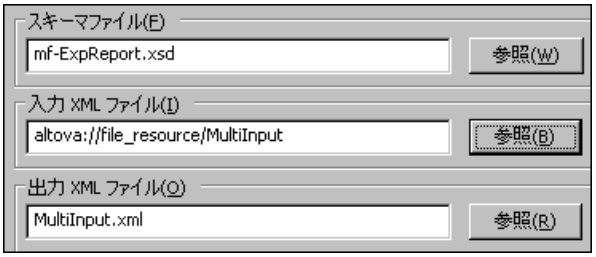
この操作により、「入力 XML インスタンスファイルを選択」ダイアログボックスが表示されます。

2. ダイアログボックスの下部にある**グローバルリソースにスイッチ**ボタンをクリックします。

3. 割り当てを行うリソース（この場合は **MultiInput** ）を選択して、開くをクリックします。



メモ：コンポーネントの入力 XML ファイルフィールドに、リソースに対する参照 **altova://file_resource/MultiInput** が表示されます

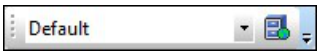


4. 「**OK**」をクリックして、コンポーネントに対するリソースの割り当てを完了します。

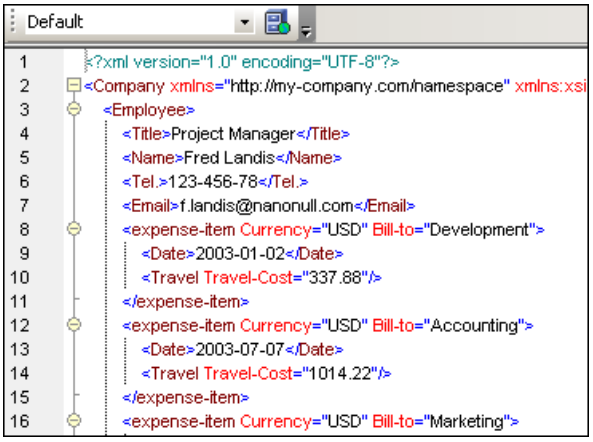
次のステップでは、[グローバルリソースの使用／アクティベート]を行います。

9.2.1.3 グローバルリソースの使用 / アクティベート

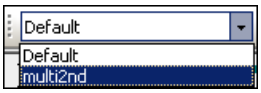
グローバルリソースの使用 / アクティベート
これまでの操作により **Default** 構成に対して定義された **MultiInput** エイリアスがアクティブになっています。これはグローバルリソースアイコンバーの表示が **"Default"** となっていることで確認することができます。



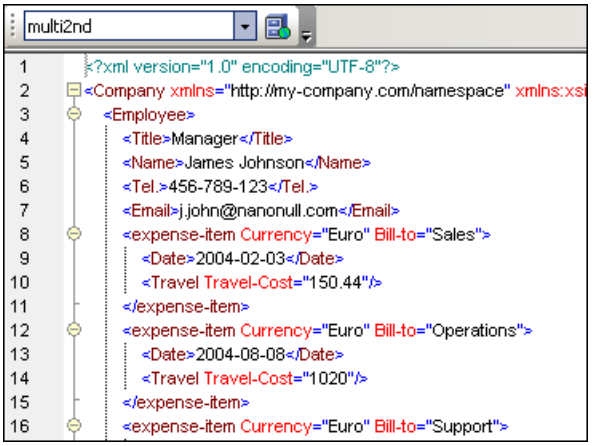
1. 出力タブをクリックして、マッピングの結果を確認します。



2. **マッピング**タブをクリックして、マッピングビューへ戻ります。
3. グローバルリソースコンボボックスをクリックして、コンボボックスから **multi2nd** を選択します。



4. 出力タブをクリックして、新たな結果を確認します。
グローバルリソースコンボボックスをクリックして、コンボボックスから multi2nd を選択します。



メモ:
現在アクティブになっているグローバルリソース（グローバルリソースツールバーにある **Multi2nd**）により、マッピング結果が決定されます。コードの生成を行った場合でも同様のことが起こります。

9.2.2 グローバルリソース - フォルダー

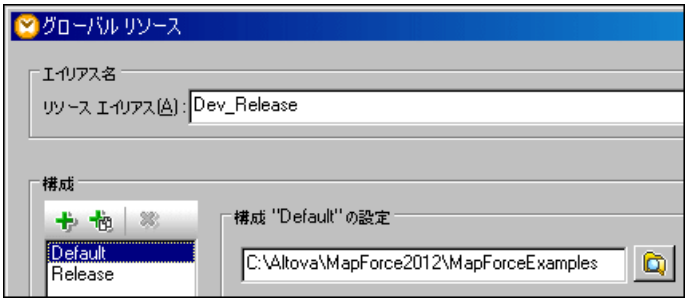
フォルダーをグローバルリソースとして定義することも可能で、例えば開発用とリリース用といった異なるフォルダーに含まれているファイルを参照するよう入力コンポーネントを定義することができます。

MapForce ではXSLT やその他のプログラムコードを生成する場合、ターゲットとなるフォルダーを指定するよう毎回促されるため、出力コンポーネントに対するフォルダーの定義はそこまで有効ではありません。

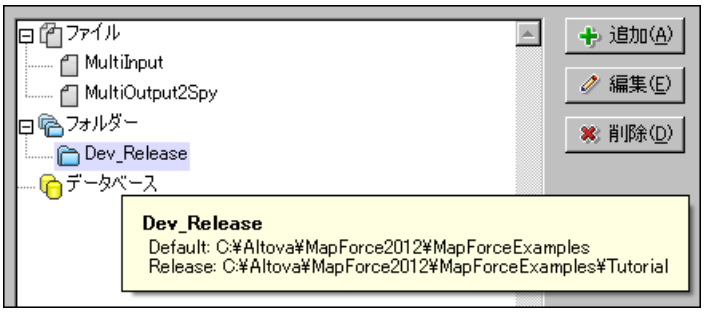
このセクションでは…\MapForceExamples\Tutorial\ フォルダーに収められているglobal-**フォルダー。mfd** ファイルを使用します

グローバルリソースフォルダーの定義 /追加

1. グローバルリソースアイコン をクリックして、ダイアログボックスを開きます。
2. **「追加」** ボタンをクリックして、ポップアップから**フォルダー**を選択します。
3. リソースエイリアスの名前 例：**Dev_Release**）を入力します。
4. フォルダーを開くアイコンをクリックし、"Default" の入力フォルダーとなる .\MapForceExamples を選択します。



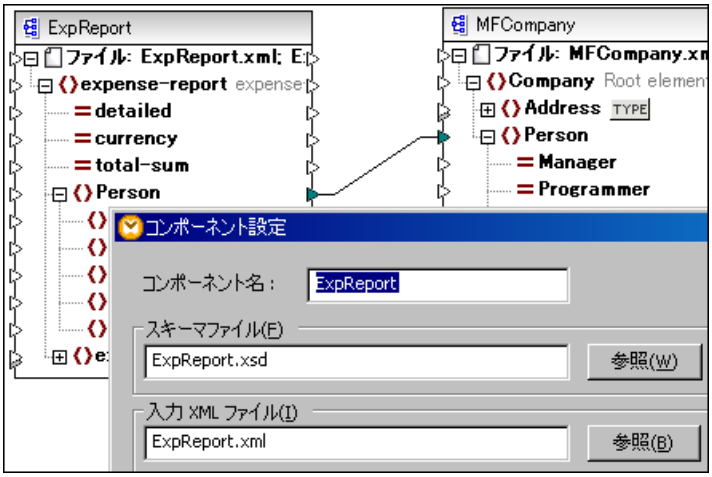
5. **構成**グループの追加ボタン をクリックして、現在のエイリアスに対して新たな構成の名前を入力 例：Release）して追加します。構成のコピーアイコンglob-copyic を使用することで選択された構成をコピーして、新しい名前にて保存することができます。
6. フォルダーを開くアイコンをクリックし、"Release" の入力フォルダーとなる .\MapForceExamples\Tutorial を選択します。



7. **「OK」** をクリックして、グローバルフォルダーの定義を完了します。

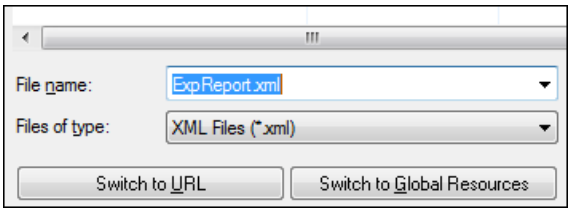
グローバルリソースフォルダーの割り当て

1. **ExpReport** コンポーネントをダブルクリックして、**入力 XML ファイルフィールド**の隣にある参照ボタンをクリックします

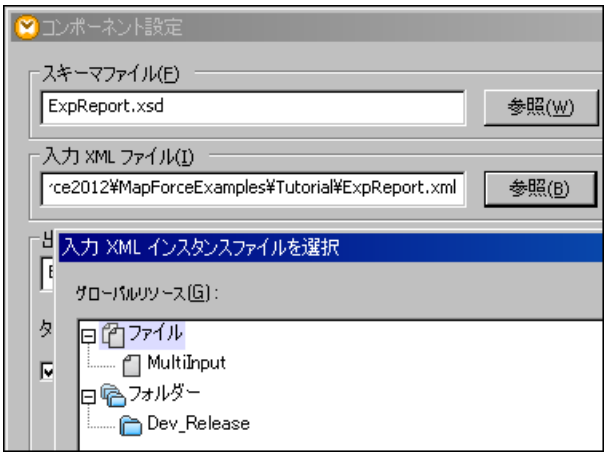


この操作により、XML インスタンスファイルを選択ダイアログボックスが表示されます。

2. ダイアログボックス下部にある**グローバルリソースにスイッチ**ボタンをクリックします。

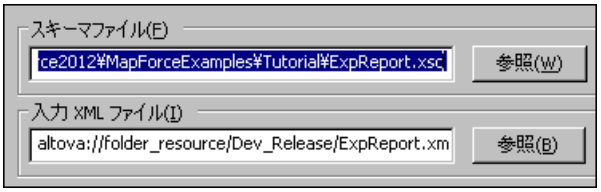


3. 割り当てを行うリソース（この場合では**Dev_Release**）をクリックして、**「OK」**をクリックします。



「ファイルを開く」ダイアログボックスが表示されます。

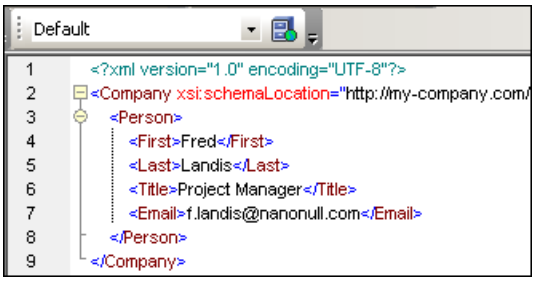
4. 両方のフォルダーにある、開発および本番リソース用の**リソース**ファイルとして使用することができるファイル（例えば**ExpReport.xml**）を選択して、**「OK」**をクリックすることでリソースフォルダーの割り当てを完了します。



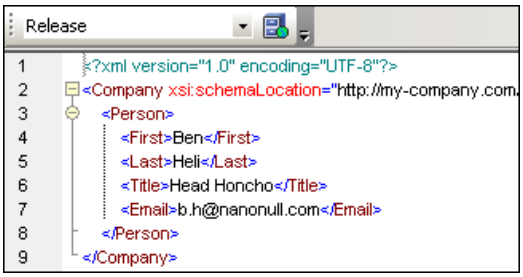
内容は違うものの、名前は同じファイルが両方のフォルダーに収められていることを確認してください。

ランタイムにてリソースフォルダーを変更する：

1. 出力タブをクリックして、変換の結果を確認します。
   Default 構成のフォルダー ...\MapforceExamples によりこの出力が作成されていることに注目してください。



2. マッピングタブをクリックしてマッピングウィンドウへ戻ります。
3. グローバルリソースコンボボックスをクリックして、"Relesae" エントリーを選択します。
4. 出力ボタンをクリックして、リリースグローバルリソースを使用し結果を確認します。

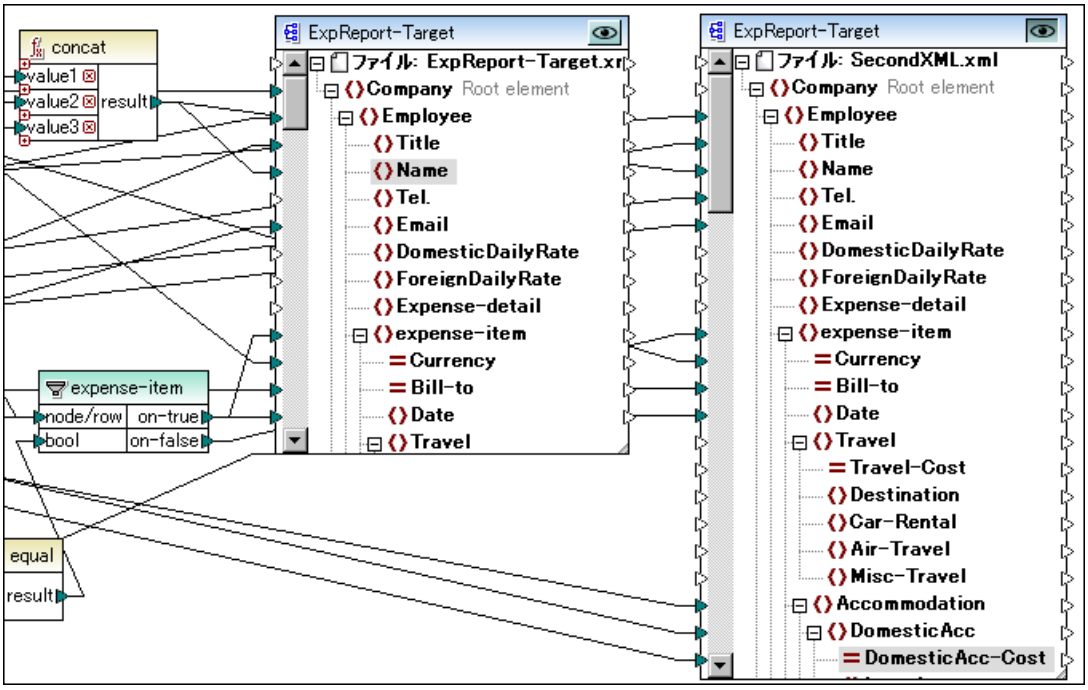


Release 構成のフォルダー ...\MapforceExamples/Tutorial のデータによる出力が表示されます。

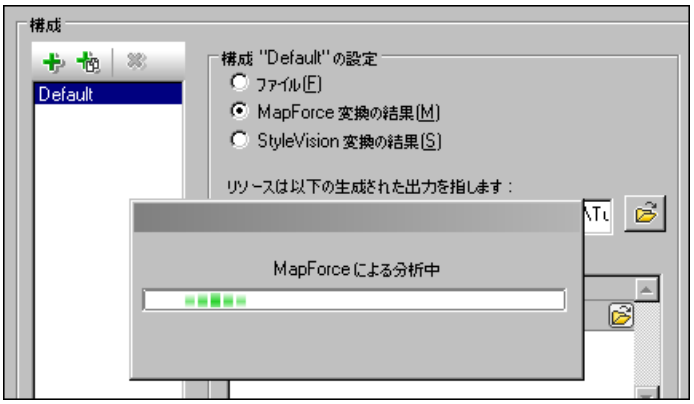
9.2.3 グローバルリソース - アプリケーションワークフロー

このセクションでは、 2 つの Altova アプリケーション間におけるワークフローを作成します。ワークフローでは XMLSpy から MapForce が起動され、生成された XML ファイル出力を XMLSpy へ戻すことで、更なる処理を行います。

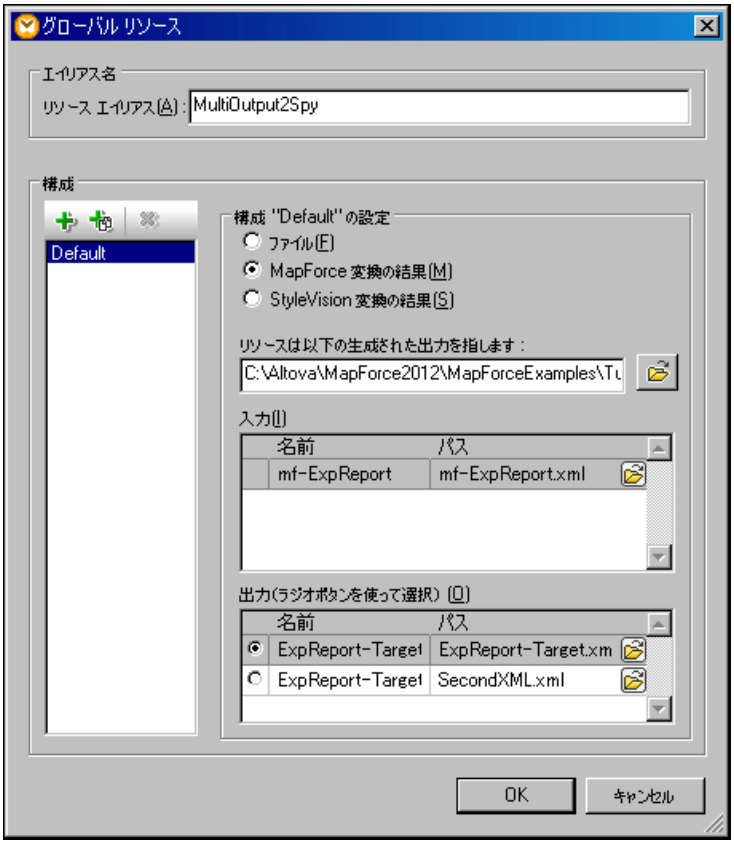
このマッピングでは 2 つの出力コンポーネントが使用され、経費報告入力ファイルの旅費に関するファイルと、それ以外の費用に関するファイルと言うかたちでフィルタリングされた、 2 つの出力が作成されます。このセクションでは ...\MapForceExamples\Tutorial\ フォルダーに収められている Tut-**ExpReport-multi.mfd** マッピングファイルが使用されます。



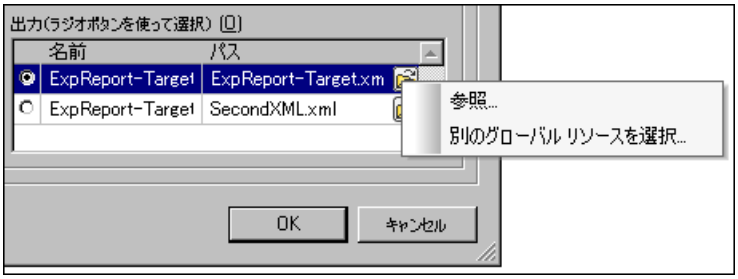
1. *グローバルリソースアイコン をクリックして、ダイアログボックスを表示します。*
2. ***追加**ボタンをクリックして、ポップアップから**ファイル**を選択します。*
3. *リソースエイリアスの名前 例えば**MultiOutput2Spy**）**を入力します。***
4. ***「MapForce　変換の結果」**ラジオボタンをクリックした後、ファイルを開くアイコンをクリックします。*
5. **Tut-ExpReport-multi.mfd** *マッピングを選択します。*



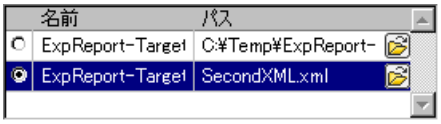
MapForce によりマッピングの解析が行われ、入力ならびに出力ファイルが別々のリストボックスにて表示されます。



6. **出力**セクションにある一番上のラジオボタンを選択します（選択されていない場合）。
出力ファイル名は**ExpReport-Target.xml** で、現在 **Default** の構成を定義しているということに注意してください。
7. アイコンをクリックして、出力ファイルの新しい場所を定義します。ポップアップメニューから参照 ...を選択します（例：C:\Temp）。

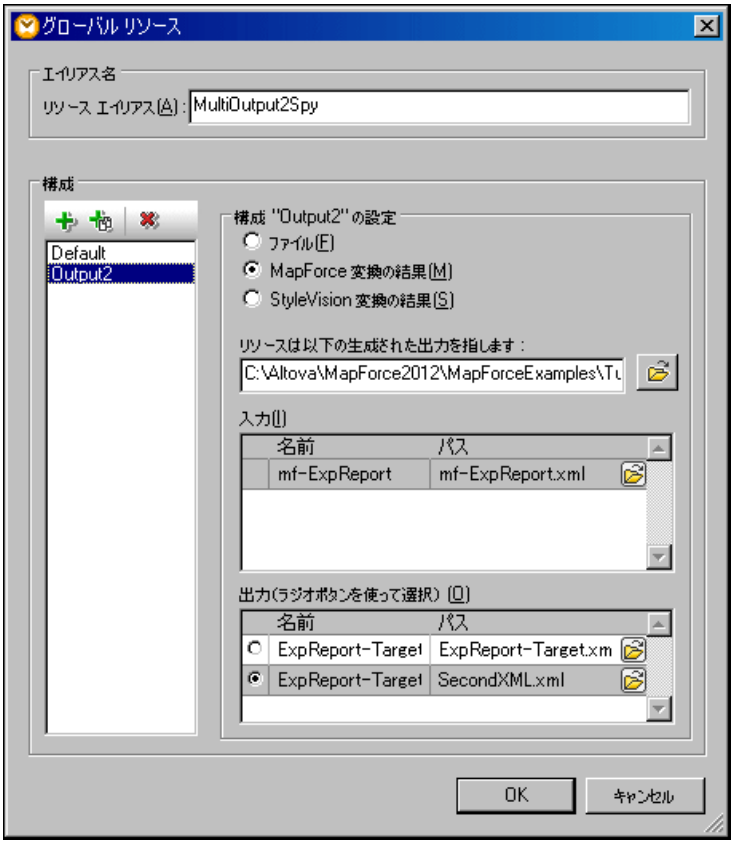


8. 新たな出力場所（例えばC:\Temp）を入力して、保存ボタンをクリックします。コンポーネント設定とは異なる場所も指定することができます。
9. ダイアログボックスの構成グループにある追加ボタン をクリックして、リソースエイリアスに新たな構成を追加します。
10. 構成の名前（例えば**Output2**）を入力し、ファイルを開くアイコンをクリックした後、**Tut-ExpReport-multi.mfd** を選択します。
11. 出力リストボックスにある2番目のラジオボタンを選択します。出力ファイル名が**SecondXML.xml** となっていることに注目してください。



12. 名前を付けて保存アイコン をクリックして、出力ファイルの新たな場所（例えば C:\Temp）をポップアップメニューから定義します。

メモ：ポップアップにて **「別のグローバルリソースを参照」** を選択すると、MapForce 出力をグローバルリソースとして保存することができます。つまり、マッピングの出力として作成されるファイルに対して、グローバルリソースの参照が行われます。



13. **「OK」** をクリックしてグローバルリソースを保存します。

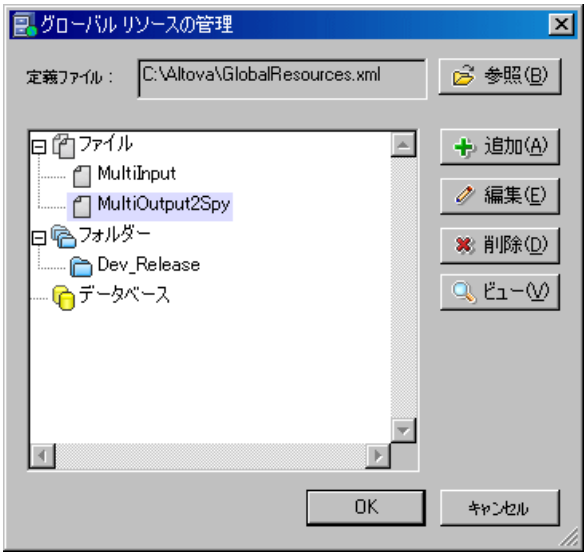
新しいリソースエイリアスの **MultiOutput2Spy** がグローバルリソースの定義ファイルへ追加されます。

14. **「OK」** をクリックして定義を完了します。

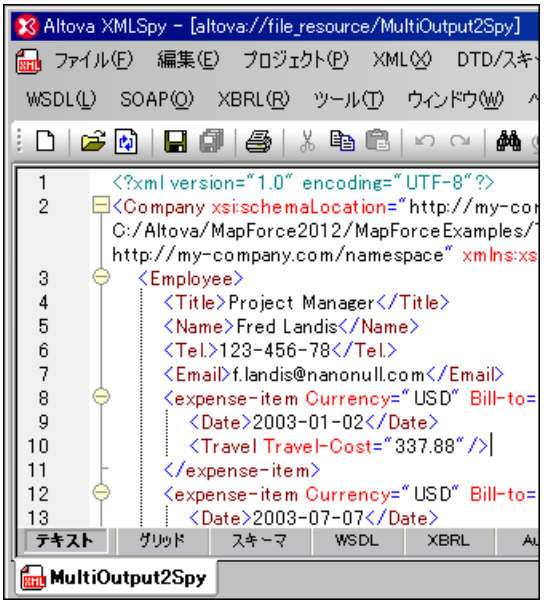
9.2.3.1 アプリケーションワークフローの起動

このセクションではXMLSpy におけるグローバルリソースの使用方法と MapForce の変換結果が返ってくるまでを説明します。

1. Start XMLSpy を起動して、MapForce が動作している場合は、(これらのアプリケーションがどのように動作するのか確認するため) MapForce を終了します。
2. XMLSpy にてメニューオプションの **「ツール | グローバルリソース」** を選択します。
3. **MultiOutput2Spy** エントリーを選択してビューボタンをクリックします



MapForce が変換を行なっているというメッセージボックスが表示され、変換結果がテキストビューウィンドウにて表示されます。



メモ:

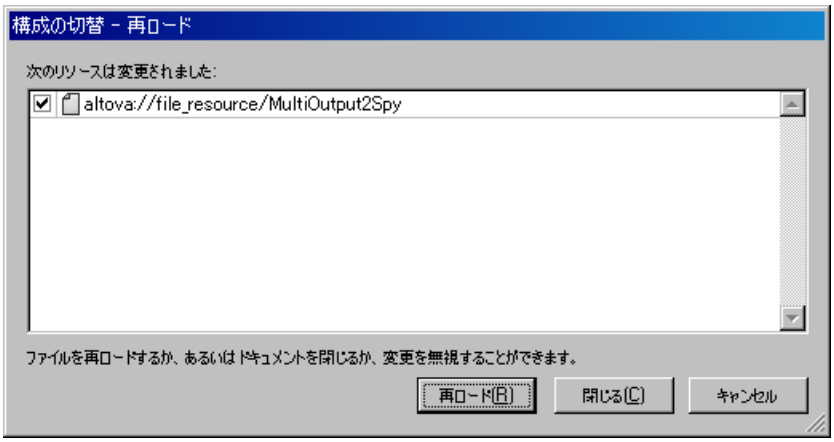
- 現在選択されている構成は **"Default"** です。
- リソースエイリアスの名前が、アプリケーションタイトルバーに表示されます：**altova://file_resource/**

MultiOutput2Spy

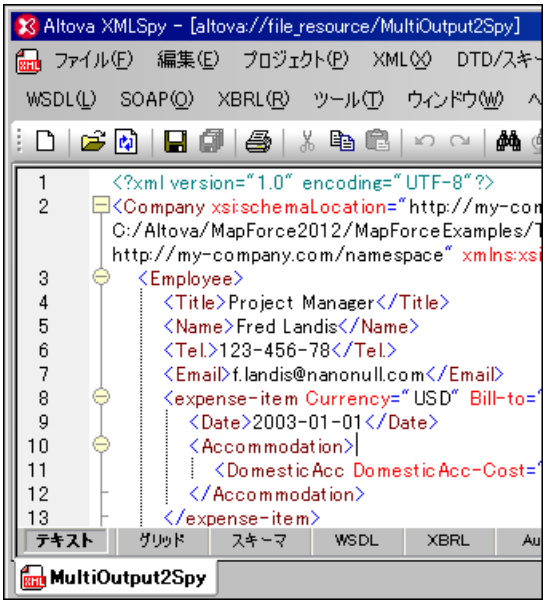
- 出力ファイルが MultiOutput2Spy という名前で開かれ、更に処理や編集などを行うことができます。
- **ExpReport-Target.xml** ファイルが C:\Temp フォルダーへコピーされます。

旅費以外の経費出力を取得する:

1. メニューオプションの「ツール| アクティブな構成」から "**Output2**" を選択します。
   以下のようなメッセージボックスが表示されます。



2. **再ロード**ボタンをクリックして、リソースにより定義されている 2 番目の出力ファイルを取得します。



変換結果がテキストビューウィンドウに表示され、それまで表示されていた "MultiOutput2Spy.xml" ファイルが上書きされます。

メモ:

- 現在選択されている構成は "**Output2**" です。
- 出力ファイルが MultiOutput2Spy という名前で開かれ、更に処理や編集などを行うことができます。
- **SecondXML.xml** ファイルが C:\Temp フォルダーへコピーされます。

9.2.4 グローバルリソース - プロパティ

グローバルリソース XML ファイル

グローバルリソースの定義は XML ファイルに収められています。デフォルトでは、この XML ファイルは GlobalResources.xml という名前で、C:\Documents and Settings\<username>\My Documents\Altova\ フォルダー以下に収められています。このファイルは全ての Altova アプリケーションにおいて、デフォルトのグローバルリソース XML ファイルとしてセットされます。全てのアプリケーションがこのファイルを使用することにより、あるアプリケーションで定義されたグローバルリソースを、他の Altova アプリケーションでも使用することができるようになります。

ファイルの名前や場所の変更はいつでも行うことができ、複数のグローバルリソース XML ファイルを持つこともできます。各アプリケーションが同時に持つことができるアクティブな XML ファイルは 1 つだけで、そのアクティブなファイルに記述されている定義だけをアプリケーションで利用することができます。

グローバルリソース XML ファイルをアクティブにするには、定義ファイルフィールドにある参照ボタンをクリックして、使用するファイルをダイアログボックスにて選択します。

グローバルリソースの管理：追加、編集、削除

グローバルリソースダイアログでは、選択されたグローバルリソース XML ファイルに対してグローバルリソースの追加、編集、削除を行うことができます。グローバルリソース XML ファイルでは、ファイルとフォルダーという分類とともにエイリアスが管理されます。

グローバルリソースを追加する：

追加ボタンをクリックして、表示されるグローバルリソースダイアログボックスにてグローバルリソースの定義を行います。グローバルリソースの定義と保存を行うと、グローバルリソースが選択されたグローバルリソース XML ファイルのグローバル定義リストに追加されます。

グローバルリソースを追加する：

グローバルリソースを選択して、編集ボタンをクリックします。グローバルリソースダイアログが表示され、必要な変更を加えることができます。

グローバルリソースを削除する：

グローバルリソースを選択して、削除ボタンをクリックします。

アプリケーションワークフローにて結果を確認する：

呼び出し元のアプリケーション（例えば XMLSpy）により、他のアプリケーション（例えば MapForce）が呼び出される場合、グローバルリソースの管理ダイアログボックスにてビューボタンが利用可能になります。

ビューボタンをクリックすることで、現在選択されているグローバルリソースが呼び出し先のアプリケーションで得られる効果を確

認することができます。サンプルについては グローバルリソース - アプリケーションワークフロー を参照ください。

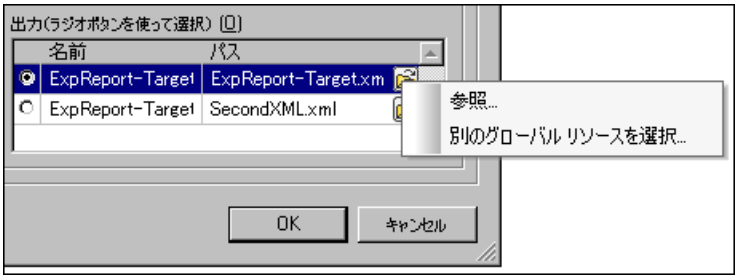
グローバルリソースの管理ダイアログボックスにて行われた変更を保存する：

グローバルリソースの追加、編集、または削除が完了した後には、グローバルリソースダイアログにある **「OK」** ボタンをクリックして、変更をグローバルリソースXML ファイルへ保存するようにしてください。

メモ：エイリアス名は、ファイルとフォルダーのセクション毎に一意な名前でなければなりません。しかし、セクション毎に同じ名前を使用することは可能で、例えば同じ名前のエイリアスをファイルとフォルダーセクションで作成することは可能です。

MapForce の変換結果をグローバルリソースとして選択

複数の出力が存在するMapForce の変換では、ラジオボタンをクリックすることで、グローバルリソースに対してどの出力ファイルが使用されるかを選択することができます



マッピングにより生成される**出力**ファイルは以下の方法により保存することができます：

- ポップアップにて別のグローバルリソースを選択エントリーを選択し**altova://file_resource/MF_output** という形式でパスが記述されます。出力は、グローバルリソースにより参照されているファイルへ記述されます。

- アイコンにより指定された（上のスクリーンショットではSecondXML.xml という名前の）ファイル。

  上にある方法のどちらも選択されていない場合、グローバルリソースが使用された時に、**一時的な** XML ファイルが作成されます。

ランタイムにてどのリソースを使用するか決定する

どのグローバルリソースを使用することができるか、またどのグローバルリソースが実際に使用されるのかを決定するには 2 つの方法があります：

- グローバルリソースダイアログにてアクティブなグローバルリソースXML ファイルを選択します。アクティブなグローバルリソースXML ファイルはいつでも変更することができ、アクティブになったファイルのグローバルリソース定義により、それまでの定義が更新されます。

  アクティブなグローバルリソースXML ファイルにより、(i) どのグローバルリソースを割り当てることができるのか、(ii) どのグローバルリソースを表示することができるのかが決定されます。 例えば、あるグローバルリソースXML ファイル内のグローバルリソースが割り当てられていたが、現在アクティブになっているグローバルリソースXML ファイルにて、同じ名前のグローバルリソースが存在しない場合、割り当てられたグローバルリソース（エイリアス）を表示することはできません）。

- メニューアイテム「ツール| アクティブな構成」またはグローバルリソースリソースツールバーにてアクティブな構成 を選択します。このコマンドを選択すると、全てのエイリアスにある構成が表示されます。

  構成を選択することで、その構成がアプリケーション全体でアクティブになります。この操作により、どこでグローバルリソース（またはエイリアス）が使用されようとも、アクティブな構成に割り当てられているリソースがロードされます。

アクティブな構成は使用されている全てのエイリアスに対して割り当てられます。アクティブな構成に対して、対応するエイリアスが存在しない場合、そのエイリアスの**Default 構成**が使用されます。アクティブな構成はリソースの割り当てを行うにあたり重要ではなく、リソースが実際に使用されるかどうかが重要になります。

リソース / 構成の変更

リソースは、異なる構成名を選択することで切り替えることができます。以下にある 2 つの方法で実現することができます：

- メニューコマンド「ツール| アクティブな構成」へマウスカーソルを移動すると、グローバルリソースXML ファイルにて定義されている全ての構成が表示されます。目的の構成をサブメニューから選択します。

- グローバルリソースツールバーのコンボボックスにて、目的の構成を選択します。メニューコマンドから「ツール| カスタマイズ」を選択し、ツールバータブをクリックした後にグローバルリソースのチェックボックスを切り替えることで、グローバルリソースツールバーの表示を切り替えることができます。.

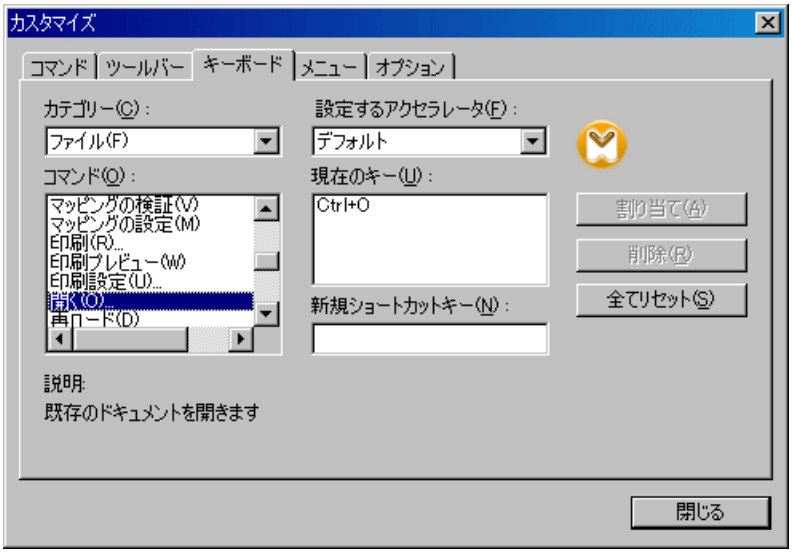
9.3 キーボードのショットカットのカスタム化

MapForce 内のキーボードのショットカットを以下のように定義または変更することができます :

1. **「ツール」** メニューから **「カスタマイズ」** をクリックします。
2. **「キーボード」** タブをクリックします。

コマンドに新規ショートカットを割り当てる :

1. **「ツール」 「カスタマイ」** コマンドを選択して、キーボードタブをクリックします。
2. **カテゴリー** コンボボックスをクリックして、メニュー名を選択します。
3. コマンドリストボックス内で新しいショートカットを割り当てる**コマンド**を選択します。
4. **新規ショートカットキー**をクリックします : テキストボックスから、コマンドを有効化するショートカットキーを押します。



ショートカットはテキストボックス内にすぐに表示されます。ショートカットが以前に割り当てられている場合、その関数がテキストボックスの下に表示されます。

5. **割り当て** ボタンをクリックして、ショートカットを割り当てます。
現在のキーリストボックスにショートカットが表示されます。
(新規ショートカットキーテキストボックス内のエントリを**クリア**するには、コントロールキーを押します **CTRL**、**ALT** または **SHIFT**)。

ショートカットを元に戻す、または削除する :

1. 現在のキーリストボックス内から削除するショートカットをクリックします。
2. **削除** ボタンをクリックします。
3. **閉じる** ボタンクリックして、確認します。

メモ : 設定するアクセラレーターには現在関数が存在しません

現在キーボードに割り当てられているショートカットは、以下のとおりです :

F1	ヘルプメニュー
F2	次のブックマークへ (出力ウィンドウにて)
F3	次を検索
F10	メニューバーをアクティベート
Num +	現在のアイテムノードを展開

Num -	アイテムノードを縮退
Num *	現在のアイテムノード以下を全て展開
CTRL + TAB	開かれているマッピング間の切り替え
CTRL + F6	開かれているウィンドウをサイクル
CTRL + F4	アクティブなマッピングドキュメントを閉じる
Alt + F4	MapForce を閉じる
Alt + F, F, 1	最後に閉じられたファイルを開く
Alt + F, T, 1	最後に閉じられたプロジェクトを開く
CTRL + N	新規ファイル
CTRL + O	ファイルを開く
CTRL + S	ファイルの保存
CTRL + P	ファイルの印刷
CTRL + A	すべて選択
CTRL + X	切り取り
CTRL + C	コピー
CTRL + V	貼り付け
CTRL + Z	元に戻す
CTRL + Y	やり直し
Del	コンポーネントを削除（プロンプト付き）
Shift + Del	コンポーネントを削除（プロンプトなし）
CTRL + F	検索
F3	次を検索
Shift + F3	前を検索

矢印キー

（上 / 下）	コンポーネント内にあるアイテムを選択
Esc	編集の破棄 / ダイアログボックスを閉じる
Return	選択の確定

出力ウィンドウホットキー

CTRL + F2	ブックマークの挿入 削除
F2	次のブックマークへ
SHIFT + F2	前のブックマークへ
CTRL + SHIFT + F2	全てのブックマークを削除

ズーム（倍率）ホットキー

CTRL + 上向きマウスホイール	ズームイン
CTRL + 下向きマウスホイール	ズームアウト
CTRL + 0 (ゼロ)	倍率のリセット

9.4 カタログファイル

MapForce は OASIS XML カタログ機能のサブセットをサポートしています。MapForce のカタログ機能により、広く使われているスキーマ (またはスタイルシートなどその他のファイル) をローカルユーザーフォルダーから取得することができます。これにより全体的な処理スピードが向上し、(ネットワークに接続していない状態でも) ユーザーはオフラインで作業することができるようになり、(URI の変更はカタログファイル内部に限定されるので) ドキュメントのポータビリティーが向上します。

MapForce のカタログ機能は以下のように動作します。

RootCatalog.xml

MapForce が開始されると、(以下に示されるような構造を持った) RootCatalog.xml というファイルがロードされ、ファイル内部に記述されているカタログファイルが検索されます。このファイルを修正して、検索するカタログファイルを増やすこともできます (各カタログファイルは nextCatalog 要素にて記述されます)。各カタログファイル内に記述されている URI が、ファイル内で指定されたマッピングに従って解決されます。

```
<?xml version="1.0" encoding="UTF-8"?>
<catalog xmlns="urn:oasis:names:tc:entity:xmlns:xml:catalog"
     xmlns:spy="http://www.altova.com/catalog_ext"
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     xsi:schemaLocation="urn:oasis:names:tc:entity:xmlns:xml:catalog
Catalog.xsd">
  <nextCatalog catalog="%PersonalFolder%/Altova/%AppAndVersionName%/
CustomCatalog.xml"/>
  <nextCatalog catalog="CoreCatalog.xml"/>
  <!-- Include all catalogs under common schemas folder on the first
directory level -->
  <nextCatalog spy:recurseFrom="%AltovaCommonFolder%/Schemas"
catalog="catalog.xml" spy:depth="1"/>
  <!-- Include all catalogs under common XBRL folder on the first directory
level -->
  <nextCatalog spy:recurseFrom="%AltovaCommonFolder%/XBRL"
catalog="catalog.xml" spy:depth="1"/>
</catalog>
```

上で示されるファイルの内容にて、%AltovaCommonFolder% という変数で指定されている スキーマならびに XBRL フォルダー内部に catalog.xml という名前のカタログファイルがあることに注目してください (%AltovaCommonFolder% 変数の値については以下を参照ください)。

Altova 共通フォルダー以下にあるカタログファイルにより、SVG や WSDL といった よく使われる定義済みのスキーマや XBRL タクソノミのパブリックならびにシステム識別子が、ローカルにコピーされた対応するスキーマを指し示す URI にマッピングされます。これらのスキーマは MapForce がインストールされる際に Altova 共通フォルダーにインストールされます。エラーが発生する可能性があるので、これらファイル内のマッピングを複製しないようご注意してください。

CoreCatalog.xml、CustomCatalog.xml、そして Catalog.xml

上で記されている RootCatalog.xml にて、CoreCatalog.xml と CustomCatalog.xml が検索のために使用されていることに注目してください:

- `CoreCatalog.xml` には `Altova` 共通フォルダーを指し示すために使われる `Altova` 独自のマッピングが含まれています。
- `CustomCatalog.xml` はマッピングを作成するためのスケルトンファイルです。`Altova` 共通フォルダー以下のカタログファイルで指定されていないスキーマファイルへのマッピングを `CustomCatalog.xml` に追加することができます。`OASIS` カタログ機能によりサポートされている要素を使用してください (以下を参照)。
- Altova 共通フォルダーには数多くの Catalog.xml ファイルが格納されています。各ファイルは Altova 共通フォルダー以下のスキーマフォルダーまたは XBRL タクソノミフォルダーに格納され、ローカルにコピーされたパブリックならびにシステム識別子のスキーマを指し示す URI のマッピングを行ないます。

カタログファイルとスキーマの場所

RootCatalog.xml ならびに CoreCatalog.xml は MapForce アプリケーションフォルダー以下にインストールされます。CustomCatalog.xml は MyDocuments\Altova\MapForce フォルダー以下に納されています。catalog.xml ファイルは各スキーマフォルダー以下にあり、これらスキーマフォルダーは %AltovaCommonFolder%\Schema ならびに %AltovaCommonFolder%\XBRL 以下に収められています。

シェル環境変数とAltova 変数

nextCatalog 要素ではシェル環境変数が使用され、システムに関係する場所へのパスが参照されます（上の RootCatalog.xml の内容を参照）。以下のシェル環境変数がサポートされます：

%AltovaCommonFolder%	C:\Program Files\Altova\CommonMapForce
%DesktopFolder%	現在のユーザーのデスクトップフォルダーへのフルパス
%ProgrammenuFolder%	現在のユーザーのプログラムメニューフォルダーへのフルパス
%StartMenuFolder%	現在のユーザーのスタートメニューフォルダーへのフルパス
%StartUpFolder%	現在のユーザーのスタートアップフォルダーへのフルパス
%templateFolder%	現在のユーザーのテンプレートフォルダーへのフルパス
%AdminToolsFolder%	現在のユーザーの管理ツールが収められているシステムディレクトリへのへのフルパス
%AppDataFolder%	現在のユーザーのアプリケーションデータフォルダーへのフルパス
%CommonAppDataFolder%	全てのユーザーのアプリケーションデータが収められているファイルディレクトリへのフルパス
%FavoritesFolder%	現在のユーザーのお気に入りフォルダーへのフルパス
%PersonalFolder%	現在のユーザーの個人用フォルダーへのフルパス
%SendToFolder%	現在のユーザーの SentTo フォルダーへのフルパス
%FontsFolder%	システムフォントフォルダーへのフルパス
%ProgramFilesFolder%	現在のユーザーの Program Files フォルダーへのフルパス
%CommonFilesFolder%	現在のユーザーの Common Files フォルダーへのフルパス
%WindowsFolder%	現在のユーザーの Windows フォルダーへのフルパス
%SystemFolder%	現在のユーザーのシステムフォルダーへのフルパス
%CommonAppDataFolder%	全てのユーザーのアプリケーションが収められているファイルディレクトリへのフルパス

%LocalAppDataFolder%	ローカルアプリケーションのデータリポジトリとして使用されるファイルシステムディレクトリへのフルパス
%MyPicturesFolder%	マイピクチャフォルダーへのフルパス

カタログの動作

DTD が呼び出された際のリダイレクトにカタログは使用されます。カタログファイル内に記述されているマッピング情報を使用することで、パブリックまたはシステム識別子をローカルの URI にマッピングします。例えば XML ファイル内で宣言されている DOCTYPE 宣言が読まれたとき、カタログファイルマッピングにより PUBLIC または SYSTEM 識別子がローカルにあるリソースを参照します。

よく使われるスキーマや PUBLIC 識別子は通常定義が定まっており、ローカルにコピーされている URI をカタログファイルが指し示すだけで事足りてしまいます。XML ドキュメントが解析されると内部にある PUBLIC 識別子が読み込まれます。カタログファイル内に識別子が見つかった場合、カタログファイル内の対応する URL が検索され、その場所からスキーマが読み込まれます。例えば、以下の SVG ファイルが MapForce で開かれた場合：

```
<?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>
```

このドキュメントが読み込まれ、PUBLIC 識別子のカタログが検索されます。例えば、カタログファイルに以下のような記述があるとしましょう：

```
<catalog>
  ...
    <public publicId="-//W3C//DTD SVG 1.1//EN" uri="schemas/svg/svg11.dtd"/>
  ...
</catalog>
```

これで PUBLIC 識別子に対するマッチが見つかりました。SVG DTD は schemas/svg/svg11.dtd にリダイレクトされ（パスはカタログファイルからの相対パスになります）、このローカルファイルが DTD として使用されます。カタログ内に対応するマッピングが存在しない場合、XML ドキュメントに記述されている URL （上の例の場合だと http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd）が使用されます。

MapForce によりサポートされるカタログサブセット

CustomeCatalog.xml （または MapForce により読み込まれる他のカタログファイル）内でエントリを作成する際、以下に示される OASIS カタログ仕様の要素だけを使うようにしてください。以下に各要素と属性の説明を示します。更に詳しい情報については、XML カタログ仕様（英文）を参照ください。各要素は xml:base 属性を持ち、要素のベース URI を指定できることに注意してください。

- `<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"/>`

始どのスタイルシートの様に、パブリックな識別子が存在しない場合、system 要素を使うことでシステム識別子をURL に直接マッピングすることができます。また、uri 要素によりURI を別のURI にマッピングすることもできます。rewriteURI と rewriteSystem 要素によりURI またはシステム識別子の最初の文字列を書き換えることができます。これによりファイルパスの最初の部分を置き換えて、別のディレクトリにターゲットを変更することができます。これら要素に関する詳しい情報については XML カタログ仕様 (英文) を参照ください。

スキーマに従ったファイル拡張子とインテリジェントな編集

指定したスキーマのルールに準拠するようMapForce のインテリジェントな編集機能が適用された場合、カタログファイルを使うことで、ドキュメントに特定の拡張子を指定することができます。例えば独自のファイル拡張子 .myhtml を作成して、HTML DTD に従ったファイルの検証を行ないたい場合、以下の要素をCustomCatalog.xml 内にある <catalog> 要素の子要素として追加することで、この拡張子を持つファイルのインテリジェント編集を行うことができます。

```
<spy:fileExtHelper ext="myhtml" uri="schemas/xhtml/xhtml1-transitional.dtd"/>
```

これによりMapForce 内で .myhtml ファイルを編集する際のインテリジェント編集 (自動補完、入力ヘルパーなど) が XHTML 1.0 Transition DTD に従った形で使えるようになります。似たようなエントリを含んでいる %AltovaCommonFolder%\Schemas\xhtml フォルダー以下の catalog.xml ファイルも参照ください。

XML スキーマとカタログ

XML スキーマ情報はMapForce に内蔵されており、XML スキーマドキュメントの妥当性は内部にある情報をベースにチェックされます。そのため、XML スキーマではいかなる参照も発生しません。

%AltovaCommonFolder%\Schemas\schema フォルダー以下にある catalog.xml ファイルには、古い XML スキーマ仕様の DTD に対する参照が記述されています。このファイルから参照されるスキーマファイルを使って XML スキーマドキュメントの検証を行うべきではありません。これらのファイルは、以前の勧告に従ったドキュメントを作成する際に、MapForce の入力ヘルパーにて該当する編集情報を表示させるためだけに使用されます。

更に詳しい情報

カタログに関する更に詳しい情報は XML カタログ仕様 (英文) を参照ください。

Chapter 10

メニューレファレンス

10 メニューレファレンス

以下のセクションでは、MapForce に用意されているメニューやメニューオプションのリストと、それらの簡単な説明を記します。

10.1 ファイル

新規作成
新規マッピングドキュメントを開きます。

開く
既に保存されたマッピング (*.mfd) ファイルを開きます。使用中のMapForce エディションで使用できない機能が含まれているファイルは、開くことができません・

上書き保存
現在アクティブなファイル名で、現在アクティブになっているマッピングを保存します。

名前をつけて保存
現在アクティブになっているマッピングを、別の名前で保存します。また保存されていないファイルの場合、ファイルに新たな名前をつけることができます。

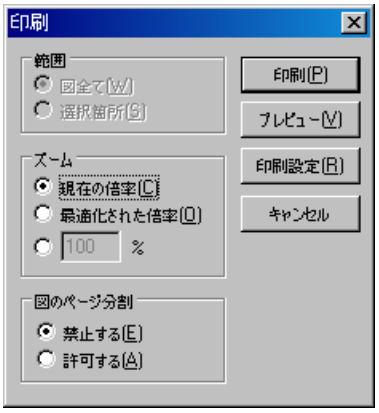
全て保存
現在開かれている全てのマッピングファイルを保存します。

再ロード
現在アクティブになっているマッピングを再ロードします。それまでに作成された変更点を破棄してもよいか尋ねられます。

閉じる
現在アクティブになっているマッピングを閉じます。ファイルを閉じる前に保存するか尋ねられます。

全て閉じる
現在開かれている全てのマッピングファイルを閉じます。保存されていないファイルを保存するか尋ねられます。

印刷
印刷ダイアログボックスが開かれ、マッピングを印刷することができます。



印刷ダイアログボックス

「現在の倍率」を選択すると、現在マッピングに対して指定されているズーム倍率が使用されます。 **「最適化された倍率」**を選択すると、マッピングがページの大きさに収められるよう、印刷が最適化されます。コンポーネントのスクロールバーは印刷されません。 1つのコンポーネントが複数のページに切り分けて印刷してもよいか指定することもできます。

印刷プレビュー
上に記述された機能を持った印刷ダイアログボックスが表示されます。

印刷設定
印刷設定ダイアログボックスが表示され、使用するプリンターや用紙の設定などを指定することができます。

マッピングの検証
マッピングを検証により、全てのマッピング（接続）が妥当であるかと、エラーや警告が表示されます。
詳細についてはマッピングの検証を参照ください。

マッピングの設定
マッピングの設定ダイアログボックスではドキュメントごとの設定を定義することができます（マッピングの設定の変更 を参照してください）。

選択された言語でコードを生成する
現在選択されているマッピングの言語によりコードが生成されます。現在選択されている言語は、ツールバーでハイライトされているプログラミング言語のアイコン（XSLT、XSLT2）で確認することができます。

コード生成 | XSLT (XSLT2)
このコマンドにより、ソースファイルからの変換に必要なXSLT ファイルが生成されます。このオプションを選択すると、フォルダーの参照ダイアログボックスが表示され、生成されたXSLT ファイルを配置する場所を指定することができます（マッピング設定の変更 を参照してください）。

最近のプロジェクト
最近開かれたプロジェクトのリストを表示します。

終了
アプリケーションを終了します。保存されていないドキュメントがある場合、それらを保存するか尋ねられます。

10.2 編集

このメニュー以下にあるコマンドの殆どは、**出力** タブにてマッピングの結果を参照している時、または XSLT タブにて XSLT コードのプレビューを行なっている際に利用することができます。

元に戻す

MapForce では使用回数に制限の無い「元に戻す」操作を行うことができ、マッピングのデザインステップを順を追って確認することができます。

やり直し

やり直しコマンドにより、元に戻すコマンドにより戻された動作を進めることができます。これらのコマンドを使用することで、操作の履歴を確認することができます。

検索

XSLT、XSLT2、または出力タブにて、指定されたテキストを検索することができます。

次を検索 F3

同じ検索文字列が次に出現する箇所を検索することができます。

前を検索 Shift F3

同じ検索文字列が前に出現する箇所を検索することができます。

切り取り/コピー/貼り付け/削除

標準的な Windows の編集コマンドです。マッピングウィンドウ内に表示されているコンポーネントや関数を切り取ったりコピーすることができます。

すべて選択

マッピングタブにある全てのコンポーネントを選択する、または XSLT、XSLT2、 XQuery、出力タブに表示されているテキスト/コード全体を選択することができます。

10.3 挿入

XML スキーマ / ファイル

XML スキーマまたはインスタンスをマッピングに追加します。スキーマを参照するXML ファイルを選択すると、追加情報をマッピングに追加する必要はありません。スキーマ参照のないXML ファイルを選択すると、自動的に一致するXML スキーマを生成するかを問われます。（XML スキーマの生成 を参照してください）。XML スキーマファイルを選択すると、プレビューのためのデータを提供するXML インスタンスを任意で追加するかを問われます。

入力の挿入

マッピングウィンドウがマッピングを表示すると、このコマンドにより入力コンポーネントをマッピングに追加することができます。簡単な入力を参照。マッピングウィンドウがユーザー定義関数を表示すると、このコマンドはユーザー定義関数の入力コンポーネントを追加します。複雑な入力コンポーネントの定義を参照。

出力の挿入

マッピングウィンドウがマッピングを表示すると、このコマンドにより出力コンポーネントをマッピングに追加することができます簡単な出力を参照。マッピングウィンドウがユーザー定義関数を表示すると、このコマンドはユーザー定義関数の出力コンポーネントを追加します。複雑な出力コンポーネントの定義を参照。

定数

常に同じ値を入力アイコンへ与える関数コンポーネントの定数を挿入します。定数コンポーネントを作成する際に表示されるダイアログボックスにてデータを入力します。定数関数には出力アイコンが 1 つだけ含まれています。データの型は、文字列、数値、その他から選択することができます。

変数

標準的な（非インライン型の）ユーザー定義関数と等価な中間変数を挿入します。変数はインスタンスファイルを持たない構造コンポーネントで、マッピング処理を単純にするために使用されます。詳細については中間変数 を参照ください。

並べ替え:node/row

ノードの並べ替えを許可するコンポーネントを挿入します。詳細に関しては、ノード行の並べ替えを参照してください。

フィルター:node/row

node/ row と**bool**、そして**on- true** と**on- false** という 2 つの入力と出力パラメーターを備えたコンポーネントを挿入します。Boolean がtrue となっている場合、node/row パラメーターの値がon-true パラメーターから渡され、Boolean がfalse となっている場合、node/row パラメーターの値がon-false パラメーターへ渡されます。フィルターの使用方法については フィルターと条件を参照してください。

Value-Map

ルックアップテーブルを使って入力された値を特定の値へ変換するコンポーネントを挿入します。コンポーネントには 1 つの入力ならびに出力アイテムが与えられています。詳細については Value-Map - 入力データの変換 を参照ください。

IF-Else 条件

"If-Else 条件 "のかたのコンポーネントを挿入します。詳細に関しては、フィルターと条件を参照してください。

10.4 コンポーネント

ルート要素の変更
XML インスタンスドキュメントのルート要素を変更することができます。

XMLSpy でスキーマ定義を編集
XML スキーマコンポーネントが選択された状態でこのオプションを選択すると、XMLSpy のスキーマビューで XML スキーマファイルが開かれ、編集を行うことができます

前に入力を複製して追加
選択されたアイテムのコピー /クローンを、現在選択されているアイテムの前に挿入します。コピーされたアイテムには出力アイコンが含まれておらず、データ構造として使用することはできません。この機能を使用した例については 複数のソースから1つのターゲットにマップ セクションを参照ください。複製されたアイテムを右クリックすると、コンテキストメニューから複製アイテムを移動することができます。必要に応じて複製されたアイテムを上 /下へ移動させてください。

後に入力を複製して追加
選択されたアイテムのコピー /クローンを、現在選択されているアイテムの後に挿入します。コピーされたアイテムには出力アイテムが含まれておらず、データ構造として使用することはできません。この機能を使用した例については 複数のソースから1つのターゲットにマップ セクションを参照ください。複製されたアイテムを右クリックすると、コンテキストメニューから複製アイテムを移動することができます。必要に応じて複製されたアイテムを上 /下へ移動させてください。

複製を削除
既に作成された複製アイテムを削除します。この機能を使用した例については 複数のソースから1つのターゲットにマップ セクションを参照ください。

ツリーを左揃えにする
全てのアイテムをコンポーネントの左端へ揃えます。

ツリーを右揃えにする
全てのアイテムをコンポーネントの右端へ揃えます。この表示方法はターゲットスキーマへのマッピングを行う際に利用することができます。

プロパティ
現在選択されているコンポーネントの設定を表示するダイアログボックスを表示します。 コンポーネント設定を変更するを参照してください。

10.5 接続

子要素の自動接続
子要素の自動接続機能を有効化または無効化します。ツールバーにあるアイコンからも選択を行うことができます。

子要素の接続設定
マッチする子要素の接続設定ダイアログボックスが表示され、接続設定を行うことができます。 (マッチする子要素の接続を参照)。

マッチする子要素の接続
このコマンドにより、ソースとターゲットスキーマ間で**同じ名前**を持った複数のアイテムに対して接続を作成することができます。このダイアログボックスで定義した設定はそのまま保持され、ツールバーにある**子要素の自動接続アイコン**  が有効になっている状態で 2 つのアイテムを接続した際に適用されます。このアイコンをクリックすることで、アイコンの有効/無効状態を切り替えることができます。詳細については マッチする子要素の接続のセクションを参照ください。

標準マッピング (ターゲット優先マッピング)
接続の種類を標準的なマッピングへ変更します。詳細については 標準マッピング (ターゲット優先マッピング) を参照してください。

全てコピー (子要素)
親コネクタのサブツリーとして表示されている子アイテムのコネクタが存在する状況で、マッチする全ての子要素に対してコネクタを作成します。詳細については すべてコピー接続 を参照ください。

ソース優先マッピング (混合コンテンツ)
接続をソース優先 (/混合コンテンツ)型のコネクタへ変更して、新たに要素を選択してマッピングすることができるようになります。詳細に関しては、 ソース優先マッピング (混合コンテンツ) を参照してください。

プロパティ
現在のコネクタに対して固有 (混合コンテンツ) の設定を定義するためのダイアログボックスが表示されます。利用することができないオプションはグレーアウトされます。これらの設定はテキストノードを持たない**複合型**のアイテムに対しても適用されます。詳細は接続設定を参照してください。

10.6 関数

ユーザー定義関数の作成：
新たなユーザー定義関数を作成します。 (詳細については ユーザー定義関数 を参照ください)。

選択からユーザー定義関数作成：
マッピングウィンドウにて現在選択されている要素をベースに、新たなユーザー定義関数を作成します。

関数設定：
現在アクティブなユーザー定義関数の設定ダイアログボックスを開き、現在の設定を変更することができます。

関数削除：
既存のユーザー定義関数を開き、その関数の名前がタブに表示されている状態で、現在アクティブなユーザー定義関数を削除します。

入力の挿入：
マッピングウィンドウがマッピングを表示すると、このコマンドは、マッピングに入力コンポーネントを追加します (詳細については 単純型入力 を参照ください)。マッピングウィンドウがユーザー定義関数を表示すると、このコマンドは、マッピングに入力コンポーネントをユーザー定義関数に追加します (詳細については 複合型の入力コンポーネントの定義 を参照ください)。

出力の挿入：
マッピングウィンドウがマッピングを表示すると、このコマンドは、マッピングに出力コンポーネントを追加します (詳細については 単純型出力 を参照ください)。マッピングウィンドウがユーザー定義関数を表示すると、このコマンドは、マッピングに出力コンポーネントをユーザー定義関数に追加します (詳細については 複合型の出力コンポーネントの定義 を参照ください)。

10.7 出力

(**XSLT 1.0、XSLT 2.0 などの**)最初のオプショングループにより、コードを生成する際の ターゲット言語を定義 することができます。

出力ファイルの検証
参照されているスキーマに対して出力 XML ファイルを検証します。

生成された出力の保存
出力タブに現在表示されているデータを保存します。

生成されたすべての出力を保存
動的マッピングにより生成された出力をファイルに保存します。詳細については 複数の入力または出力ファイルを動的に処理 を参照ください。

出力を再生成
現在のマッピングを出力ウィンドウから再生成します。

ブックマークの挿入 /削除
出力ウィンドウのカーソル位置にブックマークを挿入します

次のブックマーク
出力ウィンドウ内にある次のブックマークへ移動します。

前のブックマーク
出力ウィンドウ内にある前のブックマークへ移動します。

全てのブックマークを削除
出力ウィンドウにて現在定義されている全てのブックマークを削除します。

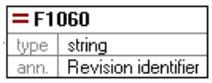
XML テキストの整形
出力ペインに表示されているXML ドキュメントの表示を、ドキュメントの構造に従ったかたちに再構成します。各子ノードが、親ノードよりインデントされたかたちで表示されます。この機能では、タブグループにて指定されたタブサイズの設定（タブまたはスペースを挿入）が使用されます。

テキストビューの設定
テキストビュー設定ダイアログボックスを表示します。このダイアログボックスにより**出力** ペインと**XSLT** ペインでテキストビュー設定をカスタマイズすることができます。またこのダイアログボックスは、現在ウィンドウ内で適用されている定義済みのホットキーを表示しています。 詳細に関しては、次を参照してください テキストビュー機能。

10.8 表示

注釈の表示

XML スキーマの注釈 をコンポーネントウィンドウにて表示します。
型の表示アイコンも有効になっている場合、両方の情報がグリッド形式で表示されます。



型の表示

各要素または属性に対してスキーマのデータ型を表示します。
注釈の表示アイコンも有効になっている場合、両方の情報がグリッド形式で表示されます。

関数ヘッダーにライブラリ名を表示

関数タイトルは括弧付きでライブラリ名を表示します。

ヒントの表示

マウスポインターが関数の上に配置された時に、ツールチップ内に関数の説明が表示されます。

選択されたコンポーネントの接続線の表示

すべてのマッピングの接続線または、現在選択されているコンポーネントに関連した接続線を切り替えることができます。

ソースからターゲットへの接続線表示

以下のように接続線の表示を切り替えることができます：

- 現在選択されているコンポーネントに**直接**接続されているコネクタ
- 現在選択されているコネクタで、ソースからターゲットコンポーネントまで

ズーム

ズームダイアログボックスが表示されます。ズームの倍率を手動で入力することができるほか、スライダーをドラッグして、ズームの倍率を変更することができます。

戻る

マッピングタブにて現在開かれているマッピングで、前に表示されていたマッピングを表示します

進む

マッピングタブにて現在開かれているマッピングで、戻るコマンドにより表示される前のマッピングを表示します。

ステータスバー

メッセージウィンドウの下に表示されているステータスバーを表示/非表示にします。

ライブラリウィンドウ

ライブラリ関数が含まれるライブラリウィンドウを表示/非表示にします。

メッセージウィンドウ

検証出力ウィンドウを表示/隠します。コードの生成を行う場合、メッセージ出力ウィンドウが自動的に有効になり、検証結果が表示されます。

概要ウィンドウ

概要ウィンドウを表示/非表示にします。長方形をドラッグすることで、マッピングの表示範囲を移動することができます

10.9 ツール

グローバルリソース
グローバルリソースの管理ダイアログボックスを表示して、グローバルリソースXML ファイル内にあるグローバルリソース情報を追加、編集、または削除することができます。詳細については グローバルリソース - プロパティを参照ください。

アクティブな構成
現在アクティブになっているグローバルリソースを、グローバルリソースの構成リストから選択/変更することができます。サブメニューから目的の構成を選択してください。

反転マッピングの作成
MapForce で現在アクティブになっているマッピングをベースに、「反転した」マッピングを作成します。反転した結果生成されるマッピングは完成したものではなく、コンポーネント間における直接接続だけが保持されるという点に注意してください。恐らく反転したマッピングは妥当なものではなく、更なる編集を行うこと無く出力タブをクリックしても、実行されません。

マッピングを反転させると、ソースコンポーネントがターゲットコンポーネントになり、ターゲットコンポーネントがソースコンポーネントになります。入力または出力 XML インスタンスファイルがコンポーネントに割り当てられている場合、これらも交換されます。

以下のデータが保持されます：

- コンポーネント間の直接接続
- チェーンマッピング内のコンポーネント間の直接接続
- 接続の種類：標準、混合コンテンツ、全てコピー
- パススルーコンポーネント設定
- データベースコンポーネント

以下のデータが保持されません：

- 関数やフィルターなどを経由した接続は、関数などとともに削除されます
- ユーザー定義関数
- Web サービスコンポーネント

ツールバーとウィンドウの復元
ツールバーやドックウィンドウなどをデフォルトの状態にリセットします。変更を反映するには、MapForce を再起動する必要があります。

カスタム化 ...
MapForce グラフィカルユーザーインターフェイスをカスタム化することのできるダイアログボックスを開きます。これはツールバーの表示と非表示、またコンテキストメニューの編集とキーボードのショートカットを含みます（キーボードショートカットのカスタム化 を参照してください）。

オプション
デフォルトのMapForce 設定を変更することのできるダイアログボックスを開きます（MapForce オプションの変更 を参照してください）。

10.10 ウィンドウ

重ねて表示
開かれている全てのウィンドウを重ねて表示されるように再配置します。

上下に並べて表示
開かれている全てのウィンドウを**上下に並べて表示**することで、同時に表示されるように再配置します。

左右に並べて表示
開かれている全てのウィンドウを**左右に並べて表示**することで、同時に表示されるように再配置します。

1
2
現在開かれているウィンドウが表示され、これらウィンドウ間で素早い切り替えを行うことができます。
Ctrl + tab または Ctrl + F6 キーボードショートカットを使用することでも、開かれているウィンドウを切り替えることができます。

10.11 ヘルプメニュー

目次

説明

ヘルプウィンドウの左側のペインに目次を表示した、MapForce の画面上のヘルプマニュアルを開きます。目次はヘルプドキュメント全体の概要を表示しています。目次のエントリをクリックしてトピックに移動することができます。

インデックス

説明

ヘルプウィンドウの左側のペインにキーワード インデックスを表示したMapForce の画面上のヘルプマニュアルを開きます。目次はヘルプドキュメント全体の概要を表示しています。インデックスはキーワードをリストし、キーワードをダブルクリックすることでトピックへ移動することができます。キーワードが1つ以上のトピックにリンクされている場合は、トピックのリストが表示されます。

検索

説明

ヘルプウィンドウの左側のペインに検索ダイアログを表示したMapForce の画面上のヘルプマニュアルを開きます。単語を検索するには、入力フィールドに検索対象を入力して、**[Return]** を押します。ヘルプシステムは、ヘルプドキュメント全体で全文検索を行いヒットしたリストを返します。アイテムを表示するためにはアイテムをダブルクリックします。

ライセンス登録

説明

Altova 製品ソフトウェアをダウンロードすると、無料評価キーまたは購入されたライセンスキーを使用して、製品のアクティベーションを行うことができます。

- ***無料評価キー:*** 初めて製品のダウンロードとインストールを行うと、ソフトウェアアクティベーションダイアログが表示されます。ダイアログでは無料評価キーコードをリクエストすることができます。ユーザーの名前、所属会社名、そして電子メールアドレスを入力して、無料評価キーを取得するボタンをクリックしてください。入力された電子メールアドレスに評価キーが送信されます。ソフトウェアアクティベーションのダイアログボックスにで送信されたキーコードを入力し、**[OK]** をクリックすることで、Altova 製品を使用することができるようになります。ソフトウェアは30日の間アンロックされます。
- ***購入されたライセンスキー:*** ソフトウェアアクティベーションダイアログにはライセンスキーのキーコードを購入するボタンが含まれています。このボタンをクリックするとAltova のオンラインショップが表示され、製品に対応したライセンスキーを購入することができます。ライセンスにはシングルユーザーとマルチユーザーという二種類のライセンスがあります。両種類のキーコードが電子メールにより送信されます。シングルユーザーライセンスにはライセンスデータや組織名、そしてキーコードが含まれています。ライセンス契約では、ライセンスされている数を超えて組織内のコンピューターにAltova 製品をインストールできないということに注意してください。電子メール内に含まれているライセンス情報を、登録ダイアログに正確に入力するようご注意してくだ

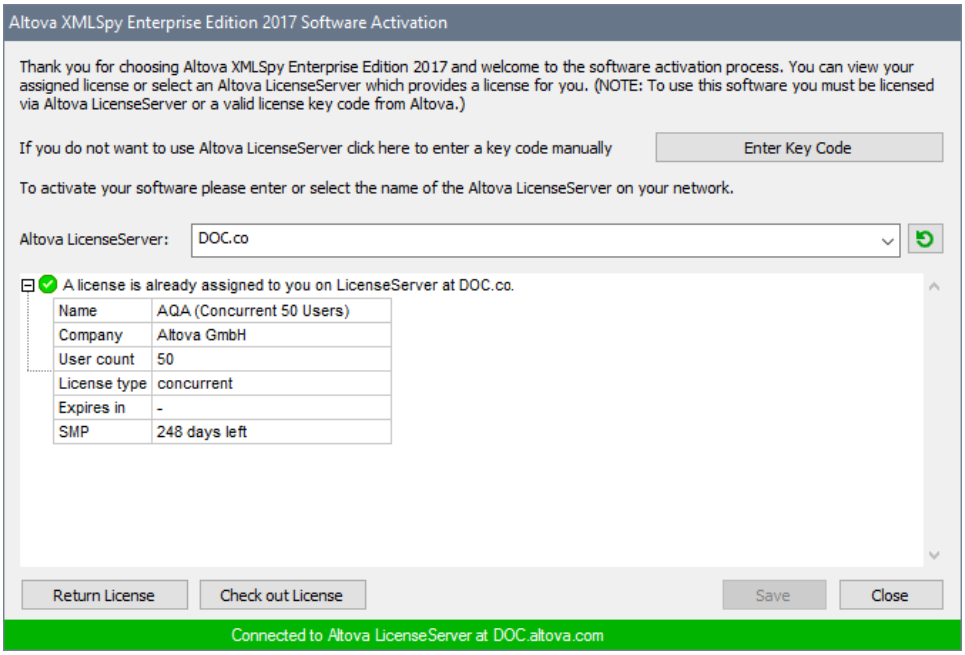
さい。

メモ: ソフトウェアアクティベーションダイアログにてライセンス情報を入力する際には、ライセンス電子メールに記された情報を正確に入力するようにしてください。マルチユーザーライセンスの場合、Name フィールドに、各ユーザーの名前を入力してください。

ソフトウェアアクティベーションダイアログ (*下のスクリーンショット*) は、**「ヘルプ | ソフトウェア アクティベーション」**をクリックすることにより常にアクセスすることができます。

以下の方法によりソフトウェアをアクティブ化することができます :

- (**「新しいキーコードを入力する」**をクリックして) ライセンスキーの情報を入力します。または、
- (**「Altova LicenseServer を使用」**をクリックして) ネットワーク上の Altova LicenseServer からライセンスを取得します。Altova LicenseServer は、使用中の Altova 製品のためのライセンスをライセンスプール上に有する必要があります。ライセンスが LicenseServer プール内に存在する場合、Software Activation ダイアログ内に表示されます (*下のスクリーンショット*)。**「保存」**をクリックしてライセンスを取得します。



ライセンスが1度取得されると、7日間は、LicenseServer に戻すことができません。7日過ぎると、(**「ライセンスを戻す」**をクリックして) ライセンスを戻すことができ、ライセンスは、他のクライアントから取得することができます。LicenseServer 管理者は、LicenseServer の Web UI を使用して、取得されたライセンスの割り当てを解除することができます。

ライセンスのチェックアウト

ライセンスが製品マシン上に保管されるように、ライセンスをライセンスプールから 30 日間チェックアウトすることができます。これにより、オフラインで作業することが可能になります。これはとても役に立ちます。Altova LicenseServer にアクセスできない環境 (例えば、旅行中に Altova 製品がインストールされたラップトップコンピューターで作業する) 場合などが挙げられます。ライセンスはチェックアウトされていますが、LicenseServer は、ライセンスが使用中と表示し、ライセンスは他のマシンで使用することができません。ライセンスはチェックアウトの期間が終わると自動的にチェックインされた状態を戻します。または、チェックアウトされたライセンスはソフトウェアのライセンスの認証ダイアログのボタンを使用して**チェックイン**することで

きます。

ライセンスをチェックアウトするには以下をおこないます：(i) ソフトウェアのライセンスの認証ダイアログで **ライセンスのチェックアウト** をクリックします（*上のスクリーンショット参照*）。 (ii) ライセンスのチェックアウトダイアログ内から、チェックアウトの期間を選択し、**チェックアウト**をクリックします。ライセンスがチェックアウトされます。ソフトウェアのライセンスの認証ダイアログは、チェックアウト期間の終わりを含むチェックアウトの情報を表示します。ダイアログ内の**ライセンスのチェックアウト** ボタンは、**チェックイン**ボタンに変わります。**チェックイン**ボタンをクリックして、ライセンスをチェックインすることができます。ライセンスは自動的にチェックイン状態に戻されるため、選択したチェックアウトの期間がオフラインで作業する期間をカバーするようご確認ください。

Altova LicenseServer を使用することにより、IT 管理者は、リアルタイムでネットワーク上の全てのライセンスの概要、および、クライアントの割り当てと、クライアントのライセンスの使用状況を確認することができます。LicenseServer を使用する利点は、ですから、多数の Altova ライセンスを管理することのできる管理機能です。 Altova LicenseServer は、Altova Web サイトで無料で提供されています。Altova LicenseServer および Altova LicenseServer を使用したライセンスの供与に関する詳細は、Altova LicenseServer ドキュメントを参照してください。

注文フォーム

説明

ソフトウェア製品のライセンスを購入する準備が整った場合、ソフトウェアアクティベーションダイアログにて **[キーコードを購入する]** ボタンを選択するか、**[ヘルプ | 注文フォーム]** コマンドを選択して、Altova オンラインショップへアクセスすることができます。

登録

説明

Altova 製品登録ページをブラウザーのタブに表示します。Altova ソフトウェアを登録することにより、最新の製品の情報が得られます。

最新情報のチェック

説明

Altova サーバーに接続して、より新しいバージョンの製品が利用可能かどうかチェックし、その結果を表示します。

サポートセンター

説明

インターネット上にある Altova サポートセンターへのリンクとなっています。サポートセンターには FAQ やディスカッ

ションフォーラムが含まれており、問題の解決方法を探ったり、Altova の技術サポートスタッフへアクセスすることができます（現在英語のみの提供となります）。

WEB 上の FAQ

説明

インターネット上にある Altova の FAQ へのリンクとなっています。FAQ データベースは Altova のサポートスタッフにより常時更新されています。

コンポーネントのダウンロード

説明

インターネット上にある Altova のコンポーネントダウンロードセンターへのリンクとなっています。このリンク先から様々なコンポーネントソフトウェアをダウンロードして、Altova 製品とともに使用することができます。ソフトウェアコンポーネントは XSLT や XSL-FO プロセッサーからアプリケーションサービスプラットフォームまで、幅広く提供されています。コンポーネントダウンロードセンターにてご利用になれるソフトウェアは、通常無料でご利用になれます。

インターネット上の MapForce

説明

インターネット上にある Altova ウェブサイトへのリンクとなっています。Altova ウェブサイトでは、MapForce や関連するテクノロジーについて確認することができます。

MapForce について

説明

スプラッシュ画面と製品のバージョン番号が表示されます。IMapForce の 64 ビットバージョンを使用している場合、アプリケーション名の後のサフィックス (x64) を示しています。32 ビットバージョンにはサフィックスはありません。

Chapter 11

付録

11 付録

以下の付録にはMapForce に関する技術的な情報や、ライセンスに関する重要な情報が収められています。各付録には以下のようにサブセクションが収められています：

技術データ

- OS ならびにメモリの要件
- Altova XML パーサー
- Altova XSLT とXQuery エンジン
- Unicode のサポート
- インターネットの使用
- ライセンス使用状況測定

ライセンス情報

- 電子的なソフトウェアの配布
- 著作権
- 使用許諾契約書

11.1 エンジン情報

このセクションには Altova XML バリデーター、Altova XSLT 1.0 エンジン、Altova XSLT 2.0 エンジン、そして Altova XQuery エンジンの実装に特化した情報が含まれます。

11.1.1 XSLT および XQuery エンジンに関する情報

MapForce の XSLT および XQuery エンジンは、W3C 仕様に従っています、ですから、XMLSpy の以前のバージョン内の Altova エンジンよりも厳密です。この結果、以前のエンジンで無視されていた小さなエラーが、MapForce によりエラーとして挙げられます。

例えば:

- パス演算子の結果がノードと非 -ノードを両方含む場合、型エラー (`err:XPTY0018`) です。
- パス式 `E1/E2` 内の `E1` がノードのシーケンスを評価しない場合、型エラー (`err:XPTY0019`) です。

この種類のエラーが発生した場合、XSLT/XQuery ドキュメントまたはインスタンスドキュメントを必要に応じて修正してください。

このセクションは、エンジンの実装固有の機能を仕様別に整理して説明します。

- XSLT 1.0
- XSLT 2.0
- XQuery 1.0

11.1.1.1 XSLT 1.0

MapForce の XSLT 1.0 エンジンは、World Wide Web Consortium (ワールド・ワイド・ウェブ・コンソーシアム) (W3C) の1999 年 11月 16日版の XSLT 1.0 勧告 および1999 年 11月 16日版の XPath 1.0 勧告 に準拠します。実装に関しての以下の情報に注意してください。

実装についての注意点

`xsl:output` の`method` 属性がHTMLに設定された場合、または、がHTML 出力 デフォルトで選択されている場合、内の特殊文字はHTML ドキュメントにHTML 文字参照として出力内に挿入されます。例えば、文字 U+00A0 (ブレーク無しのスペースのための 16進数レファレンス)がHTML コード内に文字の参照 (` ` or ` `) 、または、エンティティ参照 ` ` として挿入されます

11.1.1.2 XSLT 2.0

このセクション:

- エンジン適合性
- 下位互換性
- 名前空間
- スキーマ認識
- 実装固有の振る舞い

適合性

MapForce のXSLT 2.0 エンジンは、World Wide Web Consortium (ワールド・ワイド・ウェブ・コンソーシアム) (W3C) の2007 年 1月 23日版の XSLT 2.0 勧告 および2010 年 12月 14日版の XPath 2.0 勧告に準拠します。

下位互換性

XSLT 2.0 エンジンは下位互換性を有します。XSLT 2.0 エンジンの下位互換性が有効になるのは、XSLT 1.0 スタイルシートを処理するために XSLT 2.0 エンジン が使用される際です。XSLT 1.0 エンジンと下位互換性を持つXSLT 2.0 エンジンにより作成される出力に違いがあるかもしれないことに注意してください。

名前空間

XSLT 2.0 スタイルシートは、XSLT 2.0. プレフィックス内で使用することのできる型コンストラクタおよび関数を使用するため、以下の名前空間を宣言する必要があります。下のリストは通常使用されるリストです。希望する場合は、代替プレフィックスを使用することもできます。

名前空間	プレフィックス	名前空間 URI
XML スキーマ型	`xs:`	`http://www.w3.org/2001/XMLSchema`
XPath 2.0 関数	`fn:`	`http://www.w3.org/2005/xpath-functions`

通常これらの名前空間は、以下のリストで表示されるように`xsl:スタイルシート`または、`xsl:transform` 要素で宣言されます:

```
<xsl:スタイルシート 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:スタイルシート>
```

次の点に注意してください:

- XSLT 2.0 エンジンは、(上のテーブルでリストされている)XPath 2.0 およびXQuery 1.0 関数 名前空間 を**デフォルトの関数名前空間**として使用します。XPath 2.0 およびXSLT 2.0 関数をプレフィックス無しでスタイルシート内で使用することができます。XPath 2.0 関数 名前空間 をスタイルシート内でプレフィックスと共に宣言すると、割り当てられた宣言内でプレフィックスを追加して使用することができます。t
- XML スキーマ名前空間から型コンストラクタ型を使用する場合、名前空間 宣言内で使用された、プレフィックスを使用して型コンストラクタを呼び出さなければなりません (例えば、`xs:date`)。
- XPath 2.0 関数の一部は、XML スキーマデータ型と同じ名前を保有します。例えば、XPath 関数 `fn:string` および `fn:boolean` のためには、同じローカル名 :`xs:string` および`xs:boolean` を持つ XML スキーマデータ型 が存在します。ですから、XPath 式 `string('Hello')`を使用する場合、式は、`xs:string('Hello')`ではなく`fn:string('Hello')` として検証します。

スキーマ認識
XSLT 2.0 エンジンは、スキーマを認識します。ですから、ユーザー定義 スキーマ型 および `xsl:validate` 命令を使用することができます。

実装固有の振る舞い
以下は、XSLT 2.0 エンジンが、特定のXSLT 2.0 関数の振る舞いの実装 特定のアスペクトをどのように扱うかの説明です。

`xsl:result-document`
追加してサポートされるエンコードは以下の通りです (Altova-固有): `x-base16tobinary` および `x-base64tobinary`.

`function-available`
インスコープ関数の使用をテストする関数 (XSLT、XPath、および拡張関数).

`unparsed-text`
`href` 属性は、以下を受け入れます (i) ベースuri フォルダー内のファイルの相対パス、および (ii) 相対パスを持つまたは持たない `file://` プロトコール。追加してサポートされるエンコードは以下の通りです (Altova-固有): `x-binarytobase16` および `x-binarytobase64`.

`unparsed-text-available`
`href` 属性は、以下を受け入れます (i) ベースuri フォルダー内のファイルの絶対パス、および (ii) 絶対パスを持つまたは持たない `file://` プロトコール。追加してサポートされるエンコードは以下の通りです (Altova-固有): `x-binarytobase16` および `x-binarytobase64`.

メモ: RaptorXML の先行製品であるAltovaXML で実装されていた以下のエンコード値は使用しないでください: `base16tobinary`, `base64tobinary`, `binarytobase16` and `binarytobase64`.

11.1.1.3 XQuery 1.0

このセクション:

適合性

MapForce のXQuery 1.0 エンジンは、World Wide Web Consortium (ワールド・ワイド・ウェブ・コンソーシアム) (W3C) の2010 年 12月 14日版の XQuery 1.0 勧告に準拠します。XQuery 標準は、多数の機能の実装についての裁量を提供します。下には、XQuery 1.0 エンジンがどのようにこれらの機能を実装するかについて説明するリストが下に挙げられています。

スキーマ認識

XQuery 1.0 エンジンはスキーマを認識します。

エンコード

UTF-8 およびUTF-16 文字のエンコードは、サポートされています。

名前空間

以下の名前空間 URI と関連するバインドはは定義済みです。

名前空間	プレフィックス	名前空間 URI
XML スキーマ型	`xs:`	`http://www.w3.org/2001/XMLSchema`
スキーマインスタンス	`xsi:`	`http://www.w3.org/2001/XMLSchema-instance`
内蔵の 関数	`fn:`	`http://www.w3.org/2005/xpath-functions`
Local 関数	`local:`	`http://www.w3.org/2005/xquery-local-functions`

次の点に注意してください:

- XQuery 1.0 エンジンは、上にリストされたプレフィックスを名前空間に対応するバウンドとして認識します。
- Since the 上にリストされた内蔵の関数 名前空間は、XQuery内のデフォルトの関数です。内蔵の関数 が呼び出される際、名前空間、fn:プレフィックス を使用する必要はありません。 (例えば、`string("Hello")` が`fn:string` 関数を呼び出す場合。) しかし、プレフィックス`fn:` は、クエリプロローグ内で名前空間を宣言することなく内蔵の関数を呼び出す時に使用することができます。 (サンプル:`fn:string("Hello")`).
- クエリプロローグ内で`default function` 名前空間 式を宣言することにより、デフォルトの関数 名前空間をすることにより変更することができます。
- XML スキーマ名前空間空の型を使用する場合、プレフィックス`xs:` は、名前空間を明確に宣言することなく、また、これらのプレフィックスをクエリプロローグ内でバインドすることなく使用することができます。 (サンプル:`xs:date` および`xs:yearMonthDuration`。) XML スキーマ名前空間のために、他のプレフィックスを使用する場合は、クエリプロローグ内で明確に宣言されている必要があります。 (サンプル:`declare 名前空間 alt = "http://www.w3.org/2001/XMLSchema"; alt:date("2004-10-04")`.)
- `untypedAtomic`, `dayTimeDuration`, および`yearMonthDuration` データ型 が 23January 2007 のCR 共に、XPath データ型 名前空間からXML スキーマ名前空間へ移動されていることに注意してください。ですから以下となります :`xs:yearMonthDuration`。

関数のための名前空間、型コンストラクタ、ノードテスト、が間違って割り当てられている場合、エラーが発生します。しかし、一部の関数はスキーマデータ型と同じ名前を持つことに注意してください。 例 :`fn:string` および `fn:boolean`。(`xs:string` および `xs:boolean` は宣言されています)名前空間 プレフィックス、関数または型コンストラクタが使用されるか決定します。

XML ソースドキュメントと検証

XML ドキュメント used in executing an XQuery ドキュメント with XQuery 1.0 エンジン must be 整形式 . しかし、they do not need to be valid according to an XML スキーマ .If the ファイル is not valid, the invalid ファイル is loaded without schema information. XML ファイル が外部スキーマと関連付けられ、また有効な場合、then post-schema 検証情報 is generated for XML データ and will be used for クエリ検証 .

静的および動的な型のチェック

静的分析フェーズ checks aspects of クエリ such as syntax, whether 外部レファレンス (例 for モジュール) exist, whether invoked 関数と変数 are defined, and so on. If an error is detected in 静的分析フェーズ, it is reported and the execution is stopped.

動的な型チェック is carried out ランタイム中 ,when クエリ is actually executed. If a type is incompatible with the requirement of an operation, an error is reported. 例えば、式 `xs:string("1") + 1` は、エラーを返します。because the addition operation cannot be carried out on an operand of type `xs:string`.

ライブラリモジュール

ライブラリモジュール store 関数と変数 so they can be reused. XQuery 1.0 エンジン supports モジュール that are stored in **a single external XQuery ファイル**. Such モジュール ファイル must contain モジュール 宣言 in its prolog, which associates a target 名前空間 .以下はモジュールサンプルです :

```
module 名前空間 libns="urn:module-library";
declare variable $libns:company := "Altova";
declare function libns:webaddress() { "http://www.altova.com" };
```

すべての関数および変数は、モジュールに関連した名前空間に属するモジュール内で宣言されています。.モジュール is used by importing it into an XQuery ファイル with the `import module` statement クエリプロローグ内の . `import module` ステートメントは、ライブラリモジュールファイル内で直接宣言された、関数と変数のみをインポートします。例 :

```
import module namespace modlib = "urn:module-library" at
   "modulefilename.xq";
if  ($modlib:company = "Altova")
then        modlib:webaddress()
else        error("No match found.")
```

外部関数

外部関数 は、サポートされていません。i.e. in those 式 using the `external` keyword, as in:

```
declare function hoo($param as xs:integer) as xs:string external;
```

照合順序
デフォルトの照合順序 is the Unicode-コードポイント照合順序 ,which compares strings on the basis of their Unicode コードポイント .その他にサポートされる照合順序 are the ICU 照合順序 listed here. To use a specific 照合順序 ,supply its URI as given in the list of supported 照合順序. Any string comparisons, including for the `fn:max` and `fn:min` 関数 ,will be made according to the specified 照合順序 .照合順序オプションが指定されていない場合、デフォルトのUnicode-コードポイント照合順序 が使用されます。

数値データの精度

- The `xs:integer` データ型 is arbitrary-精度 ,i.e. it can represent any number of 桁 .
- The `xs:decimal` データ型 has a limit of 20 桁 after the decimal point.
- The `xs:float` and `xs:double` データ型 have limited-精度 of 15 桁 .

XQuery 命令サポート
`Pragma` 命令 は、サポートされていません。発生した場合、無視されフォールバックの式が検証されます。

11.1.2 XSLT と XPath/ XQuery 関数

このセクションでは、XPath および/または XQuery 式で使用することのできる、Altova 拡張関数と他の拡張関数をリストします。Altova 拡張関数は Altova の XSLT および XQuery エンジンで使用することができ、W3C 標準で定義された関数ライブラリで使用することのできる機能に追加して機能を提供します。

一般的な情報
以下の一般的な情報に注意してください:

- W3C 仕様により定義されているコア関数ライブラリの関数は、関数の呼び出しにプレフィックスは必要ありません。これは、XSLT および XQuery エンジンが、 XPath/XQuery 関数仕様で指定されている `http://www.w3.org/2005/xpath-functions` プレフィックス無しの関数をデフォルト関数の名前空間に属するものとして読み込むためです。
- 関数において、各アイテムが引数となるようなシーケンスが期待されており、2つ以上のアイテムがシーケンスにより呼び出された場合、エラーが返されます。
- 全ての比較は Unicode コードポイントコレクションを使用することで行われます。
- QName の結果は `[prefix:]localname` という形式でシリアル化されます。

xs:decimal の精度

精度とは、数値内にある桁数のことで、仕様では少なくとも18桁が求められます。xs:decimal 型に結果が収められる除算の場合、端数処理を行うことなく精度は小数点以下の19桁になります。

黙示的なタイムゾーン

2つのdate、time、またはdateTime 値を比較する場合、比較する値のタイムゾーンを明らかにする必要があります。値の中にタイムゾーンが明示的に与えられていない場合、黙示的なタイムゾーンが使用されます。黙示的なタイムゾーンはシステムクロックから取得されimplicit-timezone() 関数によりその値をチェックすることができます。

照合順序

デフォルトの照合順序は、Unicode コードポイントをベースに文字列を比較するUnicode コードポイント照合順序です。エンジンはUnicode 照合アルゴリズムを使用しています。他のサポートされる照合順序は下にリストされるICU 照合順序 です。使用するには、サポートされる照合順序のリストのURI を提供してください (*下のテーブル*)。max とmin 関数を含む、文字列の比較は、指定された照合順序に沿って行われます。照合順序オプションが指定されていない場合、デフォルトのUnicode コードポイント照合順序が使用されます。

言語	URI
da: デンマーク語	da_DK
de: ドイツ語	de_AT, de_BE, de_CH, de_DE, de_LI, de_LU
en: 英語	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: スペイン語	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: フランス語	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: イタリア語	it_CH, it_IT
ja: 日本語	ja_JP
nb: ノルウェー語 (ブークモール)	nb_NO
nl: オランダ語	nl_AW, nl_BE, nl_NL
nn: ノルウェー語 (ニーノシュク)	nn_NO
pt: ポルトガル語	pt_AO, pt_BR, pt_GW, pt_MZ, pt_PT, pt_ST
ru: ロシア語	ru_MD, ru_RU, ru_UA
sv: スウェーデン語	sv_FI, sv_SE

名前空間軸

名前空間軸はXPath 2.0 にて廃止されましたが、名前空間軸の使用はサポートされています。XPath 2.0 *メカニズム*により名前空間情報へアクセスするには、in-scope-prefixes()、namespace-uri()、namespace-uri-

for-prefix(関数を使用してください。

11.1.2.1 Altova 拡張関数

Altova 拡張関数はXPath/XQuery 式で使用することができ、XPath、XQuery、およびXSLT 関数の標準ライブラリで使用可能な機能に更なる機能性を与えます。Altova 拡張関数は**Altova 拡張関数名前空間、** `http://www.altova.com/xslt-extensions` に収められており、`altova:` プレフィックスが、このセクションでは使用されます。製品の今後のバージョンが拡張機能への継続的サポート、または個別の関数の振る舞いは変更する可能性があることに注意してください。Altova 拡張機能へのサポートに関しては、今後のリリースのドキュメントを参照してください。

W3C のXPath/XQuery 関数仕様で定義された関数は、以下で使用することができます：(i) XSLT 子アンテキスト内のXPath 式と、(ii) XQuery 文書内のXQuery 式。このドキュメントでは、前者 (XSLT 内のXPath) のコンテキストで使用することのできる関数を、`XP` シンボルと共に表示し、と称します。後者 (XQuery) で使用することのできる関数はの`XQ` シンボルと共に表示され、XQuery 関数と共に作業することができます。W3C のXSLT 仕様は、XPath/XQuery 関数の仕様ではなく、XSLT 文書内のXPath 式でも使用することのできる関数を定義します。これらの関数は、`XSLT` シンボルと共に表示され、XSLT 関数と称されます。関数を使用することのできるXPath/XQuery およびXSLT のバージョンは、関数の詳細に記載されています (*下のシンボルを参照してください*)。XPath/XQuery およびXSLT 関数ライブラリからの関数は、*プレフィックス*無しでリストされています。Altova 拡張関数などの、他のライブラリからの関数は*プレフィックス*と共にリストされています。

XPath 関数 (XSLT 内のXPath 式で使用):	`XP1` `XP2` `XP3`
XSLT 関数 (XSLT 内のXPath 式で使用):	`XSLT1` `XSLT2` `XSLT3`
XQuery 関数 (XQuery 内のXQuery 式で使用):	`XQ1` `XQ3`

XSLT 関数
XSLT 関数はXSLT 2.0 の`current-group()` や`key()` 関数と同様に、XSLT コンテキストにて使用することができます。(例えば、XQuery コンテキストなどの)非 XSLT コンテキストでは使用することができません。XBRL に対するXSLT 関数は、XBRL をサポートするエディションのAltova 製品でのみ使用することができます。

XPath/XQuery 関数
XPath/XQuery 関数は、XSLT コンテキスト、XQuery 関数のXPath 式で使用することができます：

- 日付 時刻
- 位置情報
- イメージに関連した
- 数値
- シーケンス
- 文字列
- その他

XSLT 関数

XSLT 拡張関数 はXSLT コンテキスト内のXPath 式にて使用することができます。 (例えば、XQuery コンテキストなどの)非 XSLT コンテキストでは使用することができません。

関数の名前指定と言語の適用性

Altova 拡張関数はXPath/XQuery 式で使用することができ、XPath、XQuery、およびXSLT 関数の標準ライブラリで使用可能な機能に更なる機能性を与えます。Altova 拡張関数は**Altova 拡張関数名前空間、** `http://www.altova.com/xslt-extensions` に収められており、`altova:` プレフィックスが、このセクションでは使用されます。製品の今後のバージョンが拡張機能への継続的サポート、または個別の関数の振る舞いは変更する可能性があることに注意してください。Altova 拡張機能へのサポートに関しては、今後のリリースのドキュメントを参照してください。

XPath 関数 (XSLT 内のXPath 式で使用):	XP1 XP2 XP3
XSLT 関数 (XSLT 内のXPath 式で使用):	XSLT1 XSLT2 XSLT3
XQuery 関数 (XQuery 内のXQuery 式で使用):	XQ1 XQ3

標準関数

distinct-nodes [altova:]

altova:distinct-nodes(*node()**)** を**node()*** とする XSLT1 XSLT2 XSLT3
入力として1つ以上のノードを必要とし、同じセットから重複した値を持つノードを除いたノードを返します。XPath/XQuery 関数 `fn:deep-equal` を使用して比較を行うことができます。

サンプル

- **altova:altova:distinct-nodes**(country)は重複した値を持つものを除く、全ての子 `country` ノード返します。

evaluate [altova:]

altova:evaluate(XPathExpression *as xs:string***[, ValueOf$p1, ... ValueOf$pN])**
XSLT1 XSLT2 XSLT3
XPath 式を必要とし、必須引数として文字列をパスします。評価された式の出力を返します。例えば:
altova:evaluate('//Name[1]') は、ドキュメント内の最初の`Name` 要素のコンテンツを返します。式 `//Name[1]` は、一重引用符を使用することにより、文字列としてパスされます。

`altova:evaluate` 関数は、オプションとして追加の引数を持つことができます。これらの引数は、`p1`, `p2`, `p3`... `pN` の名前を持つスコープ内の変数の値です。使用に関して以下の点に注意してください: (i) 変数は、`X` が整数である箇所のフォーム`pX` の名前と共に定義される必要があります。(ii) `altova:evaluate` 関数の引数は、(*上の署名参照*) 2 番目の引数からは、数値順の変数のシーケンスに対応した引数のシーケンス変数の値を与えます: `p1` to `pN`: 第 2引数は変数 `p1` の値で、第 3引数は、変数 `p2` の値です。 (ii) 変数の値は型 `item*` である必要があります。

サンプル

```
<xsl:variable name="xpath" select="'$p3, $p2, $p1'" />
<xsl:value-of select="altova:evaluate($xpath, 10, 20, 'hi')" />
```

outputs `"hi 20 10"`

上のリストに関して、以下の点に注意してください:

- altova:evaluate 式の第 2引数は、変数 $p1 に割り当てられた値で、第三の引数は変数 $p2 に割り当てられた値です。
- 関数の第 4番目の引数は、引用符による囲いで表示された文字列の値です。
- xs:variable 要素の select 属性は、XPath 式を提供します。この式は xs:string の型である必要があり、一重引用符で囲まれています。

変数の使用方法を更に説明するサンプル

- `<xsl:variable name="xpath" select="'$p1'" />`
  `<xsl:value-of select="altova:evaluate($xpath, //Name[1])" />`
  最初の Name *要素の出力値*

- `<xsl:variable name="xpath" select="'$p1'" />`
  `<xsl:value-of select="altova:evaluate($xpath, '//Name[1]')" />`
  Outputs "//Name[1]"

altova:evaluate() 拡張関数は、XSLT スタイルシート内のXPath 式が動的に評価される必要のある1つ以上の部分を持つシチュエーションで役に立ちます。例えば、ユーザーが並べ替えの必要条件をリクエストする場合、このシチュエーションは属性 UserReq/@sortkey に保管されます。スタイルシートでは、以下の式が使用できます：<xsl:sort select="**altova:evaluate(../UserReq/@sortkey)**" order="ascending"/>。altova:evaluate() 関数は、コンテキストノードの親の UserReq 子要素の sortkey 属性を読み込みます。sortkey 属性の値が Price の場合、Price は、altova:evaluate() 関数により返され、select 属性 <xsl:sort select="**Price**" order="ascending"/> の値になります。この sort 命令が、Order という要素のコンテキスト内で発生する場合、Order 要素は Price の子の値に従い並べ替えられます。また、@sortkey の値が、Date の場合、Order 要素は、Date の子の値に従い並べ替えられます。ですから、Order の並べ替えの条件は、ランタイムでの sortkey 属性から選択されます。これは、以下の式などでは達成することはできません：<xsl:sort select="**../UserReq/@sortkey**" order="ascending"/>。上の場合、並べ替え条件は sortkey 属性自身であり、Price または Date（または、現在の sortkey のコンテンツ）ではありません。

メモ: 静的なコンテキストは、呼び出し環境の名前空間、型、機能、しかし変数を以外を含みます。ベースURI とデフォルトの名前空間は継承されます。

追加サンプル

- 静的な変数：`<xsl:value-of select="$i3, $i2, $i1" />`
  3 つの変数の値を出力します。

- 動的な変数を持つ動的 XPath 式：
  `<xsl:variable name="xpath" select="'$p3, $p2, $p1'" />`
  `<xsl:value-of select="altova:evaluate($xpath, 10, 20, 30)" />`
  "30 20 10" を出力します。

- 動的な変数を持たないXPath 式：
  `<xsl:variable name="xpath" select="'$p3, $p2, $p1'" />`
  `<xsl:value-of select="altova:evaluate($xpath)" />`
  出力エラー：$p3 に対して定義されている変数はありません。

encode-for-rtf [altova:]

altova:encode-for-rtf(input *as xs:string*, **preserveallwhitespace** *as xs:boolean*, **preservenewlines** *as xs:boolean*) を**xs:string** とする **XSLT2** **XSLT3**

RTF のためのコードに入力文字列を変換します。空白と新しい行は、それぞれの引数により指定されるboolean の値に基づき保管されます。

[トップ]

XBRL 関数

Altova XBRL 関数はXBRL をサポートするAltova 製品のエディションのみで使用することができます。

xbrl-footnotes [altova:]

altova:xbrl-footnotes(*node()***)** を**node()*** とする XSLT2 XSLT3
ノードを入力引数として必要とし、入力ノードに参照されるXBRL フットノート ノードを返します。

xbrl-labels [altova:]

altova:xbrl-labels(*xs:QName***, *xs:string***)** を**node()*** とする XSLT2 XSLT3
以下の 2 つの入力引数を必要とします : ノード名とノードを含むタクソノミファイルロケーション。関数は、入力ノードと関連したXBRL ラベルノードを返します。

[トップ]

XPath/ XQuery 関数 :日付と時刻

Altova の日付 時刻拡張関数はXPath とXQuery 式で使用することができ、XML スキーマの異なる日付および時刻データ型で保存されているデータを処理するための追加機能を提供します。このセクションの関数は、Altova の**XPath 3.0** および**XQuery 3.0** エンジンで使用することができます。これらの関数は、XPath/XQuery コンテキストで使用することができます。

関数の名前指定と言語の適用性

Altova 拡張関数はXPath/XQuery 式で使用することができ、XPath、XQuery、およびXSLT 関数の標準ライブラリで使用可能な機能に更なる機能性を与えます。Altova 拡張関数は**Altova 拡張関数名前空間、http://www.altova.com/xslt-extensions** に収められており、**altova:** プレフィックスが、このセクションでは使用されます。製品の今後のバージョンが拡張機能への継続的サポート、または個別の関数の振る舞いは変更する可能性があることにご注意してください。Altova 拡張機能へのサポートに関しては、今後のリリースのドキュメントを参照してください。

XPath 関数 (XSLT 内のXPath 式で使用):	XP1 XP2 XP3
XSLT 関数 (XSLT 内のXPath 式で使用):	XSLT1 XSLT2 XSLT3
XQuery 関数 (XQuery 内のXQuery 式で使用):	XQ1 XQ3

機能によりグループ化

- xs:dateTime に期間を追加して、xs:dateTimeを返す
- xs:date に期間を追加して、xs:date を返す
- xs:time に期間を追加して、return xs:time を返す
- フォーマットと期間の取得

- 現在の日付 時刻を生成する関数からタイムゾーンを削除する
- 日付から整数を週の曜日として返す
- 日付から週数を整数として返す
- 各型の構文コンポーネントから日付、時刻、期間 の型を構築する
- 文字列入力から日付、日付時刻 または時刻 を構築する
- 年齢に関連した関数

▼ アルファベット順にグループ化

altova:add-days-to-date
altova:add-days-to-dateTime
altova:add-hours-to-dateTime
altova:add-hours-to-time
altova:add-minutes-to-dateTime
altova:add-minutes-to-time
altova:add-months-to-date
altova:add-months-to-dateTime
altova:add-seconds-to-dateTime
altova:add-seconds-to-time
altova:add-years-to-date
altova:add-years-to-dateTime
altova:age
altova:age-details
altova:build-date
altova:build-duration
altova:build-time
altova:current-dateTime-no-TZ
altova:current-date-no-TZ
altova:current-time-no-TZ
altova:format-duration
altova:parse-date
altova:parse-dateTime
altova:parse-duration
altova:parse-time
altova:weekday-from-date
altova:weekday-from-dateTime
altova:weeknumber-from-date
altova:weeknumber-from-dateTime

[トップ]

---

xs:dateTime に期間を追加する XP3 XQ3

これらの関数は **xs:dateTime** に期間を追加し、**xs:dateTime** を返します。xs:dateTime 型はCCYY-MM-DDThh:mm:ss.sss のフォーマットです。これはxs:date とxs:time フォーマットの連結で、T により区切られています。タイムゾーンサフィックス+01:00 (for example)は任意です。

▼ **add-years-to-dateTime [altova:]**

altova:add-years-to-dateTime(DateTime *as xs:dateTime*, **Years** を *xs:integer*) as **xs:dateTime** とする XP3 XQ3

日付までの期間を年数で表示します。第 2の引数は第 1の引数として与えられたxs:date に追加される年数です。結果はxs:date 型です。

▫ *サンプル*

- **altova:add-years-to-dateTime**(xs:dateTime("2014-01-15T14:00:00"), 10) returns 2024-01-15T14:00:00
- **altova:add-years-to-dateTime**(xs:dateTime("2014-01-15T14:00:00"), -4)

returns `2010-01-15T14:00:00`

add-months-to-dateTime [altova:]

altova:add-months-to-dateTime(DateTime *as xs:dateTime*, **Months** を *xs:integer*) as **xs:dateTime** とする **XP3 XQ3**

xs:dateTime に月数での期間を追加します (*下のサンプル参照*)。第 2の引数は、第 1の引数として与えられたxs:dateTime に追加される月数です。結果はxs:dateTime 型です。

サンプル

- **altova:add-months-to-dateTime**(xs:dateTime("2014-01-15T14:00:00"), 10) 2014-11-15T14:00:00 を返します。
- **altova:add-months-to-dateTime**(xs:dateTime("2014-01-15T14:00:00"), -2) 2013-11-15T14:00:00 を返します。

add-days-to-dateTime [altova:]

altova:add-days-to-dateTime(DateTime *as xs:dateTime*, **Days** *as xs:integer*) を **xs:dateTime** とする **XP3 XQ3**

xs:dateTime に日数での期間を追加します (*下のサンプル参照*)。第 2の引数は、第 1の引数として与えられたxs:dateTime に追加される日数です。結果はxs:dateTime 型です。

サンプル

- **altova:add-days-to-dateTime**(xs:dateTime("2014-01-15T14:00:00"), 10)は 2014-01-25T14:00:00 を返します。
- **altova:add-days-to-dateTime**(xs:dateTime("2014-01-15T14:00:00"), -8)は 2014-01-25T14:00:00 を返します。

add-hours-to-dateTime [altova:]

altova:add-hours-to-dateTime(DateTime *as xs:dateTime*, **Hours** *as xs:integer*) を**xs:dateTime** とする **XP3 XQ3**

xs:dateTime に時間数での期間を追加します (*下のサンプル参照*)。第 2の引数は、第 1の引数として与えられたxs:dateTime に追加される時間数です。結果はxs:dateTime 型です。

サンプル

- **altova:add-hours-to-dateTime**(xs:dateTime("2014-01-15T13:00:00"), 10) は 2014-01-15T23:00:00 を返します。
- **altova:add-hours-to-dateTime**(xs:dateTime("2014-01-15T13:00:00"), -8) は 2014-01-15T05:00:00 を返します。

add-minutes-to-dateTime [altova:]

altova:add-minutes-to-dateTime(DateTime *as xs:dateTime*, **Minutes** *as xs:integer*) を**xs:dateTime** とする **XP3 XQ3**

xs:dateTime に分数での期間を追加します (*下のサンプル参照*)。第 2の引数は、第 1の引数として与えられたxs:dateTime に追加される分数です。結果はxs:dateTime 型です。

サンプル

- **altova:add-minutes-to-dateTime**(xs:dateTime("2014-01-15T14:10:00"), 45) 2014-01-15T14:55:00 を返します。
- **altova:add-minutes-to-dateTime**(xs:dateTime("2014-01-15T14:10:00"), -5)

2014-01-15T14:05:00 を返します。

add-seconds-to-dateTime [altova:]

altova:add-seconds-to-dateTime(DateTime *as xs:dateTime*, **Seconds** を *xs:integer*) as **xs:dateTime** とする **XP3** **XQ3**

xs:dateTime に秒数での期間を追加します (*下のサンプル参照*)。第 2の引数は、第 1の引数として与えられたxs:dateTime に追加される秒数です。結果はxs:dateTime 型です。

サンプル

- **altova:add-seconds-to-dateTime**(xs:dateTime("2014-01-15T14:00:10"), 20)
  2014-01-15T14:00:30 を返します。
- **altova:add-seconds-to-dateTime**(xs:dateTime("2014-01-15T14:00:10"), -5)
  2014-01-15T14:00:05 を返します。

[トップ]

xs:date に期間を追加する XP3 XQ3

これらの関数は**xs:date** に期間を追加し、**xs:date** を返します。xs:date 型はCCYY-MM-DD フォーマットです。

add-years-to-date [altova:]

altova:add-years-to-date(Date *as xs:date*, **Years** *as xs:integer*) を**xs:date** とする **XP3** **XQ3**

日付までの期間を年数で表示します。第 2の引数は第 1の引数として与えられたxs:date に追加される年数です。結果はxs:date 型です。

サンプル

- **altova:add-years-to-date**(xs:date("2014-01-15"), 10) は2024-01-15 を返します。
- **altova:add-years-to-date**(xs:date("2014-01-15"), -4) は2010-01-15 を返します。

add-months-to-date [altova:]

altova:add-months-to-date(Date *as xs:date*, **Months** *as xs:integer*) を**xs:date** とする **XP3** **XQ3**

日付までの期間を月数で表示します。第 2の引数は第 1の引数として与えられたxs:date に追加される月数です。結果はxs:date 型です。

サンプル

- **altova:add-months-to-date**(xs:date("2014-01-15"), 10) 2014-11-15 を返します。
- **altova:add-months-to-date**(xs:date("2014-01-15"), -2) 2013-11-15 を返します。

add-days-to-date [altova:]

altova:add-days-to-date(Date *as xs:date*, **Days** *as xs:integer***)** を**xs:date** とする **XP3** **XQ3**

日付までの期間を日数で表示します。第 2の引数は第 1の引数として与えられたxs:date に追加される日数です。結果はxs:date 型です。

サンプル

- **altova:add-days-to-date**(xs:date("2014-01-15"), 10)は 2014-01-25 を返します。
- **altova:add-days-to-date**(xs:date("2014-01-15"), -8) は2014-01-07 を返します。

[トップ]

フォーマットと期間の取得 XP3 XQ3

これらの関数は**xs:date** に期間を追加し、**xs:date** を返します。xs:date 型はCCYY-MM-DD フォーマットです。

format-duration [altova:]

altova:format-duration(Duration *as xs:duration*, **Picture** *as xs:string***)** as **xs:string** とする **XP3** **XQ3**

第 1の引数として提出された期間を、第 2の引数として提出された文字列によりをフォーマットします。出力は、文字列によりフォーマットされたテキスト文字列です。

サンプル

- **altova:format-duration**(xs:duration("P2DT2H53M11.7S"), "Days:[D01] Hours:[H01] Minutes:[m01] Seconds:[s01] Fractions:[f0]") は"Days:02 Hours:02 Minutes:53 Seconds:11 Fractions:7" を返します。
- **altova:format-duration**(xs:duration("P3M2DT2H53M11.7S"), "Months:[M01] Days:[D01] Hours:[H01] Minutes:[m01]") は"Months:03 Days:02 Hours:02 Minutes:53" を返します。

parse-duration [altova:]

altova:parse-duration(InputString *as xs:string*, **Picture** *as xs:string***)** を **xs:duration** とする **XP3** **XQ3**

パターン化された文字列を最初の引数として、文字を第 2の引数とします。入力文字列は文字をベースに解析され、xs:duration が返されます。

サンプル

- **altova:parse-duration**("Days:02 Hours:02 Minutes:53 Seconds:11

Fractions:7"), "Days:[D01] Hours:[H01] Minutes:[m01] Seconds:[s01] Fractions:[f0]") は"P2DT2H53M11.7S" を返します。

- **altova:parse-duration**("Months:03 Days:02 Hours:02 Minutes:53 Seconds:11 Fractions:7", "Months:[M01] Days:[D01] Hours:[H01] Minutes:[m01]") は"P3M2DT2H53M" を返します。

[Top]

xs:time に期間を追加する XP3 XQ3

これらの関数はxs:time に期間を追加し、 xs:time を返します。 xs:time 型はhh:mm:ss.sss 構文フォームです。

文字 z は協定世界時 (UTC) を表します。他のタイムゾーンは、UTC との差異を+hh:mm または -hh:mm のフォーマットで表示しています。タイムゾーンの値が無い場合は、UTC ではない未知のタイムゾーンとして見なされます。

add-hours-to-time [altova:]

altova:add-hours-to-time(Time *as xs:time*, **Hours** *as xs:integer*) を**xs:time** とする XP3 XQ3

日付までの期間を時間数で表示します。第 2の引数は第 1の引数として与えられたxs:time に追加される時間数です。結果はxs:time 型です。

サンプル

- **altova:add-hours-to-time**(xs:time("11:00:00"), 10) は21:00:00 を返します。
- **altova:add-hours-to-time**(xs:time("11:00:00"), -7) は04:00:00 を返します。

add-minutes-to-time [altova:]

altova:add-minutes-to-time(Time *as xs:time*, **Minutes** *as xs:integer*) を**xs:time** とする XP3 XQ3

日付までの期間を分数で表示します。第 2の引数は第 1の引数として与えられたxs:date に追加される分数です。結果はxs:date 型です。

サンプル

- **altova:add-minutes-to-time**(xs:time("14:10:00"), 45) 14:55:00 を返します。
- **altova:add-minutes-to-time**(xs:time("14:10:00"), -5) 14:05:00 を返します。

add-seconds-to-time [altova:]

altova:add-seconds-to-time(Time *as xs:time*, **Minutes** *as xs:integer*) を**xs:time** とする XP3 XQ3

時間までの期間を秒数で表示します。第 2の引数は、第 1の引数として与えられたxs:time に追加される秒数です。結果はxs:time 型です。第 2のコンポーネントは0 から59.999 の範囲であることができます。

サンプル

- **altova:add-seconds-to-time**(xs:time("14:00:00"), 20) は14:00:20 を返します。
- **altova:add-seconds-to-time**(xs:time("14:00:00"), 20.895) は14:00:20.895 を返します。

[Top]

現在の日付 /時刻を生成する関数からタイムゾーンを削除する XP3 XQ3

これらの関数は、現在の**xs:dateTime**、**xs:date**、または**xs:time** 値からそれぞれタイムゾーンを削除します。xs:dateTime とxs:dateTimeStamp の差異は、後者のタイムゾーンが必要な場合です。前者の場合は任意です。) xs:dateTimeStamp 値のフォーマットはCCYY-MM-DDThh:mm:ss.sss±hh:mm またはCCYY-MM-DDThh:mm:ss.sssZ です。、日付と時刻がxs:dateTimeStamp としてシステムクロックから読み込まれる場合、current-dateTime-no-TZ() 関数がタイムゾーンを削除するために使用されます。

current-dateTime-no-TZ [altova:]

altova:current-dateTime-no-TZ() を**xs:dateTime** とする XP3 XQ3

この関数は引数を必要としません。current-dateTime() (システムクロックによる現在の時刻)のタイムゾーンの部分を削除し、xs:dateTime の値を返します。

サンプル

現在のdateTime が**2014-01-15T14:00:00+01:00** の場合：

- **altova:current-dateTime-no-TZ**() は2014-01-15T14:00:00 を返します。

current-date-no-TZ [altova:]

altova:current-date-no-TZ() を**xs:date** とする XP3 XQ3

この関数は引数を必要としません。current-date() (システムクロックによる現在の時刻)のタイムゾーンの部分を削除し、xs:date の値を返します。

サンプル

現在のdate が**2014-01-15+01:00** の場合：

- **altova:current-date-no-TZ**() は2014-01-15 を返します。

current-time-no-TZ [altova:]

altova:current-time-no-TZ() を**xs:time** とする XP3 XQ3

この関数は引数を必要としません。current-time() (システムクロックによる現在の時刻)のタイムゾーンの部分を削除し、xs:time の値を返します。

サンプル

現在のtime が**14:00:00+01:00** の場合：

- **altova:current-time-no-TZ**() は14:00:00 を返します。

[Top]

---

xs:dateTime または xs:date として曜日を返す XP3 XQ3

これらの関数は、xs:dateTime またはxs:date から曜日を整数として)を返します。曜日は (米国式フォーマットを使用して)1 から7 と番号づけられています。 日曜=1 と番号付けられます。欧州のフォーマットでは月曜 (=1) として番号付けられます。日曜=1 である米国フォーマットは整数 0 がフォーマットを表示するために使用できる箇所で設定することができます。

weekday-from-dateTime [altova:]

altova:weekday-from-dateTime(DateTime *as xs:dateTime*) を**xs:integer** とする XP3 XQ3

時刻付きの日付を単一の引数として、この日付の曜日を正数として返します。曜日は 日曜=1 から開始して番号を

付けます。(月曜=1 とする) ヨーロッパのフォーマットが必要な場合、この関数の他の署名を使用します (*下の次の署名を参照*)。

サンプル

- `altova:weekday-from-dateTime(xs:dateTime("2014-02-03T09:00:00"))` は月曜を表示する`2` を返します。

`altova:weekday-from-dateTime(DateTime as xs:dateTime, Format as xs:integer)` as `xs:integer` とする XP3 XQ3

時間月の日付を最初の引数として、この日付の曜日を正数として返します。曜日は月曜=1 から開始して番号を付けます。第 2の引数 (整数) が`0` の場合、日曜=1 から開始して、曜日は1から7 と番号を付けます。第 2の引数が`0` 以外の整数の場合、月曜=1です。第 2の引数がない場合、関数はこの関数の他の署名を持つと読み込まれます (*前の次の署名を参照*)。

サンプル

- `altova:weekday-from-dateTime(xs:dateTime("2014-02-03T09:00:00"), 1)` は月曜を表示する`1` を返します。
- `altova:weekday-from-dateTime(xs:dateTime("2014-02-03T09:00:00"), 4)` は月曜を表示する`1` を返します。
- `altova:weekday-from-dateTime(xs:dateTime("2014-02-03T09:00:00"), 0)` は月曜を表示する`2` を返します。

`weekday-from-date [altova:]`

`altova:weekday-from-date(Date as xs:date)` を `xs:integer` とする XP3 XQ3

日付を単一の引数として、この日付の曜日を正数として返します。曜日は日曜=1 から開始して番号を付けます。(月曜=1 とする) ヨーロッパのフォーマットが必要な場合、この関数の他の署名を使用します (*下の次の署名を参照*)。

サンプル

- `altova:weekday-from-date(xs:date("2014-02-03+01:00"))` は月曜を表示する`2` を返します。

`altova:weekday-from-date(Date as xs:date, Format as xs:integer)` を `xs:integer` とする XP3 XQ3

日付を最初の引数とし、この日付の曜日を正数として返します。曜日は月曜=1 から開始して番号を付けます。第 2 (フォーマット)引数が`0`,の場合、日曜=1 から開始し、曜日は`1` から7で番号付けられます。第 2 引数が整数で`0` 以外の場合、月曜=1です。第 2 の引数がない場合、関数はこの関数の他の署名を持つと読み込まれます (*前の署名を参照*)。

サンプル

- `altova:weekday-from-date(xs:date("2014-02-03"), 1)` は月曜を示す`1` を返します。
- `altova:weekday-from-date(xs:date("2014-02-03"), 4)` は月曜を示す`1` を返します。
- `altova:weekday-from-date(xs:date("2014-02-03"), 0)` は月曜を示す`2` を返します。

[Top]

`xs:dateTime` または `xs:date` から週数を返す XP2 XQ1 XP3 XQ3

これらの関数は週数 (整数として) をxs:dateTime からxs:date.から返します。週の番号付けは、米国、欧州、イスラムのカレンダーフォーマットで使用することができます。週の始まりが異なるため、週数の番号付けは、カレンダーのフォーマットにより異なります。(米国フォーマットでは、日曜、欧州フォーマットでは月曜、イスラムフォーマットでは土曜が週の開始日です。)

weeknumber-from-date [altova:]

altova:weeknumber-from-date(Date *as xs:date*, **Calendar** *as xs:integer*) を **xs:integer** とする XP2 XQ1 XP3 XQ3
正数として提出された**Date** 引数の週数を返します。第 2の引数 (**カレンダー**) は続くカレンダーシステムを指定します。
サポートされる **カレンダー** の値は次のとおりです :

- **0 = US 米国のカレンダー** (週の始まりは日曜日)
- **1 = ISO 標準、 欧州のカレンダー** (週の始まりは月曜日)
- **2 = イスラムのカレンダー** (週の始まりは土曜日)

デフォルトは**0** です。

サンプル

- **altova:weeknumber-from-date**(xs:date("2014-03-23"), 0) は13 を返します。
- **altova:weeknumber-from-date**(xs:date("2014-03-23"), 1) は13 を返します。
- **altova:weeknumber-from-date**(xs:date("2014-03-23"), 2) は13 を返します。
- **altova:weeknumber-from-date**(xs:date("2014-03-23")) は13 を返します。

上のサンプル (2014-03-23) のdate の曜日は日曜日です。米国およびイスラムのカレンダーは欧州カレンダーのこの日付よりも一週間先です。

weeknumber-from-dateTime [altova:]

altova:weeknumber-from-dateTime(DateTime *as xs:dateTime*, **Calendar** *as xs:integer*) を**xs:integer** とする XP2 XQ1 XP3 XQ3
正数として提出された**DateTime** 引数の週数を返します。第 2の引数 (**カレンダー**) は続くカレンダーシステムを指定します。
サポートされる **カレンダー** の値は次のとおりです :

- **0 = US 米国のカレンダー** (週の始まりは日曜日)
- **1 = ISO 標準、 欧州のカレンダー** (週の始まりは月曜日)
- **2 = イスラムのカレンダー** (週の始まりは土曜日)

Default is **0**.

サンプル

- **altova:weeknumber-from-dateTime**(xs:dateTime("2014-03-23T00:00:00"), 0) は13 を返します。
- **altova:weeknumber-from-dateTime**(xs:dateTime("2014-03-23T00:00:00"), 1) は13 を返します。
- **altova:weeknumber-from-dateTime**(xs:dateTime("2014-03-23T00:00:00"), 2) は13 を返します。
- **altova:weeknumber-from-dateTime**(xs:dateTime("2014-03-23T00:00:00")) は13 を返します。

上のサンプル (2014-03-23T00:00:00) のdateTime の曜日は日曜日です。米国およびイスラムのカレンダーは欧州カレンダーのこの日付よりも一週間先です。

[トップ]

各型の構文コンポーネントから日付、時刻、期間の型を構築する XP3 XQ3

関数は、xs:date, xs:time または xs:duration の構文コンポーネントを入力引数とし、引数を結合させて対応するデータ型を構築します。

build-date [altova:]

altova:build-date(Year *as xs:integer*, **Month** *as xs:integer*, **Date** *as xs:integer*) を **xs:date** とする XP3 XQ3

第 1、第 2、第3の引数は、それぞれ、年、月、日を表します。xs:date 型の値を構築するため結合されます。整数の値は、特定の日付の一部の適正な範囲内である必要があります。例えば第 2の引数 (月の部分)は12 以上であってはなりません。

サンプル

- **altova:build-date**(2014, 2, 03) は2014-02-03 を返します。

build-time [altova:]

altova:build-time(Hours *as xs:integer*, **Minutes** *as xs:integer*, **Seconds** *as xs:integer*) を **xs:time** とする XP3 XQ3

第 1、第 2、第3引数はそれぞれ時間数 (0 から23)、分数 (0 から59)、および秒数 (0 から59) の値です。これらの値は、xs:time 型の値を作成するために結合されます。整数の値は、それぞれの部分の正しい範囲内である必要があります。例えば、秒 (分)引数は、59 以上であってはなりません。値にタイムゾーンを追加するには、この関数の他の署名を使用してください (*次の署名を参照*)。

サンプル

- **altova:build-time**(23, 4, 57) は23:04:57 を返します。

altova:build-time(Hours *as xs:integer*, **Minutes** *as xs:integer*, **Seconds** *as xs:integer*, **TimeZone** *as xs:string*) を **xs:time** とする XP3 XQ3

第 1、第 2、第 3引数はそれぞれ時間数 (0 から23)、分数 (0 から59)、および秒数 (0 から59) の値です。第四の引数は、値の一部としてタイムゾーンを与えます。 4つの引数が結合され、xs:time 型の値を構築します。例えば、第 2の引数 (分数) は59 以上であってはなりません。

サンプル

- **altova:build-time**(23, 4, 57, '+1') は23:04:57+01:00 を返します。

build-duration [altova:]

altova:build-duration(Years *as xs:integer*, **Months** *as xs:integer*) を **xs:yearMonthDuration** とする XP3 XQ3

xs:yearMonthDuration 型の値を構築するためには 2つの引数が必要です。最初の引数は期間の年数 値の部分を与え、第 2の引数は、月数 値の部分を与えます。第 2の引数 (月数)が同じまたはより大きくなると、整数は12 により分割されます。 商は、年数 の部分を表す 最初の引数に加算され、除算の残りは の部分は月数の部分を表すために使用されます。期間のxs:dayTimeDuration 型を作成するには、次の署名を参照してください。

サンプル

- **altova:build-duration**(2, 10) はP2Y10M を返します。
- **altova:build-duration**(14, 27) はP16Y3M を返します。
- **altova:build-duration**(2, 24) はP4Y を返します。

altova:build-duration(Days *as xs:integer*, **Hours** *as xs:integer*, **Minutes** *as xs:integer*, **Seconds** *as xs:integer*) を**xs:dayTimeDuration** とする XP3 XQ3
xs:dayTimeDuration 型の値を構築するためには 4 つの引数が必要です。最初の引数は期間の日数 値の部分で、第 2、第 3、第 4引数は、それぞれ期間の時間数、分数、秒数 値のです。これらの 3 つの時間 引数は、次の高い単位の値に加算され、結果は期間の全体の値を計算するために使用されます。例えば、 72秒は、1M+12S (1 分 と12秒)に変換され、この値が期間全体の値を計算するために使用されます。
xs:yearMonthDuration 型の期間を構築するためには、次の署名を参照してください。

サンプル

- **altova:build-duration**(2, 10, 3, 56) はP2DT10H3M56S を返します。
- **altova:build-duration**(1, 0, 100, 0) はP1DT1H40M を返します。
- **altova:build-duration**(1, 0, 0, 3600) はP1DT1H を返します。

[トップ]

文字列入力から日付、日付時刻 または 時刻 を構築する XP2 XQ1 XP3 XQ3

関数は、文字列を引数として、xs:date、xs:dateTime または xs:time データ型を構築します。文字列は、データ型のコンポーネントを提出されたパターン引数をベースにして分析されます。

parse-date [altova:]

altova:parse-date(Date *as xs:string*, **DatePattern** *as xs:string*) を**xs:date** とする XP2 XQ1 XP3 XQ3
Date 入力文字列を**xs:date** の値として返します。第二の引数である**DatePattern** は、入力文字列のパターン (コンポーネントのシーケンス)を指定します。**DatePattern** は、下にリストされるコンポーネント指定子および任意の文字であるコンポーネント セパレータと共に説明されています。下のサンプルを参照してください。

D 日付
M 月
Y 年

DatePattern のパターンは、**Date** のパターンに一致する必要があります。出力は**xs:date** 型であるため、出力は常に **YYYY-MM-DD** 構文フォーマットになります。

サンプル

- **altova:parse-date**(xs:string("06-03-2014"), "[D]-[M]-[Y]") は2014-03-06 を返します。
- **altova:parse-date**(xs:string("06-03-2014"), "[M]-[D]-[Y]") は2014-03-06 を返します。
- **altova:parse-date**("06/03/2014", "[M]/[D]/[Y]") は2014-03-06 を返します。
- **altova:parse-date**("06 03 2014", "[M] [D] [Y]") は2014-03-06 を返します。
- **altova:parse-date**("6 3 2014", "[M] [D] [Y]") は2014-03-06 を返します。

parse-dateTime [altova:]

altova:parse-dateTime(DateTime *as xs:string*, **DateTimePattern** *as xs:string*) を**xs:dateTime** とする XP2 XQ1 XP3 XQ3
DateTime 入力文字列を**xs:dateTime** の値として返します。第二の引数である**DateTimePattern** は、入力文字列のパターン (コンポーネントのシーケンス)を指定します。**DateTimePattern** は、下にリストされるコンポーネント指定子および任意の文字であるコンポーネント セパレータと共に説明されています。下のサンプルを参照してください。

D	日
M	月
Y	年
H	時間
m	分
s	秒

DateTimePattern のパターンは、**DateTime** のパターンに一致する必要があります。出力は**xs:dateTime** 型であるため、出力は常に**YYYY-MM-DDTHH:mm:ss** 構文フォーマットになります。

サンプル

- **altova:parse-dateTime**(xs:string("06-03-2014 13:56:24"), "[D]-[M]-[Y] [H]:[m]:[s]") は2014-03-06T13:56:24 を返します。
- **altova:parse-dateTime**("time=13:56:24; date=06-03-2014", "time=[H]:[m]:[s]; date=[D]-[M]-[Y]") は2014-03-06T13:56:24 を返します。

parse-time [altova:]

altova:parse-time(Time *as xs:string*, **TimePattern** *as xs:string***)** を**xs:time** とする **XP2 XQ1 XP3 XQ3**

Time 入力文字列を**xs:time** の値として返します。第二の引数である**TimePattern** は、入力文字列のパターン (コンポーネントのシーケンス) を指定します。**TimePattern** は、下にリストされるコンポーネント指定子および任意の文字であるコンポーネントセパレータと共に説明されています。下のサンプルを参照してください。

H	時間
m	分
s	秒

TimePattern のパターンは、**Time** のパターンに一致する必要があります。出力は**xs:time** 型であるため、出力は常に**YYYY-HH:mm:ss** 構文フォーマットになります。

サンプル

- **altova:parse-time**(xs:string("13:56:24"), "[H]:[m]:[s]") は13:56:24 を返します。
- **altova:parse-time**("13-56-24", "[H]-[m]") は13:56:00 を返します。
- **altova:parse-time**("time=13h56m24s", "time=[H]h[m]m[s]s") は13:56:24 を返します。
- **altova:parse-time**("time=24s56m13h", "time=[s]s[m]m[H]h") は13:56:24 を返します。

[トップ]

年齢に関連した関数 XP3 XQ3

関数は計算された年齢 (i) 入力引数の日付と現在の日付の期間 (i) 2 つの入力引数の日付の期間 を返します。**altova:age** 関数は、年齢を年数で返します、**altova:age-details** は年齢を年数、月数、日数からなる3つの整数のシーケンスで返します。

age [altova:]

altova:age(StartDate *as xs:date*) を**xs:integer** とする XP3 XQ3
引数として提出された開始日からシステムクロックから取得された現在の日付 までの日数を数えて、あるオブジェクトの年齢の年数である正数を返します。入力引数が 1 年よりも大きい場合、または一年の場合、返される値は負の数です。

サンプル

If the current date is **2014-01-15**:

- **altova:age**(xs:date("2013-01-15")) は1 を返します。
- **altova:age**(xs:date("2013-01-16")) は0 を返します。
- **altova:age**(xs:date("2015-01-15")) は-1 を返します。
- **altova:age**(xs:date("2015-01-14")) は0 を返します。

altova:age(StartDate *as xs:date*, **EndDate** *as xs:date*) を**xs:integer** とする XP3 XQ3
最初の引数として提出された開始日から第 2 の引数の終了日 までの日数を数えて、あるオブジェクトの年齢の年数である正数を返します。最初の引数が 1 年または第 2 の引数よりも後の日付の場合、返される値は負の数です。

サンプル

If the current date is **2014-01-15**:

- **altova:age**(xs:date("2000-01-15"), xs:date("2010-01-15")) は10 を返します。
- **altova:age**(xs:date("2000-01-15"), current-date()) は、現在の日付が2014-01-15 の場合 14 を返します。
- **altova:age**(xs:date("2014-01-15"), xs:date("2010-01-15")) は-4 を返します。

age-details [altova:]

altova:age-details(InputDate *as xs:date*) を (**xs:integer**)* とする XP3 XQ3
引数とシステムクロックから取得された現在の日付として提出された日付の間の年数、月数、日数の 3 つの整数を返します。返されたyears+months+days の合計は、2 つの日付 (入力の日付と現在の日付) の時間差です。 入力の日付は現在の日付よりも早いまたは遅い値をもつことができますが、入力の日付が早いか遅いかは返される値で示されていません。戻される値は常に正の数です。

サンプル

現在の日付が**2014-01-15** の場合 :

- **altova:age-details**(xs:date("2014-01-16")) は(0 0 1)を返します。
- **altova:age-details**(xs:date("2014-01-14")) は(0 0 1) を返します。
- **altova:age-details**(xs:date("2013-01-16")) は(1 0 1) を返します。
- **altova:age-details**(current-date()) は(0 0 0)を返します。

altova:age-details(Date-1 *as xs:date*, **Date-2** *as xs:date*) を (**xs:integer**)* とする XP3 XQ3
二つの引数間の年数、月数、日数の 3 つの整数を返します。 返されたyears+months+days の合計は、2 つの入力の日付 の時間差です。最初の引数として提出される二つの日付はどちらが速くても、また遅くてもかまいません。返される値は入力の日付が現在の日付より早いかまた遅いかを示しません。戻される値は常に正の数です。

サンプル

- **altova:age-details**(xs:date("2014-01-16"), xs:date("2014-01-15")) は(0 0 1) を返します。
- **altova:age-details**(xs:date("2014-01-15"), xs:date("2014-01-16")) は(0 0 1)を返します。

[Top]

XPath/ XQuery 関数 :位置情報

以下の位置情報 XPath/XQuery 拡張関数は、MapForce の現在のバージョンによりサポートされています。また、次で使用することができます : (i) XSLT コンテキスト内の XPath 式、または (i) XQuery ドキュメント内の XQuery 式。

関数の名前指定と言語の適用性

Altova 拡張関数は XPath/XQuery 式で使用することができ、XPath、XQuery、および XSLT 関数の標準ライブラリで使用可能な機能に更なる機能性を与えます。Altova 拡張関数は **Altova 拡張関数名前空間、`http://www.altova.com/xslt-extensions`** に収められており、**`altova:`** プレフィックスが、このセクションでは使用されます。製品の今後のバージョンが拡張機能への継続的サポート、または個別の関数の振る舞いは変更する可能性があることにご注意してください。Altova 拡張機能へのサポートに関しては、今後のリリースのドキュメントを参照してください。

XPath 関数 (XSLT 内の XPath 式で使用):	**`XP1` `XP2` `XP3`**
XSLT 関数 (XSLT 内の XPath 式で使用):	**`XSLT1` `XSLT2` `XSLT3`**
XQuery 関数 (XQuery 内の XQuery 式で使用):	**`XQ1` `XQ3`**

`parse-geolocation [altova:]`

`altova:parse-geolocation(GeolocationInputString` *`as xs:string`***`)`** を **`xs:decimal+`** とする **`XP3` `XQ3`**

`GeolocationInputString` 引数を解析して、位置情報の緯度と経度 (この通りの順番) を 2 つの `xs:decimal` アイテムのシーケンスとして返します。位置情報入力文字列が提供されることのできるフォーマットは以下のリストの通りです。

メモ: `image-exif-data` 関数と Exif メタデータの `@Geolocation` 属性を位置情報入力文字列を提供する際に使用することができます。

サンプル

- **`altova:parse-geolocation`**`("33.33  -22.22")` は 2 つの `xs:decimals (33.33, 22.22)` のシーケンスを返します。
- **`altova:parse-geolocation`**`("48°51'29.6""N  24°17'40.2""")` は 2 つの `xs:decimals (48.8582222222222, 24.2945)` のシーケンスを返します。
- **`altova:parse-geolocation`**`('48°51''29.6"N  24°17''40.2"')` は 2 つの `xs:decimals (48.8582222222222, 24.2945)` のシーケンスを返します。
- **`altova:parse-geolocation`**`( image-exif-data(//MyImages/Image20141130.01)/@Geolocation )` は 2 つの `xs:decimals` のシーケンスを返します。

位置情報入力文字列フォーマット :

位置情報入力文字列は空白で区別された緯度と経度 (この通りの順番) を含む必要があります。緯度と経度は以下のフォーマットをとることができます。組み合わせることも可能です。緯度が 1 つのフォーマットで、経度が他のフォーマットをとることができます。緯度の値の範囲は `+90` から `-90` (北 から南) です。経度の値の範囲は `+180` から `180` (東 から西) です。

メモ: 単一およびダブル引用符が入力文字列引数を区切るために使用されていると、使用されている単一およびダブル引用が、それぞれ、分の値と秒の値、不一致をもたらします。この様な場合、分の値と秒の値を表すための使用されている引用符は、ダブルにしてエスケープされる必要があります。このセクションのサンプルでは、入力文字列を区別するために使用されている引用符は黄色い (`''`) でハイライトされており、エスケープした単位インジケーターは青い (`""`) でハイライトされています。

- 度、分、10進の秒、方角のサフィックス付き (N/S, W/E)
  D°M'S.SS"N/S D°M'S.SS"W/E
  サンプル: **33°55'11.11"N 22°44'66.66"W**

- 度、分、10進の秒、プレフィックスサイン付き (+/-); (**N/W**) のためのプラスサインは任意です。
  +/-D°M'S.SS" +/-D°M'S.SS"
  サンプル: **33°55'11.11" -22°44'66.66"**

- 度、分、10進の分、方角のサフィックス付き (N/S, W/E)
  D°M.MM'N/S D°M.MM'W/E
  サンプル: **33°55.55'N 22°44.44'W**

- 度、分、10進の分、プレフィックスサイン付き (+/-); (**N/W**) のためのプラスサインは任意です。
  +/-D°M.MM' +/-D°M.MM'
  サンプル: **+33°55.55' -22°44.44'**

- 10 進の度、方角のサフィックス付き (N/S, W/E)
  D.DDN/S D.DDW/E
  サンプル: **33.33N 22.22W**

- 10 進の度、プレフィックスサイン付き (+/-); (**N/W**) のためのプラスサインは任意です。
  +/-D.DD +/-D.DD
  サンプル: **33.33 -22.22**

フォーマットの組み合わせのサンプル:
33.33N -22°44'66.66"
33.33 22°44'66.66"W
33.33 22.c

- *Altova Exif 属性 :位置情報*

  Altova XPath/XQuery エンジンはカスタム属性 **Geolocation** を標準 Exif メタデータタグから生成します。**Geolocation** は、4 つの Exif タグの連結です :単位の追加された (下のテーブル参照) GPSLatitude、GPSLatitudeRef、GPSLongitude、GPSLongitudeRef。

GPSLatitude	GPSLatitudeRef	GPSLongitude	GPSLongitudeRef	Geolocation
33 51 21.91	S	151 13 11.73	E	33° 51'21.91"S 151° 13'11.73"E

- **geolocation-distance-km [altova:]**

  altova:geolocation-distance-km(GeolocationInputString-1 *as xs:string*, **GeolocationInputString-2** *as xs:string*) を **xs:decimal** とする **XP3 XQ3**
  2 つの位置情報の間の距離をキロメートルで計算します。位置情報入力文字列を提供することのできるフォーマットは下にリストされています。緯度の値の範囲は +90 から -90 (北 から南) です。経度の値の範囲は +180 から -180 (東 から西) です。

  メモ: image-exif-data 関数と Exif メタデータの @Geolocation 属性を位置情報入力文字列を提供する際に使用することができます。

サンプル

- **altova:geolocation-distance-km**("33.33 -22.22", "48°51'29.6""N 24° 17'40.2""") は xs:decimal 4183.08132372392 を返します。

位置情報入力文字列フォーマット:

位置情報入力文字列は空白で区別された緯度と経度 (この通りの順番) を含む必要があります。緯度と経度は以下のフォーマットをとることができます。組み合わせることも可能です。緯度が1つのフォーマットで、経度が他のフォーマットをとることができます。緯度の値の範囲は +90 から -90 (北 から南) です。経度の値の範囲は +180 から 180 (東 から西) です。

メモ: 単一およびダブル引用符が入力文字列引数を区切るために使用されていると、使用されている単一およびダブル引用が、それぞれ、分の値と秒の値、不一致をもたらします。この様な場合、分の値と秒の値を表すための使用されている引用符は、ダブルにしてエスケープされる必要があります。このセクションのサンプルでは、入力文字列を区別するために使用されている引用符は黄色い(") でハイライトされており、エスケープした単位インジケーターは青い("") でハイライトされています。

- 度、分、10進の秒、方角のサフィックス付き (N/S, W/E)
  D°M'S.SS"N/S D°M'S.SS"W/E
  サンプル: **33°55'11.11"N 22°44'66.66"W**

- 度、分、10進の秒、プレフィックスサイン付き (+/-); (**N/W**) のためのプラスサインは任意です。
  +/-D°M'S.SS" +/-D°M'S.SS"
  サンプル: **33°55'11.11" -22°44'66.66"**

- 度、分、10進の分、方角のサフィックス付き (N/S, W/E)
  D°M.MM'N/S D°M.MM'W/E
  サンプル: **33°55.55'N 22°44.44'W**

- 度、分、10進の分、プレフィックスサイン付き (+/-); (**N/W**) のためのプラスサインは任意です。
  +/-D°M.MM' +/-D°M.MM'
  サンプル: **+33°55.55' -22°44.44'**

- 10 進の度、方角のサフィックス付き (N/S, W/E)
  D.DDN/S D.DDW/E
  サンプル: **33.33N 22.22W**

- 10 進の度、プレフィックスサイン付き (+/-); (**N/W**) のためのプラスサインは任意です。
  +/-D.DD +/-D.DD
  サンプル: **33.33 -22.22**

フォーマットの組み合わせのサンプル:
33.33N -22°44'66.66"
33.33 22°44'66.66"W
33.33 22.c

Altova Exif 属性 :位置情報

Altova XPath/XQuery エンジンはカスタム属性 **Geolocation** を標準 Exif メタデータタグから生成します。**Geolocation** は、4つの Exif タグの連結です :単位の追加された (下のテーブル参照) GPSLatitude、GPSLatitudeRef、GPSLongitude、GPSLongitudeRef。

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 *as xs:string*, **GeolocationInputString-2** *as xs:string*) を**xs:decimal** とする **XP3 XQ3**

2 つの位置情報の間の距離をマイルで計算します。位置情報入力文字列を提供することのできるフォーマットは下にリストされています。緯度の値の範囲は +90 から -90 (北 から南) です。経度の値の範囲は +180 から 180 (東 から西) です。

メモ: image-exif-data 関数とExif メタデータの @Geolocation 属性を位置情報入力文字列を提供する際に使用することができます。.

サンプル

- **altova:geolocation-distance-mi**("33.33 -22.22", "48°51'29.6""N 24° 17'40.2""") は xs:decimal 2599.40652340653 を返します。

位置情報入力文字列フォーマット:

位置情報入力文字列は空白で区別された緯度と経度 (この通りの順番) を含む必要があります。緯度と経度は以下のフォーマットをとることができます。組み合わせることも可能です。緯度が1つのフォーマットで、経度が他のフォーマットをとることができます。緯度の値の範囲は +90 から -90 (北 から南) です。経度の値の範囲は +180 から 180 (東 から西) です。

メモ: 単一およびダブル引用符が入力文字列引数を区切るために使用されていると、使用されている単一およびダブル引用符が、それぞれ、分の値と秒の値、不一致をもたらします。この様な場合、分の値と秒の値を表すための使用されている引用符は、ダブルにしてエスケープされる必要があります。このセクションのサンプルでは、入力文字列を区別するために使用されている引用符は黄色い(") でハイライトされており、エスケープした単位インジケーターは青い("") でハイライトされています。

- 度、分、 10進の秒、方角のサフィックス付き (N/S, W/E)
  D°M'S.SS"N/S D°M'S.SS"W/E
  サンプル: **33°55'11.11"N 22°44'66.66"W**

- 度、分、 10進の秒、プレフィックスサイン付き (+/-); (**N/W**) のためのプラスサインは任意です。
  +/-D°M'S.SS" +/-D°M'S.SS"
  サンプル: **33°55'11.11" -22°44'66.66"**

- 度、分、 10進の分、方角のサフィックス付き (N/S, W/E)
  D°M.MM'N/S D°M.MM'W/E
  サンプル: **33°55.55'N 22°44.44'W**

- 度、分、 10進の分、プレフィックスサイン付き (+/-); (**N/W**) のためのプラスサインは任意です。
  +/-D°M.MM' +/-D°M.MM'
  サンプル: **+33°55.55' -22°44.44'**

- 10 進の度、方角のサフィックス付き (N/S, W/E)
  D.DDN/S D.DDW/E
  サンプル: **33.33N 22.22W**

- 10 進の度、プレフィックスサイン付き (+/-); (N/W) のためのプラスサインは任意です。
  +/-D.DD +/-D.DD
  サンプル: 33.33 -22.22

フォーマットの組み合わせのサンプル:
33.33N -22°44'66.66"
33.33 22°44'66.66"W
33.33 22.c

Altova Exif 属性 :位置情報

Altova XPath/XQuery エンジンはカスタム属性 Geolocation を標準 Exif メタデータタグから生成します。Geolocation は、4 つの Exif タグの連結です :単位の追加された (下のテーブル参照) GPSLatitude、GPSLatitudeRef、GPSLongitude、GPSLongitudeRef。

GPSLatitude	GPSLatitudeRef	GPSLongitude	GPSLongitudeRef	Geolocation
33 51 21.91	S	151 13 11.73	E	33° 51'21.91"S 151° 13'11.73"E

geolocation-within-polygon [altova:]

altova:geolocation-within-polygon(Geolocation as xs:string, ((PolygonPoint as xs:string)+)) を xs:boolean とする XP3 XQ3

PolygonPoint 引数により説明されている Geolocation (最初の引数) が多角形のエリア内に存在するかを決定します。もし、PolygonPoint 引数が最初と最後のポイントが同じ場合に作成される閉じられたフィギュアを作成しない場合、フィギュアを閉じるために、最初のポイントが明示的に最後のポイントとして追加されます。全ての引数 (Geolocation および PolygonPoint+) は、(下にリストされるフォーマットの) 位置情報入力文字列により提供されます。Geolocation 引数が多角形エリア内にある場合、関数は true(); を返します。その他の場合は false() を返します。緯度の値の範囲は +90 から -90 (北 から南) です。経度の値の範囲は +180 から -180 (東 から西) です。

メモ: image-exif-data 関数と Exif メタデータの @Geolocation 属性を位置情報入力文字列を提供する際に使用することができます。

サンプル

- altova:geolocation-within-polygon("33 -22", ("58 -32", "-78 -55", "48 24", "58 -32")) は true() を返します。
- altova:geolocation-within-polygon("33 -22", ("58 -32", "-78 -55", "48 24")) は true() を返します。
- altova:geolocation-within-polygon("33 -22", ("58 -32", "-78 -55", "48°51'29.6""N 24°17'40.2""")) は true() を返します。

位置情報入力文字列フォーマット:

位置情報入力文字列は空白で区別された緯度と経度 (この通りの順番) を含む必要があります。緯度と経度は以下のフォーマットをとることができます。組み合わせることも可能です。緯度が 1 つのフォーマットで、経度が他のフォーマットをとることができます。緯度の値の範囲は +90 から -90 (北 から南) です。経度の値の範囲は +180 から 180 (東 から西) です。

メモ: 単一およびダブル引用符が入力文字列引数を区切るために使用されていると、使用されている単一およびダブル引用が、それぞれ、分の値と秒の値、不一致をもたらします。この様な場合、分の値と秒の値を表すための使用されている引用符は、ダブルにしてエスケープされる必要があります。このセクションのサンプルでは、入力文字列を区別するために使用されている引用符は黄色い(**''**) でハイライトされており、エスケープした単位インジケーターは青い(**""**) でハイライトされています。

- 度、分、10進の秒、方角のサフィックス付き (N/S, W/E)
  D°M'S.SS"N/S D°M'S.SS"W/E
  サンプル: **33°55'11.11"N 22°44'66.66"W**

- 度、分、10進の秒、プレフィックスサイン付き (+/-); (**N/W**) のためのプラスサインは任意です。
  +/-D°M'S.SS" +/-D°M'S.SS"
  サンプル: **33°55'11.11" -22°44'66.66"**

- 度、分、10進の分、方角のサフィックス付き (N/S, W/E)
  D°M.MM'N/S D°M.MM'W/E
  サンプル: **33°55.55'N 22°44.44'W**

- 度、分、10進の分、プレフィックスサイン付き (+/-); (**N/W**) のためのプラスサインは任意です。
  +/-D°M.MM' +/-D°M.MM'
  サンプル: **+33°55.55' -22°44.44'**

- 10 進の度、方角のサフィックス付き (N/S, W/E)
  D.DDN/S D.DDW/E
  サンプル: **33.33N 22.22W**

- 10 進の度、プレフィックスサイン付き (+/-); (**N/W**) のためのプラスサインは任意です。
  +/-D.DD +/-D.DD
  サンプル: **33.33 -22.22**

フォーマットの組み合わせのサンプル:
33.33N -22°44'66.66"
33.33 22°44'66.66"W
33.33 22.c

Altova Exif 属性 :位置情報

Altova XPath/XQuery エンジンはカスタム属性 **Geolocation** を標準 Exif メタデータタグから生成します。**Geolocation** は、4 つの Exif タグの連結です :単位の追加された (*下のテーブル参照*) GPSLatitude、GPSLatitudeRef、GPSLongitude、GPSLongitudeRef。

GPSLatitude	GPSLatitudeRef	GPSLongitude	GPSLongitudeRef	Geolocation
33 51 21.91	S	151 13 11.73	E	33° 51'21.91"S 151° 13'11.73"E

geolocation-within-rectangle [altova:]

altova:geolocation-within-rectangle(Geolocation *as xs:string*, **RectCorner-1** *as xs:string*, **RectCorner-2** *as xs:string*) を **xs:boolean** とする **XP3 XQ3**

長方形の対格の角を指定する第 2及び第 3引数 (**RectCorner-1**、**RectCorner-2**)により説明されている**Geolocation** (最初の引数)が長方形のエリア内に存在するかを決定します。全ての引数 (Geolocation、**RectCorner-1** および**RectCorner-2**) は、 (*下にリストされるフォーマットの*) 位置情報入力文字列により提供されます。Geolocation 引数が長方形エリア内にある場合、関数はtrue(); を返します。その他の場合はfalse()を返します。緯度の値の範囲は+90 から-90 (北 から南)です。経度の値の範囲は+180 から 180 (東 から西)です。

メモ:image-exif-data 関数とExif メタデータの@Geolocation 属性を位置情報入力文字列を提供する際に使用することができます。

- *サンプル*
  - **altova:geolocation-within-rectangle(**"33 -22", "58 -32", "-48 24") は true() を返します。
  - **altova:geolocation-within-rectangle(**"33 -22", "58 -32", "48 24") は false() を返します。
  - **altova:geolocation-within-rectangle(**"33 -22", "58 -32", **"**48°51'29.6**""**S 24°17'40.2**""**") はtrue() を返します。

- *位置情報入力文字列フォーマット:*

  位置情報入力文字列は空白で区別された緯度と経度 (この通りの順番)を含む必要があります。緯度と経度は以下のフォーマットをとることができます。組み合わせることも可能です。緯度が1つのフォーマットで、経度が他のフォーマットをとることができます。緯度の値の範囲は+90 から-90 (北 から南)です。経度の値の範囲は+180 から 180 (東 から西)です。

  メモ:単一およびダブル引用符が入力文字列引数を区切るために使用されていると、使用されている単一およびダブル引用が、それぞれ、分の値と秒の値、不一致をもたらします。この様な場合、分の値と秒の値を表すための使用されている引用符は、ダブルにしてエスケープされる必要があります。このセクションのサンプルでは、入力文字列を区別するために使用されている引用符は黄色い(**"**) でハイライトされており、エスケープした単位インジケーターは青い(**""**) でハイライトされています。

  - 度、分、 10進の秒、方角のサフィックス付き (N/S, W/E)
    D°M'S.SS"N/S D°M'S.SS"W/E
    サンプル: **33°55'11.11"N 22°44'66.66"W**

  - 度、分、 10進の秒、プレフィックスサイン付き (+/-); (**N/W**) のためのプラスサインは任意です。
    +/-D°M'S.SS" +/-D°M'S.SS"
    サンプル: **33°55'11.11" -22°44'66.66"**

  - 度、分、 10進の分、方角のサフィックス付き (N/S, W/E)
    D°M.MM'N/S D°M.MM'W/E
    サンプル: **33°55.55'N 22°44.44'W**

  - 度、分、 10進の分、プレフィックスサイン付き (+/-); (**N/W**) のためのプラスサインは任意です。
    +/-D°M.MM' +/-D°M.MM'
    サンプル: **+33°55.55' -22°44.44'**

  - 10 進の度、方角のサフィックス付き (N/S, W/E)
    D.DDN/S D.DDW/E
    サンプル: **33.33N 22.22W**

  - 10 進の度、プレフィックスサイン付き (+/-); (**N/W**) のためのプラスサインは任意です。
    +/-D.DD +/-D.DD

サンプル: **33.33 -22.22**

フォーマットの組み合わせのサンプル:
33.33N -22°44'66.66"
33.33 22°44'66.66"W
33.33 22.c

- *Altova Exif 属性 :位置情報*

  Altova XPath/XQuery エンジンはカスタム属性 **Geolocation**を標準 Exif メタデータタグから生成します。**Geolocation** は、4つのExif タグの連結です :単位の追加された (下のテーブル参照) GPSLatitude、GPSLatitudeRef、GPSLongitude、GPSLongitudeRef。

GPSLatitude	GPSLatitudeRef	GPSLongitude	GPSLongitudeRef	Geolocation
33 51 21.91	S	151 13 11.73	E	33° 51'21.91"S 151° 13'11.73"E

[Top]

XPath/ XQuery 関数 :イメージに関連

以下のイメージに関連したXPath/XQuery 拡張関数は、MapForce の現在のバージョンによりサポートされています。また、次で使用することができます : (i) XSLT コンテキスト内のXPath 式、または (ii) XQuery ドキュメント内のXQuery 式。

関数の名前指定と言語の適用性

Altova 拡張関数はXPath/XQuery 式で使用することができ、XPath、XQuery、およびXSLT 関数の標準ライブラリで使用可能な機能に更なる機能性を与えます。Altova 拡張関数は**Altova 拡張関数名前空間、http://www.altova.com/xslt-extensions** に収められており、**altova:** プレフィックスが、このセクションでは使用されます。製品の今後のバージョンが拡張機能への継続的サポート、または個別の関数の振る舞いは変更する可能性があることにご注意してください。Altova 拡張機能へのサポートに関しては、今後のリリースのドキュメントを参照してください。

XPath 関数 (XSLT 内のXPath 式で使用):	**XP1** **XP2** **XP3**
XSLT 関数 (XSLT 内のXPath 式で使用):	**XSLT1** **XSLT2** **XSLT3**
XQuery 関数 (XQuery 内のXQuery 式で使用):	**XQ1** **XQ3**

suggested-image-file-extension [altova:]

altova:suggested-image-file-extension(Base64String *as string***)** を**string?** とする **XP3** **XQ3**

イメージファイルのBase64 エンコードを引数として、イメージのファイル拡張子を、イメージのBase64エンコード内の記録として返します。返された値は、エンコード内で使用することのできるイメージ型情報を基にしたヒントです。この情報が使用できない場合、空の文字列が返されます。この関数は、Base64 イメージをファイルとして保存し、適切なファイル拡張子を動的取得する際に役に立ちます。

- *サンプル*

- **altova:suggested-image-file-extension**(/MyImages/MobilePhone/Image20141130.01) は 'jpg' を返します。
- **altova:suggested-image-file-extension**($XML1/Staff/Person/@photo) は '' を返します。

上のサンプルでは、関数の引数として与えられたノードはBase64 エンコードイメージを含むと仮定します。最初のサンプルは jpg をファイルの型および拡張子として取得します。二番目のサンプルでは、与えられたBase64 エンコードは使用できる拡張子の情報を提供しません。

mt-transform-image [altova:]

altova:mt-transform-image(Base64Image *as Base64BinaryString*, **Size** *as item()*+, **Rotation** *as xs:integer*, **Quality** *as xs:integer*) を**Base64BinaryString** とする **XP3** **XQ3**

Base64-エンコードイメージを最初の引数として、変換されたBase64-エンコードイメージを返します。第 2、第 3、第 4引数は変換された以下のイメージパラメーターです :サイズ、回転、およびクオリティ。

- サイズ引数には 3つのサイズ変更のオプションがあります。

(X,Y)	絶対ピクセルの値。アスペクト率は保持されません。高さと幅は自動的にイメージの長いおよび短い辺に逢わされるため、高さと幅の指示は関係ありません。 2つの整数アイテムのシーケンスとして値が入力されます。かっこが必要です。
X	X をピクセルの新しい長い辺として、イメージの縦横比のサイズ変更が行われます。アスペクト率は保持されます。値は整数で引用符なしで入力されます。
'X%'	元のディメンションの与えられたパーセンテージにイメージのサイズを変更します。値は文字列として引用符を付けて入力される必要があります。

- 回転は以下の値を持つことができます :90、180、270、-90、-180、-270。これらの値は回転の度数です。正の値はイメージを時計回りに回転させます。負の値はイメージを反時計回りに回転させます。Altova Exif 属性 **OrientationDegree** を使用して、イメージの現在の回転の度数 (0, 90, 180, 270) をイメージのExif **Orientation** タグから取得することができます。ですが、**OrientationDegree** 属性はデータの**Orientation** タグから取得されるため、Exif データ内に **Orientation** タグが存在する場合のみ使用することができます。 (*下のOrientationDegree 説明を参照してください*)。
- クオリティは、0 から100 の値で、JPEG 圧縮のIJG クオリティスケールの値を参照していますが、クオリティのパーセンテージのインジケーターではありません。サイズとクオリティのどちらかを優先すると、もう一方の優先度が下がります。フルカラーソースのイメージでは、75 が通常最高値と見なされています。もし、75 が満足のいく結果をもたらさない場合は、値を上げてください。

メモ: Exif データが元のイメージに存在する場合、変換時に削除され、変換されたイメージにはExif データは存在しません。

サンプル

- **mt-transform-image**(Images/Image[@id='43'], '50%', 90, 75)
  関数は、43 の@id 値を持つイメージ子孫ノード内でBase64-エンコード文字列として保管されているイメージを入力とします。関数は変換されたイメージを返します。変換されたイメージは 50% まで、サイズ変更され、時計回りに 90度回転され、 75のクオリティレベルを与えられます。
- **mt-transform-image**(Images/Image[@id='43'], 400, 90, 75)
  関数は前のサンプルと同じ結果を出しますが、長い辺は400 ピクセルの特定の値に設定されています。元のイメージのアスペクト率は保持されます。
- **mt-transform-image**(Images/Image[@id='43'], (400, 280), image-exif-

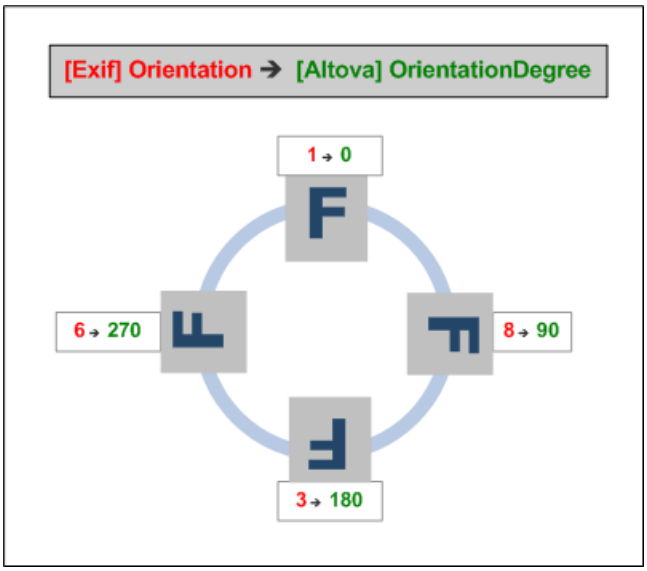
```
data($XML1/$XML1/Images/ReferenceImage)/@OrientationDegree, 75)
```

このサンプルは、前のサンプルと同じイメージを選択し、同じクオリティ値 (75) を設定します。イメージのサイズは 400x280 ピクセルに設定されています。回転値は、ノード内の Base64-エンコードイメージの **@OrientationDegree** 属性から得られます。

Altova Exif 属性 :OrientationDegree

Altova XPath/XQuery エンジンはカスタム属性 **OrientationDegree** を Exif メタデータタグ **Orientation** から生成します。

OrientationDegree は標準 Exif タグ Orientation を正数の値 (1、8、3 または 6) からそれぞれ対応する度数の値 (0, 90, 180, 270) へ下の図に示されているように変換します。2、4、5、7 の Orientation 値の変換はないことに注意してください。これらの向きはイメージ 1 を垂直方向中央軸で反転して、2 の値を持つイメージを取得します。そして、このイメージを 90 度ごとに時計回りにジャンプさせそれぞれ 7、4、および 5 の値を取得します。)。



標準 Exif メタタグ

- ImageWidth
- ImageLength
- BitsPerSample
- Compression
- PhotometricInterpretation
- Orientation
- SamplesPerPixel
- PlanarConfiguration
- YCbCrSubSampling
- YCbCrPositioning
- XResolution
- YResolution
- ResolutionUnit
- StripOffsets
- RowsPerStrip
- StripByteCounts

- JPEGInterchangeFormat
- JPEGInterchangeFormatLength
- TransferFunction
- WhitePoint
- PrimaryChromaticities
- YCbCrCoefficients
- ReferenceBlackWhite
- DateTime
- ImageDescription
- Make
- Model
- Software
- Artist
- Copyright

- ExifVersion
- FlashpixVersion
- ColorSpace
- ComponentsConfiguration
- CompressedBitsPerPixel
- PixelXDimension
- PixelYDimension
- MakerNote
- UserComment
- RelatedSoundFile
- DateTimeOriginal
- DateTimeDigitized
- SubSecTime
- SubSecTimeOriginal
- SubSecTimeDigitized
- ExposureTime
- FNumber
- ExposureProgram
- SpectralSensitivity
- ISOSpeedRatings
- OECF
- ShutterSpeedValue
- ApertureValue
- BrightnessValue
- ExposureBiasValue
- MaxApertureValue
- SubjectDistance
- MeteringMode
- LightSource
- Flash
- FocalLength
- SubjectArea
- FlashEnergy
- SpatialFrequencyResponse
- FocalPlaneXResolution
- FocalPlaneYResolution
- FocalPlaneResolutionUnit
- SubjectLocation
- ExposureIndex
- SensingMethod

- FileSource
- SceneType
- CFAPattern
- CustomRendered
- ExposureMode
- WhiteBalance
- DigitalZoomRatio
- FocalLengthIn35mmFilm
- SceneCaptureType
- GainControl
- Contrast
- Saturation
- Sharpness
- DeviceSettingDescription
- SubjectDistanceRange
- ImageUniqueID

- GPSVersionID
- GPSLatitudeRef
- GPSLatitude
- GPSLongitudeRef
- GPSLongitude
- GPSAltitudeRef
- GPSAltitude
- GPSTimeStamp
- GPSSatellites
- GPSStatus
- GPSMeasureMode
- GPSDOP
- GPSSpeedRef
- GPSSpeed
- GPSTrackRef
- GPSTrack
- GPSImgDirectionRef
- GPSImgDirection
- GPSMapDatum
- GPSDestLatitudeRef
- GPSDestLatitude
- GPSDestLongitudeRef
- GPSDestLongitude
- GPSDestBearingRef
- GPSDestBearing
- GPSDestDistanceRef
- GPSDestDistance
- GPSProcessingMethod
- GPSAreaInformation
- GPSDateStamp
- GPSDifferential

image-exif-data [altova:]

altova:image-exif-data(Base64BinaryString *as string*) を **element?** とする **XP3 XQ3**

Base64 エンコードイメージを引数として、イメージの Exif メタデータを含む **Exif** という名の要素を返します。The

Exif メタデータは Exif 要素の属性の値ペアとして作成されます。属性名は、Base64エンコード内で検出された Exif データタグです。Exif 仕様タグのリストは以下の通りです。ベンダー特有のタグが Exif データ内に存在する場合、タグとその値も属性値のペアとして返されます。標準 Exif メタデータタグに追加して (下のリスト参照) Altova 特有の属性値のペアもまた生成されます。これらの Altova Exif 属性は以下のとおりです。

サンプル

- 属性にアクセスするには、以下の関数を使用します :
  ```
  image-exif-data(//MyImages/Image20141130.01)/@GPSLatitude
  image-exif-data(//MyImages/Image20141130.01)/@Geolocation
  ```
- 全ての属性にアクセスするには、以下の関数を使用します :
  ```
  image-exif-data(//MyImages/Image20141130.01)/@*
  ```
- 全ての属性の名前にアクセスするには、以下の式を使用します :
  ```
  for $i in image-exif-data(//MyImages/Image20141130.01)/@* return
  name($i)
  ```
  関数により返される属性の名前を検出するために役に立ちます。

Altova Exif 属性 :位置情報

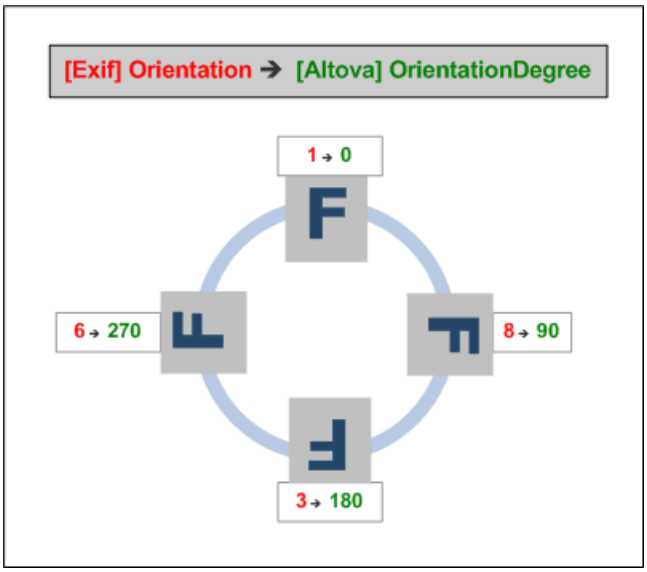
Altova XPath/XQuery エンジンはカスタム属性 **Geolocation** を標準 Exif メタデータタグから生成します。**Geolocation** は、4つの Exif タグの連結です :単位の追加された (下のテーブル参照) GPSLatitude、GPSLatitudeRef、GPSLongitude、GPSLongitudeRef。

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 属性 :OrientationDegree

Altova XPath/XQuery エンジンはカスタム属性 **OrientationDegree** を Exif メタデータタグ **Orientation** から生成します。

OrientationDegree は標準 Exif タグ Orientation を正数の値 (1、8、3 または 6) からそれぞれ対応する度数の値 (0, 90, 180, 270) へ下の図に示されているように変換します。2、4、5、7 の Orientation 値の変換はないことに注意してください。(これらの向きはイメージ 1 を垂直方向中央軸で反転して、2 の値を持つイメージを取得します。そして、このイメージを 90 度ごとに時計回りにジャンプさせそれぞれ 7、4、および 5 の値を取得します。)



標準 Exif メタデータタグのリスト

- ImageWidth
- ImageLength
- BitsPerSample
- Compression
- PhotometricInterpretation
- Orientation
- SamplesPerPixel
- PlanarConfiguration
- YCbCrSubSampling
- YCbCrPositioning
- XResolution
- YResolution
- ResolutionUnit
- StripOffsets
- RowsPerStrip
- StripByteCounts
- JPEGInterchangeFormat
- JPEGInterchangeFormatLength
- TransferFunction
- WhitePoint
- PrimaryChromaticities
- YCbCrCoefficients
- ReferenceBlackWhite
- DateTime
- ImageDescription
- Make
- Model
- Software
- Artist
- Copyright

- ExifVersion
- FlashpixVersion

- ColorSpace
- ComponentsConfiguration
- CompressedBitsPerPixel
- PixelXDimension
- PixelYDimension
- MakerNote
- UserComment
- RelatedSoundFile
- DateTimeOriginal
- DateTimeDigitized
- SubSecTime
- SubSecTimeOriginal
- SubSecTimeDigitized
- ExposureTime
- FNumber
- ExposureProgram
- SpectralSensitivity
- ISOSpeedRatings
- OECF
- ShutterSpeedValue
- ApertureValue
- BrightnessValue
- ExposureBiasValue
- MaxApertureValue
- SubjectDistance
- MeteringMode
- LightSource
- Flash
- FocalLength
- SubjectArea
- FlashEnergy
- SpatialFrequencyResponse
- FocalPlaneXResolution
- FocalPlaneYResolution
- FocalPlaneResolutionUnit
- SubjectLocation
- ExposureIndex
- SensingMethod
- FileSource
- SceneType
- CFAPattern
- CustomRendered
- ExposureMode
- WhiteBalance
- DigitalZoomRatio
- FocalLengthIn35mmFilm
- SceneCaptureType
- GainControl
- Contrast
- Saturation
- Sharpness
- DeviceSettingDescription
- SubjectDistanceRange
- ImageUniqueID

```
------------------------------
```

- GPSVersionID
- GPSLatitudeRef
- GPSLatitude
- GPSLongitudeRef
- GPSLongitude
- GPSAltitudeRef
- GPSAltitude
- GPSTimeStamp
- GPSSatellites
- GPSStatus
- GPSMeasureMode
- GPSDOP
- GPSSpeedRef
- GPSSpeed
- GPSTrackRef
- GPSTrack
- GPSImgDirectionRef
- GPSImgDirection
- GPSMapDatum
- GPSDestLatitudeRef
- GPSDestLatitude
- GPSDestLongitudeRef
- GPSDestLongitude
- GPSDestBearingRef
- GPSDestBearing
- GPSDestDistanceRef
- GPSDestDistance
- GPSProcessingMethod
- GPSAreaInformation
- GPSDateStamp
- GPSDifferential

[トップ]

XPath/ XQuery 関数 :数値

Altova の数値拡張関数は XPath とXQuery 式で使用することができ、XML スキーマの異なる日付および時刻データ型で保存されているデータを処理するための追加機能を提供します。このセクションの関数は、Altova の**XPath 3.0** および**XQuery 3.0** エンジンで使用することができます。これらの関数は、XPath/XQuery コンテキストで使用することができます。

関数の名前指定と言語の適用性

Altova 拡張関数はXPath/XQuery 式で使用することができ、XPath、XQuery、およびXSLT 関数の標準ライブラリで使用可能な機能に更なる機能性を与えます。Altova 拡張関数は**Altova 拡張関数名前空間、** `http://www.altova.com/xslt-extensions` に収められており、`altova:` プレフィックスが、このセクションでは使用されます。製品の今後のバージョンが拡張機能への継続的サポート、または個別の関数の振る舞いは変更する可能性があることにご注意ください。Altova 拡張機能へのサポートに関しては、今後のリリースのドキュメントを参照してください。

XPath 関数 (XSLT 内のXPath 式で使用):	XP1 XP2 XP3
XSLT 関数 (XSLT 内のXPath 式で使用):	XSLT1 XSLT2

	XSLT3
XQuery 関数 (XQuery 内のXQuery 式で使用):	XQ1 XQ3

自動付番関数

generate-auto-number [altova:]

altova:generate-auto-number(ID *as xs:string*, **StartsWith** *as xs:double*, **Increment** *as xs:double*, **ResetOnChange** *as xs:string*) を xs:integer とする XP1 XP2 XQ1 XP3 XQ3

関数が呼び出される度に番号を生成します。関数が初めて呼び出される際に生成される最初の番号は StartsWith 引数により指定されます。その後の関数の呼び出しは新しい番号を生成します。この番号は前に生成された番号を、Increment 引数で指定された値ごとにインクリメントします。実質的には、altova:generate-auto-number 関数は、ID 引数により指定されたカウンターを作成し、このカウンターが関数が呼び出されるごとにインクリメントされます。ResetOnChange 引数の値が、前の関数呼び出しから変更されると、生成される番号の値がの StartsWith 値にリセットされます。自動付番は altova:reset-auto-number 関数を使用してリセットすることができます。

サンプル

- **altova:generate-auto-number**("ChapterNumber", 1, 1, "SomeString") は、関数が呼び出される度に番号を 1 つ返します。starting with 1, から始まり、関数が呼び出される度に 1 ずつインクリメントします。その後の呼び出しの 4 番目の"SomeString"引数がある限り、インクリメントは継続されます。4 番目の引数の値が変更されると、(ChapterNumber と呼ばれる)カウンターは 1 にリセットされます。ChapterNumber の値も altova:reset-auto-number 関数の呼び出しにより altova:reset-auto-number("ChapterNumber") ようにリセットされます。

reset-auto-number [altova:]

altova:reset-auto-number(ID *as xs:string*) XP1 XP2 XQ1 XP3 XQ3

この関数は、ID 引数内で名づけられた自動付番カウンターの番号をリセットします。引数内のカウンター名を作成した関数の引数により指定された数値にリセットされます。altova:generate-auto-number

サンプル

- **altova:reset-auto-number**("ChapterNumber") は、altova:generate-auto-number 関数により作成された ChapterNumber という名の自動付番カウンターをリセットします。
- ChapterNumber を作成した altova:generate-auto-number 関数の StartsWith 引数の値に数値をリセットします。

[Top]

数値関数

hex-string-to-integer [altova:]

altova:hex-string-to-integer(HexString *as xs:string*) を xs:integer とする XP3 XQ3

10 進のシステム(10進)内の正数の16進の同値の文字列引数を必要とし、10進の整数を返します。

サンプル

- **altova:hex-string-to-integer**('1') は 1 を返します。
- **altova:hex-string-to-integer**('9') は 9 を返します。

- **altova:hex-string-to-integer**('A') は10 を返します。
- **altova:hex-string-to-integer**('B') は11 を返します。
- **altova:hex-string-to-integer**('F') は15 を返します。
- **altova:hex-string-to-integer**('G') はエラーを返します。
- **altova:hex-string-to-integer**('10') は16 を返します。
- **altova:hex-string-to-integer**('01') は1 を返します。
- **altova:hex-string-to-integer**('20') は32 を返します。
- **altova:hex-string-to-integer**('21') は33 を返します。
- **altova:hex-string-to-integer**('5A') は90 を返します。
- **altova:hex-string-to-integer**('USA') はエラーを返します。

integer-to-hex-string [altova:]

altova:integer-to-hex-string(Integer *as xs:integer*) を**xs:string** とする XP3 XQ3

正数を引数として必要とし、文字列として自身のベース16の同値を返します。

サンプル

- **altova:integer-to-hex-string**(1) は'1' を返します。
- **altova:integer-to-hex-string**(9) は'9' を返します。
- **altova:integer-to-hex-string**(10) は'A' を返します。
- **altova:integer-to-hex-string**(11) は'B' を返します。
- **altova:integer-to-hex-string**(15) は'F' を返します。
- **altova:integer-to-hex-string**(16) は'10' を返します。
- **altova:integer-to-hex-string**(32) は'20' を返します。
- **altova:integer-to-hex-string**(33) は'21' を返します。
- **altova:integer-to-hex-string**(90) は'5A' を返します。

[トップ]

数値をフォーマットする関数

generate-auto-number [altova:]

altova:generate-auto-number(ID *as xs:string*, **StartsWith** *as xs:double*, **Increment** *as xs:double*, **ResetOnChange** *as xs:string*) を xs:integer とする XP1 XP2 XQ1 XP3 XQ3

関数が呼び出される度に番号を生成します。関数が初めて呼び出される際に生成される最初の番号は StartsWith 引数により指定されます。その後の関数の呼び出しは新しい番号を生成します。この番号は前に生成された番号を、Increment 引数で指定された値ごとにインクリメントします。実質的には、altova:generate-auto-number 関数は、ID 引数により指定されたカウンターを作成し、このカウンターが関数が呼び出されるごとにインクリメントされます。ResetOnChange 引数の値が、前の関数呼び出しから変更されると、生成される番号の値がの StartsWith 値にリセットされます。自動付番は altova:reset-auto-number 関数を使用してリセットすることができます。

サンプル

- **altova:generate-auto-number**("ChapterNumber", 1, 1, "SomeString") は、関数が呼び出される度に番号を1つ返します。starting with 1, から始まり、関数が呼び出される度に1 ずつインクリメントします。その後の呼び出しの 4番目の"SomeString"引数がである限り、インクリメントは継続されます。4番目の引数の値が変更されると、(ChapterNumber と呼ばれる)カウンターは 1にリセットされます。ChapterNumber の値も altova:reset-auto-number 関数の呼び出しによりの

altova:reset-auto-number("ChapterNumber")ようにリセットされます。

[トップ]

XPath/ XQuery 関数 :シーケンス

Altova のシーケンス拡張関数は XPath とXQuery 式で使用することができ、XML スキーマの異なる日付および時刻データ型で保存されているデータを処理するための追加機能を提供します。このセクションの関数は、Altova の**XPath 3.0** および**XQuery 3.0** エンジンで使用することができます。これらの関数は、XPath/XQuery コンテキストで使用することができます。

関数の名前指定と言語の適用性

Altova 拡張関数はXPath/XQuery 式で使用することができ、XPath、XQuery、およびXSLT 関数の標準ライブラリで使用可能な機能に更なる機能性を与えます。Altova 拡張関数は**Altova 拡張関数名前空間、http://www.altova.com/xslt-extensions** に収められており、**altova:** プレフィックスが、このセクションでは使用されます。製品の今後のバージョンが拡張機能への継続的サポート、または個別の関数の振る舞いは変更する可能性があることにご注意してください。Altova 拡張機能へのサポートに関しては、今後のリリースのドキュメントを参照してください。

XPath 関数 (XSLT 内のXPath 式で使用):	XP1 XP2 XP3
XSLT 関数 (XSLT 内のXPath 式で使用):	XSLT1 XSLT2 XSLT3
XQuery 関数 (XQuery 内のXQuery 式で使用):	XQ1 XQ3

attributes [altova:]

altova:attributes(AttributeName *as xs:string***)** を**attribute()*** とする XP3 XQ3
入力引数 AttributeName 内で与えられた名前と同じローカル名を持つすべての属性を返します。検索は大文字と小文字を区別し、attribute:: 軸に対して行われます。これは、コンテキストノードが親要素ノードである必要があることを意味します。

サンプル

- **altova:attributes**("MyAttribute") はMyAttribute()* を返します。

altova:attributes(AttributeName *as xs:string*, **SearchOptions** *as xs:string***)** as **attribute()*** XP3 XQ3
入力引数 AttributeName 内で与えられた名前と同じローカル名を持つすべての属性を返します。検索は大文字と小文字を区別し、attribute:: 軸に対して行われます。コンテキストノードが親要素ノードである必要があります。第 2引数はオプションのフラグを含みます。使用することのできるフラグは以下の通りです :
r = 正規表現検索を切り替えます ;AttributeName は正規表現検索文字列である必要があります ;
f = このオプションが指定されている場合、AttributeName は完全一致を提供します。それ以外の場合は、AttributeName は属性名に部分的に一致するとその属性を返します。例えば : **f** が指定されていない場合、MyAtt はMyAttribute を返します。
i = 大文字と小文字を一致させる検索に切り替えます。
p = 検索に名前空間プレフィックスを含みます。AttributeName は、名前空間プレフィックスを含みます。例えば : altova:MyAttribute。
フラグは順序に関わりなく書き込むことができます。無効なフラグはエラーを生成します。1つまたは複数のフラグを省略することができます。空の文字列は許可されていますが、引数を1つしか持たない関数と同じ効果を持ちます (前の署名)、ですが、空のシーケンスは第 2の引数としては許可されていません。

サンプル

- **altova:attributes**("MyAttribute", "rfip") はMyAttribute()* を返します。

- **altova:attributes**("MyAttribute", "pri") は MyAttribute()* を返します。
- **altova:attributes**("MyAtt", "rip") は MyAttribute()* を返します。
- **altova:attributes**("MyAttributes", "rfip") は不一致を返します。
- **altova:attributes**("MyAttribute", "") は MyAttribute()* を返します。
- **altova:attributes**("MyAttribute", "Rip") 認識されないフラグエラーを返します。
- **altova:attributes**("MyAttribute",) 見つからない第 2引数を返します。

elements [altova:]

altova:elements(ElementName *as xs:string*) を **element()*** とする XP3 XQ3
入力引数 ElementName で与えられた名前と同じローカル名を持つすべての要素を返します。検索は大文字と小文字を区別して child:: 軸に対して実行されます。コンテキストノードは、検索される要素の親ノードである必要があります。

サンプル

- **altova:elements**("MyElement") は MyElement()* を返します。

altova:elements(ElementName *as xs:string*, **SearchOptions** *as xs:string*) as **element()*** XP3 XQ3
入力引数 ElementName 内で与えられた名前と同じローカル名を持つすべての属性を返します。検索は大文字と小文字を区別し、child:: 軸に対して行われます。コンテキストノードが親要素ノードである必要があります。第 2引数はオプションのフラグを含みます。使用することのできるフラグは以下の通りです：
r = 正規表現検索を切り替えます；ElementName は正規表現検索文字列である必要があります；
f = このオプションが指定されている場合、ElementName は完全一致を提供します。それ以外の場合は、ElementName は属性名に部分的に一致するとその属性を返します。例えば：**f** が指定されていない場合、MyElem は MyElement を返します。
i = 大文字と小文字を一致させる検索に切り替えます。
p = 検索に名前空間プレフィックスを含みます。ElementName は、名前空間プレフィックスを含みます。例えば：altova:MyElement。
フラグは順序に関わりなく書き込むことができます。無効なフラグはエラーを生成します。1つまたは複数のフラグを省略することができます。空の文字列は許可されていますが、引数を1つしか持たない関数と同じ効果を持ちます (前の署名)、ですが、空のシーケンスは第 2の引数としては許可されていません。

サンプル

- **altova:elements**("MyElement", "rip") は MyElement()* を返します。
- **altova:elements**("MyElement", "pri") は MyElement()* を返します。
- **altova:elements**("MyElement", "") は MyElement()* を返します。
- **altova:attributes**("MyElem", "rip") は MyElement()* を返します。
- **altova:attributes**("MyElements", "rfip") 不一致を課wします。
- **altova:elements**("MyElement", "Rip") 認識されないフラグエラーを返します。
- **altova:elements**("MyElement",) 見つからない第 2引数を返します。

find-first [altova:]

altova:find-first((Sequence *as item()**), **(Condition(Sequence-Item** *as xs:boolean*)) を **item()?** とする XP3 XQ3
この関数は 2つの引数を必要とします。最初の引数は 1つ、または 1つ以上のデータ型のアイテムのシーケンスです。第 2の引数Condition は、(**1** のアリティを持つ) 1つの引数 を必要とし、boolean を返す XPath 関数に対する参照です。Condition で参照された関数の代わりに、**Sequence** の各アイテムが提出されます。(注意：この関数は 1つの引数のみを必要とします。) **Condition** 内の関数に **true()** と評価させる最初の **Sequence** アイテムは、**altova:find-first**、反復の終了の結果として返されます。

サンプル

- `altova:find-first(5 to 10, function($a) {$a mod 2 = 0})` は `xs:integer 6` を返します。

Condition 引数は、**$a** という名のインライン関数を宣言し、定義します。XPath 3.0 インライン関数 **function()** を参照します。**altova: Sequence** 引数内の各アイテムでは、**find-first** がパスされ、代わりに、**$a** を入力値とします。入力値は関数定義 (`$a mod 2 = 0`)内の条件に対してテストされます。この条件を満たす最初の入力値が**altova:find-first** (この場合は**6**) の結果として返されます。

- `altova:find-first((1 to 10), (function($a) {$a+3=7}))` は `xs:integer 4` を返します。

更なるサンプル

ファイル**C:\Temp\Customers.xml** が存在する場合 :

- `altova:find-first( ("C:\Temp\Customers.xml", "http://www.altova.com/index.html"), (doc-available#1) )` は `xs:string C:\Temp\Customers.xml` を返します。

ファイル**C:\Temp\Customers.xml** が存在せず、**http://www.altova.com/index.html** が存在する場合 :

- `altova:find-first( ("C:\Temp\Customers.xml", "http://www.altova.com/index.html"), (doc-available#1) )` は `xs:string http://www.altova.com/index.html` を返します。

ファイル**C:\Temp\Customers.xml** が存在せず、**http://www.altova.com/index.html** も存在しない場合 :

- `altova:find-first( ("C:\Temp\Customers.xml", "http://www.altova.com/index.html"), (doc-available#1) )` 結果無しを返します。

上のサンプルについての注意点

- XPath 3.0 関数 `doc-available` はURI として使用され、ドキュメントノードが提出されたURI で検出される場合 `true` を返す単一の引数を必要とします。 (ですから、提出されたURI でのドキュメントはXML ドキュメントである必要があります。)
- `doc-available` 関数は、`altova:find-first` の第 2引数である、**Condition** で使用することができます。これは、 1つの引数 (アリティ=1)のみを必要とするからであり、`item()`を入力 (URI として使用される文字列) として、boolean の値を返すからです。
- `doc-available` 関数は、参照されているだけで、呼び出されていない点に注意してください。アタッチされている`#1` サフィックスは関数が 1つのアリティであることを表示するためです。`doc-available#1` の意味は以下のとおりです :アリティ=1 を持つ doc-availabe() 関数を使用し、最初のシーケンスの各アイテムの代わりに単一引数としてパスします。 この結果、 2つの文字列の各自つは、文字列をURI として使用し、URI にドキュメントノードが存在するかテストする**doc-available()**にパスされます。1つが各当する場合、**doc-available()** 関数は`true()` を評価し、シーケンス内のその文字列のインデックス ポジションは、 **altova:find-first** 関数の結果として返されます。doc-available() 関数に関しての注意点 :相対パスは、デフォルトで関数がロードされるXML ドキュメントの現在のベースURI に対して相対的に解決されます。

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*)) を**item()*** とする **XP3** **XQ3**

この関数は 3つの引数を必要とします :

- 最初の 2 つの引数 **Seq-01** と**Seq-02**, は、1 つまたは 1 つ以上のデータ型のアイテムです。
- 第 3 の引数 **Condition** は、(2 のアリティを持つ) 2 つの引数を必要とし、boolean を返す XPath 関数に対しての参照です。

Seq-01 と**Seq-02** のアイテムは指定された組み合わせ (各シーケンスからの 1 つずつにアイテムで構成されるペア) で、**Condition** 内の関数の引数としてパスされました。組み合わせは以下のように指定されています。

```
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)
```

Condition 関数に**true()** と評価するように指示する最初のペアは**altova:find-first-combination** の結果として返されます。以下の点に注意してください: (i) **Condition** 関数が提出された引数ペア内で繰り返され、**true()** を評価しない場合、**altova:find-first-combination** は、結果を返しません; (ii) **altova:find-first-combination** の結果が常に (データ型の) アイテムのペアである場合またはアイテムでない場合。

サンプル

- **altova:find-first-combination**(11 to 20, 21 to 30, function($a, $b) {$a+$b = 32}) は xs:integers (11, 21) のシーケンスを返します。
- **altova:find-first-combination**(11 to 20, 21 to 30, function($a, $b) {$a+$b = 33}) は xs:integers (11, 22) のシーケンを返します。
- **altova:find-first-combination**(11 to 20, 21 to 30, function($a, $b) {$a+$b = 34}) は xs:integers (11, 23) のシーケンを返します。

find-first-pair [altova:]

altova:find-first-pair((Seq-01 *as item()**)**, **(Seq-02** *as item()**)**, **(Condition(Seq-01-Item, Seq-02-Item** *as xs:boolean***))** を**item()*** とする **XP3 XQ3**

この関数は 3 つの引数を必要とします:

- 最初の 2 つの引数 **Seq-01** と**Seq-02**, は、1 つまたは 1 つ以上のデータ型のアイテムです。
- 第 3 の引数 **Condition** は、(2 のアリティを持つ) 2 つの引数を必要とし、boolean を返す XPath 関数に対しての参照です。

Seq-01 と**Seq-02** のアイテムは指定された組み合わせで、**Condition** 内の関数の引数としてパスされました。組み合わせは以下のように指定されています。

```
If   Seq-01 = X1, X2, X3 ... Xn
And  Seq-02 = Y1, Y2, Y3 ... Yn
Then (X1 Y1), (X2 Y2), (X3 Y3) ... (Xn Yn)
```

Condition 関数に**true()** と評価するように指示する最初のペアは**altova:find-first-pair** の結果として返されます。以下の点に注意してください: (i) **Condition** 関数が提出された引数ペア内で繰り返され、**true()** を評価しない場合、**altova:find-first-pair** は、結果を返しません; (ii) **altova:find-first-combination** の結果が常に (データ型の) アイテムのペアである場合またはアイテムでない場合。

サンプル

- **altova:find-first-pair**(11 to 20, 21 to 30, function($a, $b) {$a+$b = 32}) は xs:integers (11, 21) のシーケンスを返します。
- **altova:find-first-pair**(11 to 20, 21 to 30, function($a, $b) {$a+$b = 33}) は*結果無し*を返します。

上の 2 つのサンプルに表示される通り、ペアの順序は以下の通りです: (11, 21) (12, 22) (13, 23)...(20, 30)。(33 を返す指示されたペアがないため) この理由で第 2 のサンプルは*結果無し*を返

します。

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***))** を**xs:integer** とする **XP3 XQ3**

この関数は 3 つの引数を必要とします :

- 最初の 2 つの引数 **Seq-01** と**Seq-02**, は、1 つまたは1つ以上のデータ型のアイテムです。
- 第 3の引数 **Condition** は、(2 のアリティを持つ) 2 つの引数を必要とし、boolean を返す XPath 関数に対しての参照です。

Seq-01 と**Seq-02** のアイテムは指定された組み合わせで、**Condition** 内の関数の引数としてパスされました。組み合わせは以下のように指定されています。

```
If   Seq-01 = X1, X2, X3 ... Xn
And  Seq-02 = Y1, Y2, Y3 ... Yn
Then (X1 Y1), (X2 Y2), (X3 Y3) ... (Xn Yn)
```

Condition 関数に**true()**を評価させる、最初に指示されたペアのインデックスポジションは、は **altova:find-first-pair-pos** の結果として返されます。関数が提出された引数ペア内で繰り返され、**true()**を一度も評価しない場合、**altova:find-first-pair-pos** *結果無し*が返します。

サンプル

- **altova:find-first-pair-pos**(11 to 20, 21 to 30, function($a, $b) {$a+$b = 32}) は1 を返します
- **altova:find-first-pair-pos**(11 to 20, 21 to 30, function($a, $b) {$a+$b = 33}) は*結果無し*を返します。

上の 2 つのサンプルに表示される通り、ペアの順序は以下の通りです :(11, 21) (12, 22) (13, 23)...(20, 30)。最初のサンプルでは、最初のペアは**Condition** 関数に**true()**を評価させ、シーケンス内のインデックスポジションには**1** が返されます。第 2 のサンプルでは、33 を返すペアがないため、*結果無し*が返します。

find-first-pos [altova:]

altova:find-first-pos((Sequence *as item()**)**,* **(Condition(Sequence-Item** *as xs:boolean***))** を**xs:integer** とする **XP3 XQ3**

この関数は 2 つの引数を必要とします。最初の引数は 1 つ、または 1 つ以上のデータ型のアイテムのシーケンスです。第 2 の引数 Condition は、(**1** のアリティを持つ) 1 つの引数 を必要とし、boolean を返す XPath 関数に対する参照です。Condition で参照された関数の代わりに、**Sequence** の各アイテムが提出されます。(*注意 : この関数は 1 つの引数のみを必要とします。*)**Condition** 内の関数に**true()**と評価させる最初の **Sequence** アイテムは、**altova:find-first-pos** の結果として返された**Sequence** 内インデックスポジションを持ちます。

サンプル

- **altova:find-first-pos**(5 to 10, function($a) {$a mod 2 = 0}) は xs:integer 2 を返します。

Condition 引数は、**$a** という名のインライン関数を宣言し、定義します。XPath 3.0 インライン関数 **function()**を参照します。 **Sequence** 引数内の各アイテムでは、**find-first-pos** がパスされ、代わりに、**$a** を入力値とします。入力値は関数定義 ($a mod 2 = 0)内の条件に対してテストされます。この条件を満たす最初の入力値が**altova:find-first-pos** ((シーケンス内で条件を満たす最初の値である 6 がシーケンスのインデックス位置 2 にあるためこの場合は 2、)の結果として返されます。

- **altova:find-first-pos**((2 to 10), (function($a) {$a+3=7})) は xs:integer 3 を返します。

更なるサンプル
ファイル **C:\Temp\Customers.xml** が存在する場合：

- **altova:find-first-pos**(("C:\Temp\Customers.xml", "http://www.altova.com/index.html"), (doc-available#1)) returns 1

ファイル **C:\Temp\Customers.xml** が存在せず、**http://www.altova.com/index.html** が存在する場合：

- **altova:find-first-pos**(("C:\Temp\Customers.xml", "http://www.altova.com/index.html"), (doc-available#1)) returns 2

ファイル **C:\Temp\Customers.xml** が存在せず、**http://www.altova.com/index.html** も存在しない場合：

- **altova:find-first-pos**(("C:\Temp\Customers.xml", "http://www.altova.com/index.html"), (doc-available#1)) 結果無しを返します。

上のサンプルについての注意点

- XPath 3.0 関数 doc-available は URI として使用され、ドキュメントノードが提出された URI で検出される場合 true を返す単一の引数を必要とします。(ですから、提出された URI でのドキュメントは XML ドキュメントである必要があります。)
- doc-available 関数は、altova:find-first-pos の第 2 引数である、**Condition** で使用することができます。これは、1 つの引数 (アリティ=1) のみを必要とするからであり、item() を入力 (URI として使用される文字列) として、boolean の値を返すからです。
- doc-available 関数は、参照されているだけで、呼び出されていない点に注意してください。アタッチされている #1 サフィックスは関数が 1 つのアリティであることを表示するためです。doc-available#1 の意味は以下のとおりです：アリティ=1 を持つ doc-availabe() 関数を使用し、最初のシーケンスの各アイテムの代わりに単一引数としてパスします。この結果、2 つの文字列の各自つは、文字列を URI として使用し、URI にドキュメントノードが存在するかテストする **doc-available()** にパスされます。1 つが各当する場合、**doc-available()** 関数は true() を評価し、シーケンス内のその文字列のインデックス ポジションは、**altova:find-first-pos** 関数の結果として返されます。doc-available() 関数に関しての注意点：相対パスは、デフォルトで関数がコードされる XML ドキュメントの現在のベース URI に対して相対的に解決されます。

substitute-empty [altova:]

altova:substitute-empty(FirstSequence *as item()**, **SecondSequence** *as item()***)** を **item()*** とする XP3 XQ3

FirstSequence が空の場合、SecondSequence を返します。FirstSequence が空でない場合、FirstSequence を返します。

サンプル

- **altova:substitute-empty**((1,2,3), (4,5,6)) は (1,2,3) を返します。
- **altova:substitute-empty**((), (4,5,6)) は (4,5,6) を返します。

XPath/ XQuery 関数 :文字列

Altova の文字列拡張関数は XPath と XQuery 式で使用することができ、XML スキーマの異なる日付および時刻データ型で保存されているデータを処理するための追加機能を提供します。このセクションの関数は、Altova の**XPath 3.0** および**XQuery 3.0** エンジンで使用することができます。これらの関数は、XPath/XQuery コンテキストで使用することができます。

関数の名前指定と言語の適用性

Altova 拡張関数は XPath/XQuery 式で使用することができ、XPath、XQuery、および XSLT 関数の標準ライブラリで使用可能な機能に更なる機能性を与えます。Altova 拡張関数は**Altova 拡張関数名前空間、`http://www.altova.com/xslt-extensions`** に収められており、**`altova:`** プレフィックスが、このセクションでは使用されます。製品の今後のバージョンが拡張機能への継続的サポート、または個別の関数の振る舞いは変更する可能性があることにご注意ください。Altova 拡張機能へのサポートに関しては、今後のリリースのドキュメントを参照してください。

XPath 関数 (XSLT 内の XPath 式で使用):	**XP1** **XP2** **XP3**
XSLT 関数 (XSLT 内の XPath 式で使用):	**XSLT1** **XSLT2** **XSLT3**
XQuery 関数 (XQuery 内の XQuery 式で使用):	**XQ1** **XQ3**

`camel-case [altova:]`

`altova:camel-case(InputString` *`as xs:string`***`)`** を**`xs:string`** とする **XP3** **XQ3**
入力文字列を**`InputString`** をキャメルケースで返します。文字列は空白スペースのショートカットである)正規表現 **`'\s'`** を使用して分析されます。空白文字または連続する空白文字のシーケンスの後の最初の非空白スペース文字は大文字です。出力文字列の最初の文字は大文字です。

サンプル

- **`altova:camel-case`**`("max")` `Max` を返します。
- **`altova:camel-case`**`("max max")` `Max Max` を返します。
- **`altova:camel-case`**`("file01.xml")` `File01.xml` を返します。
- **`altova:camel-case`**`("file01.xml file02.xml")` `File01.xml File02.xml` を返します。
- **`altova:camel-case`**`("file01.xml  file02.xml")` `File01.xml  File02.xml` を返します。
- **`altova:camel-case`**`("file01.xml -file02.xml")` `File01.xml -file02.xml` を返します。

`altova:camel-case(InputString` *`as xs:string`***`, SplitChars`** *`as xs:string`***`, IsRegex`** *`as xs:boolean`***`)`** を**`xs:string`** とする **XP3** **XQ3**
`SplitChars` を使用して、次の大文字をトリガーする文字を決定し、入力文字列を**`InputString`** キャメルケースに変換します。**`SplitChars`** は、**`IsRegex = true()`**の場合、または、**`IsRegex = false()`**の場合、プレーン文字は正規表現として使用されます。出力文字列の最初の文字は大文字です。

サンプル

- **`altova:camel-case`**`("setname getname", "set|get", true())` `setName getName` を返します。
- **`altova:camel-case`**`("altova\documents\testcases", "\", false())` `Altova \Documents\Testcases` を返します。

`char [altova:]`

altova:char(Position *as xs:integer*) を**xs:string** とする XP3 XQ3
xs:string.Position に対してのコンテキストアイテムの値を変換することにより得られた文字列内の Position 引数により指定されたポジションにある文字を含む文字列を返します。引数により提出されたインデックスに文字が存在しない場合、結果文字列は空です。

サンプル

コンテキストアイテムが**1234ABCD** の場合：

- **altova:char**(2) は2 を返します。
- **altova:char**(5) はA を返します。
- **altova:char**(9) は空の文字列を返します。
- **altova:char**(-2) は空の文字列を返します。

altova:char(InputString *as xs:string*, **Position** *as xs:integer*) を**xs:string** とする XP3 XQ3
引数として提出された文字列内の Position 引数により指定されたポジションでの文字を含む文字列を返します。Position 引数により提出されたインデックスに文字が存在しない場合、結果文字列は空です。

サンプル

- **altova:char**("2014-01-15", 5) は- を返します。
- **altova:char**("USA", 1) はU を返します。
- **altova:char**("USA", 10) は空の文字列を返します。
- **altova:char**("USA", -2) は空の文字列を返します。

first-chars [altova:]

altova:first-chars(X-Number *as xs:integer*) を**xs:string** とする XP3 XQ3
xs:string に対するコンテキストアイテムの値を変換することにより得られた最初のX-Number 文字を含む文字列を返します。

サンプル

コンテキストアイテムが**1234ABCD** の場合：

- **altova:first-chars**(2) は12 を返します。
- **altova:first-chars**(5) は1234A を返します。
- **altova:first-chars**(9) は1234ABCD を返します。

altova:first-chars(InputString *as xs:string*, **X-Number** *as xs:integer*) を**xs:string** とする XP3 XQ3
InputString 引数として提出された文字列の最初の文字を含む文字列を返します。

サンプル

- **altova:first-chars**("2014-01-15", 5) は2014- を返します。
- **altova:first-chars**("USA", 1) はU を返します。

last-chars [altova:]

altova:last-chars(X-Number *as xs:integer*) を**xs:string** とする XP3 XQ3
xs:string に対してのコンテキストアイテムの値の変換より取得された文字列の最後のX-Number 文字を含んでいる文字列を返します。

サンプル

コンテキストアイテムが**1234ABCD** の場合：

- **altova:last-chars**(2) はCD を返します。
- **altova:last-chars**(5) は4ABCD を返します。

- **altova:last-chars**(9) は1234ABCD を返します。

altova:last-chars(InputString *as xs:string*, **X-Number** *as xs:integer*) as **xs:string** とする XP3 XQ3
引数として提出された文字列の最後のX-Number 文字を含んでいる文字列を返します。

サンプル

- **altova:last-chars**("2014-01-15", 5) は01-15 を返します。
- **altova:last-chars**("USA", 10) はUSA を返します。

pad-string-left [altova:]

altova:pad-string-left(StringToPad *as xs:string*, **StringLength** *as xs:integer*, **PadCharacter** *as xs:string*) を**xs:string** とする XP3 XQ3
PadCharacter 引数は1文字です。文字列の左側にパッドされ、この数が、StringLength 引数の整数の値と等しくなるようにStringToPad の文字の数を増やします。StringLength 引数は任意の整数の値（正数または負数）を持つことができますが、パディングは、StringLength の値がStringToPad 内の文字数よりも多い場合の場合のみ発生します。もし、StringToPad がStringLength の値よりも多くの文字数も持つ場合、StringToPad は変更されません。

サンプル

- **altova:pad-string-left**('AP', 1, 'Z') は'AP' を返します。
- **altova:pad-string-left**('AP', 2, 'Z') は'AP' を返します。
- **altova:pad-string-left**('AP', 3, 'Z') は'ZAP' を返します。
- **altova:pad-string-left**('AP', 4, 'Z') は'ZZAP' を返します。
- **altova:pad-string-left**('AP', -3, 'Z') は'AP' を返します。
- **altova:pad-string-left**('AP', 3, 'YZ')は[パッド文字が長すぎます] エラーを返します。

pad-string-right [altova:]

altova:pad-string-right(StringToPad *as xs:string*, **StringLength** *as xs:integer*, **PadCharacter** *as xs:string*) を**xs:string** とする XP3 XQ3
PadCharacter 引数は1文字です。文字列の右側にパッドされ、この数が、StringLength 引数の整数の値と等しくなるようにStringToPad の文字の数を増やします。StringLength 引数は任意の整数の値（正数または負数）を持つことができますが、パディングは、StringLength の値がStringToPad 内の文字数よりも多い場合の場合のみ発生します。もし、StringToPad がStringLength の値よりも多くの文字数も持つ場合、StringToPad は変更されません。

サンプル

- **altova:pad-string-right**('AP', 1, 'Z') を'AP' を返します。
- **altova:pad-string-right**('AP', 2, 'Z') を'AP' を返します。
- **altova:pad-string-right**('AP', 3, 'Z') を'APZ' を返します。
- **altova:pad-string-right**('AP', 4, 'Z') を'APZZ' を返します。
- **altova:pad-string-right**('AP', -3, 'Z') を'AP' を返します。
- **altova:pad-string-right**('AP', 3, 'YZ') は[パッド文字が長すぎます] エラーを返します。

repeat-string [altova:]

altova:repeat-string(InputString *as xs:string*, **Repeats** *as xs:integer*) を**xs:string** とする XP2 XQ1 XP3 XQ3
最初のInputString引数により構成される文字列、Repeats 回繰り返してを生成します。

サンプル

- **altova:repeat-string**("Altova #", 3) は"Altova #Altova #Altova #" を返します。

substring-after-last [altova:]

altova:substring-after-last(MainString *as xs:string*, **CheckString** *as xs:string*) を**xs:string** とする **XP3** **XQ3**

CheckString がMainString 内で検出された場合、MainString 内のCheckString が発生した後のサブ文字列が返されます。MainString 内でCheckString が検出されない場合、空の文字列が返されます。CheckString が空の文字列の場合、MainString 全体が返されます。一度以上発生する場合、CheckString の最後の発生の後の サブ文字列が返されます。

サンプル

- **altova:substring-after-last**('ABCDEFGH', 'B') は'CDEFGH' を返します。
- **altova:substring-after-last**('ABCDEFGH', 'BC') は'DEFGH' を返します。
- **altova:substring-after-last**('ABCDEFGH', 'BD') は'' を返します。
- **altova:substring-after-last**('ABCDEFGH', 'Z') は'' を返します。
- **altova:substring-after-last**('ABCDEFGH', '') は'ABCDEFGH' を返します。
- **altova:substring-after-last**('ABCD-ABCD', 'B') は'CD' を返します。
- **altova:substring-after-last**('ABCD-ABCD-ABCD', 'BCD') は'' を返します。

substring-before-last [altova:]

altova:substring-before-last(MainString *as xs:string*, **CheckString** *as xs:string*) を**xs:string** とする **XP3** **XQ3**

CheckString がMainString 内で検出された場合、MainString 内のCheckString が発生する前のサブ文字列が返されます。MainString 内でCheckString が一度以上発生する場合、CheckString の最後の発生の前の サブ文字列が返されます。

サンプル

- **altova:substring-before-last**('ABCDEFGH', 'B') は'A' を返します。
- **altova:substring-before-last**('ABCDEFGH', 'BC') は'A' を返します。
- **altova:substring-before-last**('ABCDEFGH', 'BD') は'' を返します。
- **altova:substring-before-last**('ABCDEFGH', 'Z') は'' を返します。
- **altova:substring-before-last**('ABCDEFGH', '') は'' を返します。
- **altova:substring-before-last**('ABCD-ABCD', 'B') は'ABCD-A' を返します。
- **altova:substring-before-last**('ABCD-ABCD-ABCD', 'ABCD') は'ABCD-ABCD-' を返します。

substring-pos [altova:]

altova:substring-pos(StringToCheck *as xs:string*, **StringToFind** *as xs:string*) を**xs:integer** とする **XP3** **XQ3**

StringToCheck 内でのStringToFind の最初の発生の文字位置を整数として返します。StringToCheck の最初の文字は、位置 1 にあります。StringToFind がStringToCheck 内で発生しない場合、整数 0 が返されます。第 2 またはその後のStringToCheck, 発生を確認するには、この関数の次の署名を確認してください。

サンプル

- **altova:substring-pos**('Altova', 'to') 3 を返します。

- **altova:substring-pos**('Altova', 'tov') 3 を返します。
- **altova:substring-pos**('Altova', 'tv') returns 0 を返します。
- **altova:substring-pos**('AltovaAltova', 'to') 3 を返します。

altova:substring-pos(StringToCheck *as xs:string*, **StringToFind** *as xs:string*, **Integer** *as xs:integer*) を**xs:integer** とする XP3 XQ3
StringToCheck 内でのStringToFind の最初の発生の文字位置を返します。Integer 引数により与えられた文字位置からStringToFind の検索が開始されます。この位置の前の文字サブ文字列は検索されません。しかし、返された整数は、全体文字列 StringToCheck の検索された文字列の位置です。この署名は、StringToCheck 内で複数回発生する第 2 または後の発生を検索する際に役に立ちます。StringToFind がStringToCheck 内で発生しない場合、整数 0 が返されます。

サンプル

- **altova:substring-pos**('Altova', 'to', 1) 3 を返します。
- **altova:substring-pos**('Altova', 'to', 3) 3 を返します。
- **altova:substring-pos**('Altova', 'to', 4) 0 を返します。
- **altova:substring-pos**('Altova-Altova', 'to', 0) 3 を返します。
- **altova:substring-pos**('Altova-Altova', 'to', 4) 10 を返します。

trim-string [altova:]

altova:trim-string(InputString *as xs:string*) を**xs:string** とする XP3 XQ3
この関数はxs:string 引数を必要とし、先頭または後続の空白を削除し、トリミングされたxs:string を返します。

サンプル

- **altova:trim-string**(" Hello World ")) は"Hello World" を返します。
- **altova:trim-string**("Hello World ")) は"Hello World" を返します。
- **altova:trim-string**(" Hello World")) は"Hello World" を返します。
- **altova:trim-string**("Hello World")) は"Hello World" を返します。
- **altova:trim-string**("Hello World"))は "Hello World" を返します。

trim-string-left [altova:]

altova:trim-string-left(InputString *as xs:string*) を**xs:string** とする XP3 XQ3
この関数はxs:string 引数を必要とし、先頭または後続の空白を削除し、トリミングされたxs:string を返します。

サンプル

- **altova:trim-string-left**(" Hello World "))は "Hello World " を返します。
- **altova:trim-string-left**("Hello World ")) は"Hello World " を返します。
- **altova:trim-string-left**(" Hello World")) は"Hello World" を返します。
- **altova:trim-string-left**("Hello World")) は"Hello World" を返します。
- **altova:trim-string-left**("Hello World")) は"Hello World" を返します。

trim-string-right [altova:]

altova:trim-string-right(InputString *as xs:string*) を**xs:string** とする XP3 XQ3
この関数はxs:string 引数を必要とし、先頭または後続の空白を削除し、トリミングされたxs:string を返します。

サンプル

- **altova:trim-string-right**(" Hello World "))は" Hello World" を返します。
- **altova:trim-string-right**("Hello World "))は"Hello World" を返します。
- **altova:trim-string-right**(" Hello World")) は" Hello World" を返します。
- **altova:trim-string-right**("Hello World")) は"Hello World" を返します。
- **altova:trim-string-right**("Hello World")) は"Hello World" を返します。

XPath/ XQuery 関数 :その他

以下の一般使用のためのXPath/XQuery 拡張関数は、MapForce の現在のバージョンによりサポートされています。また、次で使用することができます : (i) XSLT コンテキスト内のXPath 式、または (i) XQuery ドキュメント内のXQuery 式。

関数の名前指定と言語の適用性

Altova 拡張関数はXPath/XQuery 式で使用することができ、XPath、XQuery、およびXSLT 関数の標準ライブラリで使用可能な機能に更なる機能性を与えます。Altova 拡張関数は**Altova 拡張関数名前空間、http://www.altova.com/xslt-extensions** に収められており、**altova:** プレフィックスが、このセクションでは使用されます。製品の今後のバージョンが拡張機能への継続的サポート、または個別の関数の振る舞いは変更する可能性があることにご注意してください。Altova 拡張機能へのサポートに関しては、今後のリリースのドキュメントを参照してください。

XPath 関数 (XSLT 内のXPath 式で使用):	**XP1** **XP2** **XP3**
XSLT 関数 (XSLT 内のXPath 式で使用):	**XSLT1** **XSLT2** **XSLT3**
XQuery 関数 (XQuery 内のXQuery 式で使用):	**XQ1** **XQ3**

URI 関数

get-temp-folder [altova:]

altova:get-temp-folder() をxs:string とする **XP2** **XQ1** **XP3** **XQ3**

この関数は引数を必要としません。この関数は現在のユーザーの一時的なフォルダーへのパスを返します。

サンプル

- **altova:get-temp-folder**() は、マシーン上で xs:string としてC:\Users\<UserName>\AppData\Local\Temp\ と類似したパスを返します。

[Top]

11.1.2.2 その他の拡張関数

Java や C# などのプログラミング言語には、XPath 2.0 / XQuery 関数、または XSLT 2.0 関数として利用できない関数がいくつかあります。そのような関数の良い例として、Java で利用することのできる `sin()` や `cos()` という数学関数があります。XSLT スタイルシートや XQuery のクエリにてこれらの関数が利用できるのであれば、スタイルシートやクエリの適用範囲を大幅に拡張することができ、スタイルシート作成タスクの負担が大幅に軽減されます。Altova 製品で使用されている Altova エンジン (XSLT 1.0、XSLT 2.0、XQuery 1.0) では、Java や .NET および MSXSL scripts for XSLT における拡張関数の使用がサポートされます。このセクションでは、拡張機能および XSLT スタイルシート内で MSXSL スクリプト、および XQuery ドキュメントを使用する方法について記述します。使用できる拡張関数は以下のように構成されます :

- Java 拡張関数
- .NET 拡張関数
- XSLT に対する MSXSL スクリプト

記述の中では特に、(i) 関連するライブラリ内の関数がどのように呼ばれるか、(ii) 関数呼び出しを行う際に入力として使用される引数を変換するのにどのようなルールが適用され、return により値が返される際にどのような変換ルールが適用されるのか (XSLT/XQuery データオブジェクトに対する関数の結果) について説明されます。

必要条件
拡張関数のサポートを有効にするには、XSLT 変換や XQuery の実行を行うコンピューターに Java Runtime Environment (Java 関数にアクセスする場合) ならびに .NET Framework 2.0 以上 (.NET 関数にアクセスする場合) がインストールされている、またはアクセスできる環境が整っている必要があります。

Java 拡張関数

Java 拡張関数は、XPath または XQuery 条件式にて使用することができるほか、Java のコンストラクターを呼び出したり、Java の (静的または*インスタンス*) メソッドを呼び出すことができます。

Java *クラス*のフィールドは、引数を持たないメソッドとして扱われます。フィールドは静的または*インスタンス*として存在することができます。フィールドへのアクセス方法については、静的と*インスタンス*の両方について、以下のサブセクションにて記述されます。

このセクションは以下のサブセクションにより構成されます :

- Java : コンストラクター
- Java : 静的メソッドと静的フィールド
- Java : インスタンスメソッドとインスタンスフィールド
- データ型 : XPath/XQuery から Java へ
- データ型 : Java から XPath/XQuery へ

拡張関数のフォーム
XPath/XQuery 条件式における拡張関数では、`prefix:fname()` の形式を取る必要があります。

- prefix: 部により拡張関数がJava 関数として認識されます。java: から始まるURI のスコープ内の名前空間宣言に拡張関数を関連付けることでJava 関数であるという認識が行われます。名前空間の宣言により、例えばxmlns:myns="java:java.lang.Math" というJava クラスが特定されます。名前空間の宣言は、(コロン無しの) xmlns:myns="java" という形式で、Java クラスの識別子を拡張関数にあるfname() 部の左型に配置することでも行うことができます。
- fname() 部により、呼び出されている Java メソッドが識別され、メソッドの引数が提供されます (以下の例を参照ください)。prefix: 部にて識別された名前空間 URI がJava クラスを識別できなかった場合、Java クラスの識別はクラスの前にくるfname() 部にて行うことになり、ピリオドによりクラスから分離されることになります (以下にある2番目のXSLT サンプルを参照)。

メモ: 呼び出されるクラスはコンピューターのクラスパス上にある必要があります。

XSLT サンプル

以下に静的メソッドを呼び出す2つのサンプルを示します。最初のサンプルでは、クラス名 (java.lang.Math) が名前空間 URI に加えられており、fname() へ加えることはできません。2番目のサンプルでは、prefix: 部にjava: が与えられており、fname() 部にてクラスとメソッドが識別されます。

```
<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)" />
```

拡張関数内にあるメソッド名 (上の例ではcos()) は、名前付きJava クラス (上の例ではjava.lang.Math) のpublic な静的メソッドの名前に一致する必要があります。

XQuery サンプル

以下にXSLT のサンプルに似たXQuery のサンプルを示します:

```
<cosine xmlns:jMath="java:java.lang.Math">
  {jMath:cos(3.14)}
</cosine>
```

ユーザー定義されたJava クラス

独自のJava クラスやメソッドを作成した場合、(i) JAR ファイル (またはclass ファイル) を介してこれらクラスファイルへアクセスしているか、(ii) これら (JAR またはclass) ファイルが、カレントディレクトリ (XSLT やXQuery ドキュメントが存在するディレクトリ) に配置されているかにより、これらクラスの呼び出し方法が変わってきます。これらファイルの特定方法については、ユーザー定義クラスファイルならびにユーザー定義 JAR ファイルを参照ください。カレントディレクトリには無いクラスファイルやJAR ファイルへのパスは指定しなければならないことにご注意してください。

ユーザー定義のクラスファイル

アクセスがクラスファイルを介したものである場合、4つのケースが考えられます:

- クラスファイルはパッケージである。XSLT または XQuery ファイルが Java パッケージと同じ場所に収められている。 (下のサンプルを参照)
- クラスファイルはパッケージでは無い。XSLT または XQuery ファイルが Java パッケージと同じ場所に収められている。(下のサンプルを参照)
- クラスファイルはパッケージである。XSLT または XQuery ファイルがランダムな場所に収められている。 (下のサンプルを参照)
- クラスファイルはパッケージである。XSLT または XQuery ファイルがランダムな場所に収められている。 (下のサンプルを参照)

クラスファイルがパッケージではなく、XSLT または XQuery ドキュメントと同じ場所に収められているケースを考えてみましょう。この場合、フォルダー内の全クラスを発見することができるため、ファイルの場所を指定する必要はありません。クラスの識別を行う構文は以下のようになります：

```
java:classname
```

ここで、

`java:` によりユーザー定義の Java 関数が呼ばれていることが示されます (デフォルトでカレントディレクトリにある Java クラスがロードされます)。

`classname` は目的となるメソッドのクラスが含まれているクラスの名前です。

クラスが名前空間 URI にて識別され、名前空間がメソッド呼び出しにて使用されます。

クラスファイルがパッケージで、XSLT または XQuery ファイルが Java パッケージと同じ場所に収められている

以下の例では、`com.altova.extfunc` パッケージにある `Car` クラスの `getVehicleType()` メソッドが呼び出されています。com.altova.extfunc パッケージは `JavaProject` という名前のフォルダーに置かれており、XSLT ファイルも同じフォルダーに配置されています。

```
<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>
```

クラスファイルがパッケージではなく、XSLT または XQuery ファイルが Java パッケージと同じ場所に収められている

以下の例では、`com.altova.extfunc` パッケージにある Car クラスの `getVehicleType()` メソッドが呼び出されています。Car クラスファイルは `JavaProject/com/altova/extfunc` 以下に配置されています。XSLT ファイルも `JavaProject/com/altova/extfunc` フォルダーに配置されています。

```
<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>
```

クラスファイルがパッケージで、XSLT または XQuery ファイルがランダムな場所に収められている

以下の例では、com.altova.extfunc パッケージにある Car クラスの getVehicleColor() メソッドが呼び出されています。com.altova.extfunc パッケージは JavaProject という名前のフォルダーに置かれており、XSLT ファイルが任意の場所に配置されています。この場合、以下のような構文でパッケージの場所をクエリ文字列として URI 内にて指定する必要があります：

```
java:classname[?path=uri-of-package]
```

ここで、

java: によりユーザー定義の Java 関数が呼ばれていることを表します。
uri-of-package は Java パッケージの URI です。
classname は目的のメソッドが含まれているクラス名です。

クラスは名前空間 URI により特定され、名前空間がメソッド呼び出しのプレフィックスで使用されます。以下の例ではカレントディレクトリ以外にあるクラスファイルへのアクセスを行うことができます。

```
<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>
```

クラスファイルがパッケージではなく、XSLT または XQuery ファイルがランダムな場所に収められている

以下の例では、com.altova.extfunc パッケージにある Car クラスの getCarColor() メソッドが呼び出されています。com.altova.extfunc パッケージは JavaProject という名前のフォルダーに置かれており、XSLT ファイルが任意の場所に配置されています。以下のような構文で、クラスファイルの場所をクエリ文字列として URI 内にて指定する必要があります:

```
java:classname[?path=uri-of-classfile]
```

ここで

`java:` により*ユーザー定義*の Java 関数が呼ばれていることを表します。
`uri-of-classfile` は Java パッケージの URI です。
`classname` は目的のメソッドが含まれている*クラス*名です。

*クラス*は名前空間 URI により特定され、名前空間はメソッド呼び出しの*プレフィックス*で使用されます。以下の例ではカレントディレクトリ以外にある*クラスファイル*へのアクセスを行うことができます。

```
<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 xs xsl"/>

<xsl:template match="/">
  <xsl:variable name="myCar" select="car:new('red')" />
  <a><xsl:value-of select="car:getCarColor($myCar)"/></a>
</xsl:template>

</xsl:stylesheet>
```

メモ: パスが外部関数により与えられている場合、ClassLoader により*パス*が追加されます。

ユーザー定義の JAR ファイル

JAR ファイル経由でアクセスが行われた場合、以下の構文により JAR ファイルの URI を指定する必要があります：

```
xmlns:classNS="java:classname?path=jar:uri-of-jarfile!/"
```

*クラス*の識別を行う名前空間 URI の*プレフィックス*を使用してメソッドが呼び出されます：`classNS:method()`

上の例に対する説明は以下のとおりです:

`java:` 関数が呼び出されていることを表します。
`classname` が*ユーザー定義*された*クラス*の名前になります。
`?` は*クラス*名とパスを分離するために使用されます。
`path=jar:` により、JAR ファイルへのパスが与えられていることを示します。
`uri-of-jarfile` は JAR ファイルの URI となります。
`!/` は、終了を表すデリミタとなります。
`classNS:method()` により、メソッドの呼び出しが行われます。

その他にも、メソッド名とともに*クラス*名を与えることができます。構文の例を以下に示します：

```
            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()
```

以下にJAR ファイルを使ったJava 拡張関数を呼び出すXSLT サンプルを記します：

```
<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>
```

メモ： 拡張関数によりパスが与えられている場合、ClassLoader にパスが追加されます。

Java：コンストラクター

拡張関数を使用することでJava コンストラクターを呼び出すことができます。`new()` により全てのコンストラクターを呼び出すことができます。

Java コンストラクターの呼び出し結果を、默示的にXPath/XQuery データ型へ変換できる場合、Java 拡張関数によりXPath/XQuery データ型のシーケンスが返されます。Java コンストラクターの呼び出し結果がXPath/XQuery データ型へ変換できない場合、値を返すクラス名でラップしたJava オブジェクトがコンストラクターにより作成されます。例えば、`java.util.Date` クラスに対するコンストラクターが呼び出された場合（`java.util.Date.new()`）、`java.util.Date` を持ったオブジェクトが返されます。返されたオブジェクトのレキシカルフォーマットは、XPath データ型のレキシカルフォーマットにマッチしない場合もあり、目的のXPath データ型に対するレキシカルフォーマットへ値の変換を行い、その後目的のXPath データ型へ変換を行う必要があります。

コンストラクターにより作成されたJava オブジェクトにより2つのことが行えます：

- 変数への割り当てを行うことができます：
  `<xsl:variable name="currentdate" select="date:new()" xmlns:date="java:java.util.Date" />`
- 拡張関数への受け渡しを行うことができます（*インスタンスメソッドならびにインスタンスフィールド*を参照ください）：
  `<xsl:value-of select="date:toString(date:new())" xmlns:date="java:java.util.Date" />`

Java：静的メソッドと静的フィールド

静的メソッドは、Java 名ならびにメソッドの引数により直接呼び出すことができます。`E` や`PI` といった定数の静的フィールド（引数を持たないメソッド）は、引数を指定することなくアクセスすることができます。

XSLT の例

静的メソッドならびにフィールドを呼び出す例を以下に示します：

```
<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)" />
```

上の拡張関数は `prefix:fname()` という形式を使用していることに注意してください。3つの例にあるプレフィックスは全てjMath: となっており、このプレフィックスは、`java:java.lang.Math` という名前空間 URI に関連付けられています。名前空間 URI は java: で開始しなければなりません。上の例では、クラス名（`java.lang.Math`）を含むように拡張されています。拡張関数の fname() 部は（`java.lang.Math` のような）public クラスにマッチする必要があり、その後 public な静的メソッドが引数とともに続くか（例：`cos(3.14)`）、public な静的フィールドが続きます（例：PI()）。

上の例では、クラス名が名前空間 URI に含まれています。クラス名が名前空間 URI に含まれていない場合、以下の例にあるように、拡張関数の `fname()` 部にて追加する必要があります：

```
<xsl:value-of xmlns:java="java:"
              select="java:java.lang.Math:cos(3.14)" />
```

XQuery の例

XQuery における似たようなサンプルを以下に示します：

```
<cosine xmlns:jMath="java:java.lang.Math">
  {jMath:cos(3.14)}
</cosine>
```

Java：インスタンスメソッドとインスタンスフィールド

メソッド呼び出しの第一引数としてパースされるJava オブジェクトが、インスタンスメソッドには与えられています。このようなJava オブジェクトは通常、拡張関数を使うことで作成される（例：コンストラクターの呼び出し）か、スタイルシートパラメーター変数により作成されます。以下にXSLT サンプルを示します：

```
<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>
```

上の例では、ノード `enrollment/@type` の値が以下のように作成されます：

1. `java.util.Date` クラスに対するオブジェクトがコンストラクター（`date:new()` コンストラクター）とともに作成されます。
2. `jlang.Object.getClass` メソッドの引数としてJava オブジェクトがパースされます。
3. `getClass` メソッドにより得られたオブジェクトが `jlang.Object.toString` メソッドの引数としてパースされます。

結果 (@type の値) は java.util.Date を持った文字列となります。

*インスタンスフィールド*は、引数として*インスタンスフィールド*へ渡されるJava オブジェクトではないという点で、理論的には*インスタンスメソッド*と異なります。パラメーターや変数がその代わりに引数として渡されますが、パラメーター変数そのものにJava オブジェクトから返された値が含まれている場合もあります。例えば、CurrentDate パラメーターには java.util.Date *クラス*のコンストラクターから返された値が含まれます。この値は引数として、date:toString *インスタンスメソッド*へ渡され、/enrollment/@date の値として使用されます。

データ型 : XPath/ XQuery から Java へ

XPath/XQuery 条件式内部からJava 関数が呼び出された場合、複数ある同名のJava *クラス*のうち、*どのクラス*が呼び出されたのか決定するのに、関数へ渡される引数のデータ型が重要になります。

Java では、以下のルールが適用されます :

- 同名のJava メソッドが2つ以上あり、それぞれが違う数の引数を受け取る場合、呼び出しに使用されている引数の数に一番マッチするメソッドが選択されます。
- XPath/XQuery の文字列、数値、boolean データ型は、黙示的に対応するJava データ型へ変換されます (以下のリストを参照)。与えられたXPath/XQuery 型が2つ以上のJava 型へ変換できる場合 (例 : xs:integer)、選択されたメソッドにて宣言されているJava 型が使用されます。例えば、呼び出されたJava メソッドが fx(decimal) で、与えられたXPath/XQuery データ型が xs:integer の場合、xs:integer がJava の decimal データ型へ変換されます。

以下のテーブルに、XPath/XQuery の文字列、数値、boolean 型からJava データ型への黙示的な変換リストを示します。

xs:string	java.lang.String
xs:boolean	boolean (*プリミティブ型*), java.lang.Boolean
xs:integer	int, long, short, byte, float, double, ならびに java.lang.Integer のようなこれらのラッパー*クラス*
xs:float	float (*プリミティブ型*),java.lang.Float, double (*プリミティブ型*)
xs:double	double (*プリミティブ型*), java.lang.Double
xs:decimal	float (*プリミティブ型*), java.lang.Float, double(*プリミティブ型*), java.lang.Double

上のリストにあるXML *スキーマデータ型* (ならびにXPath やXQuery で使用されているデータ型) のサブタイプも、対応する祖先のサブタイプとしてJava のデータ型へ変換されます。

場合によっては、与えられた情報から正しいJava メソッドを選択することができない場合もあります。例えば、以下のような場合を考えてみましょう:

- 与えられた引数が 10 という値を持ったxs:untypedAtomic 型で、mymethod(float) メソッドへ渡されるのを意図している。

- しかし、そのクラスには別のデータ型を取る`mymethod(double)` というメソッドも存在する。
- メソッド名が同じで、与えられた型（`xs:untypedAtomic`）も`float` と`double` の両方に変換することができるため、`xs:untypedAtomic` が`float` ではなく`double` に変換される可能性もある。
- 結果として、意図したメソッドは選択されず、予期しない動作結果が招く可能性がある。この問題を回避するには、意図したメソッドを使用するユーザー定義のメソッドを別の名前で新たに作成する必要があります。

上のリストでカバーされていない型（例：`xs:date`）は変換されず、エラーとなります。しかし場合によっては、Java コンストラクターを使用して、目的のJava データ型を作成することが可能だということも留意してください。

データ型：Java から XPath/ XQuery へ

Java メソッドにより値が返され、値のデータ型が文字列、数値、またはboolean 型の場合、対応するXPath/XQuery 型への変換が行われます。例えば、Java の`java.lang.Boolean` や`boolean` データ型は`xsd:boolean` へ変換されます。

関数から返された一次元配列は、シーケンス（sequence）に展開されます。2次元以上の配列は変換されることが無いため、ラップして使用するべきでしょう。

Java オブジェクトや文字列、数値、boolean 以外のデータ型がラップされて返された場合、最初に（例えば`toString` という）Java メソッドを使用してJava オブジェクトを文字列へ変換することで、目的のXPath/XQuery 型への変換を行います。XPath/XQuery では、文字列を目的となる型のレキシカルフォーマットへ変換し、目的の型への変換を（例えば`cast as` 式を使用することで）行うことができます。

.NET 拡張関数

.NET プラットフォームにて作業を行なっている場合、.NET 言語（例えばC#）で記述された拡張関数を使用することができます。.NET 拡張関数はXPath やXQuery 条件式内部から使用することができ、.NET クラス内部にあるコンストラクターや（static またはインスタンス変数）プロパティを呼び出すことができます。

`get_PropertyName()` 構文を使用することにより .NET クラスのプロパティを呼び出すことができます。

このセクションは、以下のサブセクションにより構成されています：

- .NET コンストラクター
- .NET 静的メソッドならびに静的フィールド
- .NET：インスタンスメソッドとインスタンスフィールド
- データ型：XPath/XQuery から .NET へ
- データ型：.NET からXPath/XQuery へ

拡張関数のフォーム

XPath/XQuery 条件式にある拡張関数は、`prefix:fname()` の形式を取る必要があります。

- `prefix:` 部は、呼び出されている .NET クラスを特定するURI となります。
- `fname()` 部により、.NET クラス内にあるコンストラクター、プロパティ、または（静的またはインスタンス）メソッドが特定され、必要な場合は引数が与えられます。
- URI は`clitype:` で開始する必要があり、これにより関数が .NET 拡張関数であることが認識されます。
- 拡張関数の `prefix:fname()` 形式は、システムクラスならびにロードされたアセンブリとともに使用することもできます。しかしクラスをロードする必要がある場合、必要な情報が含まれるパラメーターが必要になります。

パラメーター

アセンブリをロードするには以下のパラメーターを使用してください：

asm	ロードするアセンブリの名前。
ver	バージョン番号（ピリオドにより分離された最大 4 桁の整数）。
sn	アセンブリ厳密名のキートークン（16 新数の数値）。
from	ロードするアセンブリ (DLL) の場所を特定する URI。URI が相対パスの場合、XSLT や XQuery ドキュメントに対して相対的になります。このパラメーターが指定された場合、その他のパラメーターが無視されます。
partialname	アセンブリ名の一部。Assembly.LoadWith.PartialName() へ渡され、アセンブリへのロードが試みられます。 partialname が指定された場合、その他のパラメーターが無視されます。
loc	例えば en-US というロケール。デフォルトは neutral です。

アセンブリが DLL からロードされる場合、 from パラメータが使用して、sn パラメーターは使用しないでください。アセンブリがグローバルアセンブリキャッシュ (GAC) からロードされる場合、sn パラメータを使用して from パラメーターは使用しないでください。

最初のパラメーターの前に疑問符 (?) を挿入し、パラメーター同士はセミコロンで分離する必要があります。パラメーター名へ値を受け渡すには、統合符号 (=) を使用します（以下の例を参照ください）。

名前空間宣言の例

XSLT において、システムクラス System.Environment を特定する名前空間宣言の例を以下に示します：

```
xmlns:myns="clitype:System.Environment"
```

XSLT において、ロードするクラスを Trade.Forward.Scrip として特定する名前空間宣言の例を以下に示します：

```
xmlns:myns="clitype:Trade.Forward.Scrip?asm=forward;version=10.6.2.1"
```

XQuery において、システムクラス MyManagedDLL.testClass を特定する名前空間宣言の例を以下に示します。2 つのケースが考えられます：

1. アセンブリが GAC からロードされた場合：

```
declare namespace cs="clitype:MyManagedDLL.testClass?asm=MyManagedDLL;
        ver=1.2.3.4;loc=neutral;sn=b9f091b72dccfba8";
```

2. アセンブリが DLL からロードされた場合（完全参照と一部の参照）：

```
        declare namespace cs="clitype:MyManagedDLL.testClass?
from=file:///C:/Altova
        Projects/extFunctions/MyManagedDLL.dll;

    declare namespace cs="clitype:MyManagedDLL.testClass?
    from=MyManagedDLL.dll;
```

XSLT の例

システムクラス System.Math 内の関数を呼び出すための完全な XSLT の例を以下に示します：

```
<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>
```

math 要素にある名前空間宣言により、math: プレフィックスと clitype:System.Math URI が関連付けられます。URI の最初にある clitype: により、それ以降の記述がシステムクラスまたはロードされたクラスを特定するものであることが示されます。XPath 条件式にある math: プレフィックスにより、拡張関数が URI（そしてクラス）System.Math に関連付けられます。拡張関数により、System.Math クラス内のメソッドが特定され、必要な場所に引数が与えられます。

XQuery の例

上の XSLT に対する例と同様の XQuery 例を以下に示します：

```
<math xmlns:math="clitype:System.Math">
  {math:Sqrt(9)}
</math>
```

上の XSLT と同様に、名前空間宣言により .NET クラス（この場合はシステムクラス）が特定されます。XQuery 式により呼び出されるメソッドが特定され、引数が与えられます。

.NET コンストラクター

拡張関数を使用することで、.NET コンストラクターを呼び出すことができます。new() により全てのコンストラクターを呼び出すことができます。クラス内に 2 つ以上のコンストラクターがある場合、与えられた引数の数が最もマッチするコンストラクターが選択されます。与えられた引数に対してマッチするコンストラクターが見つからない場合、"No constructor found'" エラーが返されます。

XPath/XQuery データ型を返すコンストラクター

.NET コンストラクター呼び出しの結果が XPath/XQuery データ型へ黙示的に変換することができる場合、.NET 拡張関数から XPath/XQuery データ型のシーケンスが返されます。

.NET オブジェクトを返すコンストラクター

.NET コンストラクター呼び出しの結果がXPath/XQuery データ型へ適切に変換できない場合、値を返すクラス名でラップした .NET オブジェクトがコンストラクターにより作成されます。例えば、System.DateTime クラスのコンストラクターが（System.DateTime.new() により）呼ばれた場合 System.DateTime 型を持ったオブジェクトが返されます。

返されたオブジェクトのレキシカルフォーマットは、目的のXPath データ型と違っている場合があります。その場合、返された値を：(i) 目的のXPath データ型のレキシカルフォーマットへ変換し、(i) 目的のXPath データ型へキャストする必要があります。

コンストラクターにより作成された .NET オブジェクトに対して3つのことを行うことができます：

- 変数内で使用することができます：
  ```
  <xsl:variable name="currentdate" select="date:new(2008,4,29)"
  xmlns:date="clitype:System.DateTime" />
  ```
- 拡張関数へ渡すことができます（*インスタンスメソッドとインスタンスフィールド*を参照ください）：
  ```
  <xsl:value-of select="date:ToString(date:new(2008,4,29))"
  xmlns:date="clitype:System.DateTime" />
  ```
- 文字列、数値、または boolean へ変換することができます：
  ```
  <xsl:value-of select="xs:integer(date:get_Month(date:new(2008,4,29)))"
  xmlns:date="clitype:System.DateTime" />
  ```

.NET：静的メソッドと静的フィールド

メソッド名と引数を与えることで、静的メソッドを直接呼び出すことができます。呼び出しに使用される名前は、クラス内にある public static メソッドと完全に一致する必要があります。関数の呼び出しに使用されたメソッド名と引数の数にマッチするものがクラス内に複数ある場合、与えられた引数が評価され、最もマッチするものが選択されます。マッチする結果が得られない場合、エラーが返されます。

メモ： .NET クラス内にあるフィールドは、引数を持たないメソッドとしてみなされます。プロパティは get_PropertyName() 構文により呼び出されます。

例

1つの引数とともにメソッド（system.Math.Sin(arg)）を呼び出す XSLT サンプルを以下に示します：

```
<xsl:value-of select="math:Sin(30)" xmlns:math="clitype:System.Math"/>
```

（引数なしのメソッドとしてみなされる）フィールド（system.Double.MaxValue()）を呼び出す XSLT サンプルを以下に示します：

```
<xsl:value-of select="double:MaxValue()" xmlns:double="clitype:System.Double"/>
```

（get_PropertyName() 構文を使って）プロパティ（system.String()）を呼び出す XSLT サンプルを以下に示します：

```
<xsl:value-of select="string:get_Length('my string')" xmlns:string="clitype:System.String"/>
```

1つの引数とともにメソッド（system.Math.Sin(arg)）を呼び出す XQuery サンプルを以下に示します：

```
<sin xmlns:math="clitype:System.Math">
  {math:Sin(30)}
</sin>
```

.NET：インスタンスメソッドとインスタンスフィールド

インスタンスメソッドには、メソッド呼び出しの第一引数として .NET オブジェクトが渡されます。通常この .NET オブジェクトは、拡張関数（例えばコンストラクター呼び出し）またはスタイルシートパラメーター変数により作成されます。以下にXSLTの例を示します：

```
<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>
```

上の例では、System.DateTime コンストラクター（`new(2008, 4, 29)`）が、System.DateTime 型の .NET オブジェクトの作成に使用されます。このオブジェクトは、最初に`releasedate` 変数の値として、次に`System.DateTime.ToString()` メソッドの引数として作成されます。`System.DateTime.ToString()` インスタンスメソッドは、`System.DateTime` コンストラクターの（new(2008, 4, 29)）における引数として2度呼び出されます。これらインスタンスにおいて`releasedate` 変数が .NET オブジェクトを取得するのに使用されます。

インスタンスメソッドとインスタンスフィールド

インスタンスメソッドとインスタンスフィールドの違いは理論的なものです。インスタンスメソッドでは .NET オブジェクトが直接引数に渡され、インスタンスフィールドではパラメーターや変数が（.NET オブジェクトそのものを含めることはできるものの）代わりに渡されます。例えば上の例では、`releasedate` 変数に .NET オブジェクトが含まれており、この変数が2番目のdate要素コンストラクターにて`ToString()` の引数として渡されます。そのため、最初のdate 要素にある`ToString()` インスタンスがインスタンスメソッドであるのに対し、2番目はインスタンスフィールドとしてみなされます。両方のインスタンスで求められる結果は等価です。

データ型：XPath/ XQuery から .NET へ

.NET 拡張関数がXPath/XQuery 条件式内部で使用された場合、複数ある .NET メソッドのうち、どれが呼び出されたのか決定するのに関数の引数に使用されるデータ型が重要になります。

.NET では、以下のルールが適用されます：

- クラス内に同名のメソッドが2つ以上ある場合、呼び出しに使用された引数の数がマッチするメソッドだが、呼び出

される関数の候補に絞められます。

- XPath/XQuery の文字列、数値、boolean データ型は黙示的に対応する .NET データ型へ変換されます（以下のリストを参照）。与えられた XPath/XQuery 型が 2 つ以上の .NET 型へ変換できる場合（例：`xs:integer`）、選択されたメソッドにて宣言されている .NET 型が使用されます。例えば、呼び出された .NET メソッドが fx(fouble) で、与えられた XPath/XQuery データ型が xs:integer の場合、`xs:integer` が .NET の `double` データ型へ変換されます。

以下のテーブルに、XPath/XQuery の文字列、数値、boolean 型から .NET データ型へ行われる黙示的な変換リストを示します。

`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`

上のリストにある XML スキーマ型（ならびに XPath や XQuery で使用されているデータ型）のサブタイプも、対応する祖先のサブタイプとして .NET のデータ型へ変換されます。

場合によっては、与えられた情報から正しい .NET メソッドを選択することができない場合もあります。例えば、以下のような場合を考えてみましょう：

- 与えられた引数が 10 ある `xs:untypedAtomic` 値で、`mymethod(float)` メソッドへ渡されるのを意図している。
- しかし、そのクラスには別のデータ型をとる `mymethod(double)` というメソッドも存在する。
- メソッド名が同じで、与えられた型（`xs:untypedAtomic`）も `float` と `double` の両方に変換することができるため、`xs:untypedAtomic` が `float` ではなく `double` に変換される可能性もある。
- 結果として、意図したメソッドは選択されず、予期しない動作結果が招く可能性がある。この問題を回避するには、意図したメソッドを使用するユーザー定義のメソッドを別の名前で新たに作成する必要があります。

上のリストでカバーされていない型（例：`xs:date`）は変換されずエラーとなります。

データ型：.NET から XPath/ XQuery へ

.NET メソッドにより値が返される際に値のデータ型が文字列、数値、または boolean 型の場合、対応する XPath/ XQuery 型への変換が行われます。例えば、.NET の `decimal` データ型は `xsd:decimal` へ変換されます。

.NET オブジェクトや文字列、数値、boolean 以外のデータ型が返された場合、最初に（例えば `System.DateTime.ToString()` といった）.NET メソッドを使用して .NET オブジェクトを文字列へ変換します。XPath/XQuery では、文字列を目的となる型のレキシカルフォーマットへ変換し、目的の型への変換を（例えば `cast as` 式を使用することで）行うことができます。

XSLT に対する MSXSL スクリプト

<msxsl:script> 要素にはユーザー定義の関数や変数が含まれており、XSLT スタイルシート内の XPath 条件式内部から呼び出しを行うことができます。<msxsl:script> はトップレベル要素で、<xsl:stylesheet> または <xsl:transform> の子要素である必要があります。

<msxsl:script> 要素は urn:schemas-microsoft-com:xslt 名前空間内に存在する必要があります（以下を参照ください）。

スクリプト言語と名前空間

ブロック内で使用されるスクリプト言語は <msxsl:script> 要素の language 属性にて指定され、XPath 条件式における関数の呼び出しに対して使用される名前空間は implements-prefix 属性により特定されます（以下を参照）。

```
<msxsl:script language="scripting-language" implements-prefix="user-namespace-
prefix">

   function-1 or variable-1
   ...
   function-n or variable-n

</msxsl:script>
```

<msxsl:script> 要素は Windows Scripting Runtime を使ったやりとりを行うため、お使いのコンピューターにインストールされた言語だけしか <msxsl:script> 要素では使用することができません。**MSXSL スクリプトを使用するには .NET Framework 2.0 以上のプラットフォームをインストールする必要があります。結果として** <msxsl:script> 言語から .NET スクリプト言語を使用することができます。

HTML の <script> 要素における language 属性と同じ値が language 属性では受理されます。language 属性が指定されていない場合、Microsoft JScript がデフォルトとして想定されます。

implements-prefix 属性には名前空間スコープ内で宣言されたプレフィックスが与えられます。通常この名前空間は関数ライブラリのために予約されたユーザーの名前空間となります。<msxsl:script> 要素内で定義された全ての関数ならびに変数は、implements-prefix 属性にて指定されたプレフィックスで特定される名前空間に収められます。XPath 条件式内部から関数が呼ばれる場合、完全修飾関数名が同じ名前空間内に関数として定義されていなければなりません。

例

<msxsl:script> 要素内で定義された関数を使用する XSLT スタイルシートの例を以下に示します：

```
<?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>TotalRetailPrice =
      $<xsl:value-of select="user:AddMargin(50)"/>
     </b>
     <br/>
     <b>TotalWholesale Price =
      $<xsl:value-of select="50"/>
     </b>
    </p>
   </body>
  </html>
 </xsl:template>
</xsl:stylesheet>
```

データ型

スクリプトブロックとのやりとりに使用されるパラメーターの値はXPath データ型に限定されます。スクリプトブロック内にある関数にてやりとりされるデータや変数に、この制限はありません。

アセンブリ

msxsl:assembly 要素を使用することで、アセンブリをスクリプト内部へインポートすることができます。アセンブリは名前やURI により特定されます。アセンブリのインポートは、コンパイル時に行われます。以下に **msxsl:assembly** 要素の簡単な使用例を示します：

```
<msxsl:script>
       <msxsl:assembly name="myAssembly.assemblyName" />
       <msxsl:assembly href="pathToAssembly" />

       ...

</msxsl:script>
```

アセンブリ名は、以下のような完全な名前でも：

```
"system.Math, Version=3.1.4500.1 Culture=neutral
PublicKeyToken=a46b3f648229c514"
```

"myAssembly.Draw" のような短い名前でも指定できます。

名前空間

msxsl:using 要素により名前空間の宣言を行うことができます。これにより、スクリプト内において名前空間無しでアセンブリクラスを使用することができ、タイピングの手間を軽減することができます。以下に msxsl:using 要素の簡単な使用例を示します：

```
<msxsl:script>
       <msxsl:using namespace="myAssemblyNS.NamespaceName" />

       ...

</msxsl:script>
```

namespace 属性の値は名前空間の名前となります。

11.2 技術データ

このセクションは、ソフトウェアの技術面に関する役に立つ背景情報を含んでいます。以下のように整理されています：

- OS とメモリ要件
- Altova XML バリデーター
- Altova XSLT とXQuery エンジン
- Unicode のサポート
- インターネットの使用

11.2.1 OS とメモリ要件

オペレーティングシステム
Altova ソフトウェア アプリケーションは、以下のプラットフォームでご使用いただけます：

- Windows XP/Vista, Windows 7/8/10
- Windows Server 2003/2008/2012/2016

メモリ
ソフトウェアがC++ で書かれているため、Java Runtime Environment をダウンロードする必要はなく、Java ベースのアプリケーションに比べ、通常少ないメモリを必要とします。しかしながら、各ドキュメントは完全に解析するため、また、ビューと編集の速度を向上するためにメモリにダウンロードされます。メモリの要件は、ドキュメントのサイズを増やします。

メモリ要件は、制限のない「元に戻す」履歴により影響を受けます。大きなドキュメントの大きなセクションの切り取り、貼り付け操作を繰り返し行うと、使用できるメモリがすぐに消費されます。

11.2.2 Altova XML バリデーター

XML ドキュメントを開くと、アプリケーションは、内蔵のXML バリデーターを使用して、指定されている場合、スキーマに対して整形式をチェックし、ツリーとインフォセットを作成します。XML バリデーターは、ドキュメントを編集する際にインテリジェントな編集ヘルプを提供し、発生する検証エラーを表示するために使用されます。

内蔵のXML バリデーターは、W3C のXML スキーマ 1.0 と 1.1 仕様の最終勧告を実装しています。 .New developments recommended by the W3C XML スキーマ作業グループにより勧告される新しい項目は、XML バリデーターに継続的に組み込まれるため、Altova 製品は最高水準の開発環境を届けることができます。

11.2.3 Altova XSLT と XQuery エンジン

Altova 製品 は、Altova XSLT 1.0、 2.0、 および 3.0 エンジンとAltova XQuery 1.0 と 3.1 エンジンを使用しています。各エンジンのためのドキュメントと実装に固有の振る舞いに関しては、製品で使用されるエンジンの各ドキュメントの付属書 (エンジン情報) で確認することができます。

メモ： Altova MapForce は、XSLT 1.0、2.0 および XQuery 1.0 エンジンを使用したコードを生成します。

11.2.4 Unicode のサポート

Altova XML 製品は、Unicode を完全にサポートします。XML ドキュメントを編集するには、ドキュメント内で使用されているUnicode 文字をサポートするフォントが必要です。

フォントの多くは、Unicode 範囲全体の特定のサブセットを含む場合があり、このため、通常は対応する表記システムをターゲットとします。テキストの一部が、文字化けして表示された場合、理由としては、選択されたフォントが必要とする字形を含まない場合があげられます。ですから、特に、異なる言語、または、異なる言語システムのXML ドキュメントを編集する場合、範囲全体をカバーするフォントを使用することが役にたちます。典型的なUnicode フォントは、Windows PC の Arial Unicode MS で確認することができます。

アプリケーションフォルダーの /Examples フォルダー内で、異なる言語システムで表記された次の文章を含むUnicodeUTF-8.htmlというXHTML ファイルを確認してください:

- *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。)

XHTML ファイルを開くと、Unicode の可能性を確認することができ、使用中のPC の使用することのできるフォントによりサポートされている表記システムが表示されます。

11.2.5 インターネットの使用

Altova アプリケーションは、次の状況でインターネット接続を開始します:

- 登録ダイアログ (**「ヘルプ| ソフトウェアのライセンス認証」**)内の「評価キーコードをリクエスト」をクリックした場合、登録ダイアログボックス内の3つのフィールドが通常のhttp (ポート 80)接続を使用し、サーバーに転送され、無料の評価キーが顧客に通常のSMTP 電子メールを使用して送り返されます。
- Altova 製品の一部では、インターネットからファイルを開くことができます (**「ファイル| 開く| URL に切り替える」**)。この場合、ドキュメントは、次のプロトコルメソッドと接続の1つを使用して取得されます:HTTP (通常、ポート 80)、FTP (通常、ポート 20/21)、HTTPS (通常、ポート 443)。HTTP サーバーをポート 8080 で作動することもできます (URL ダイアログ内で、サーバー名とコロンの後にポートを指定します)。
- XML スキーマ、または、DTD を参照するXML ドキュメント、と、URL により指定されているドキュメントを開くと、参照されているスキーマドキュメントは、HTTP 接続 (ポート 80)またはURL により指定されている他のプロトコール (上のポイント 2参照)により抽出されます。XML ファイルが検証されている場合、スキーマドキュメントも抽出されます (オプションダイアログのファイルタブ内の (**「ツール| オプション」**))。アプリケーションに命令している場合、ドキュメントが開かれると検証が自動的に行われる場合もあります。
- WSDL とSOAP を使用するAltova アプリケーションでは、Web サービスを使用する接続は、WSDL ドキュメントにより定義されています。
- XMLSpy 内で、**「電子メールで送信」** コマンドを使用する場合、(**「ファイル| 電子メールで送信」**) 現在選択されている範囲、または、ファイルは、ユーザーのマシンにインストールされているMAPI コンプライアント電子メールプログラムにより送信されます。
- ソフトウェアの荒いセンス認証 とLiveUpdate の一部として、Altova ソフトウェア使用許諾書内で更に詳しい説明を確認することができます。

11.3 ライセンス情報

ライセンス情報

- ソフトウェアの配布 に関する情報
- ソフトウェアのアクティベーションとライセンスの計測 に関する情報
- このソフトウェアに関連する知的所有権 に関する情報
- このソフトウェア製品の使用に関するエンドユーザー使用許諾契約書

本製品を使用する前に、上記の情報をよくお読みください。ソフトウェアのインストール時に上記のすべての条件に同意したとみなされ、お客様は上記の条件に拘束されることを同意したとみなされます。

11.3.1 電子的なソフトウェアの配布

この製品は電子的なソフトウェアの配布により利用することが可能で、この配布方法により、以下のユニークなメリットがあります：

- 購入を決定する前に、無料でソフトウェアを試用することができます。
- ソフトウェアの購入を決定した際には、Altova Web サイトにて注文を行います。すぐにライセンス登録された製品の使用を開始することができます。
- オンラインにて注文を行うと、常に最新のソフトウェアをご利用いただけます。
- 製品パッケージには包括的なヘルプシステムが画面上に表示されます。最新バージョンのユーザーマニュアルは www.altova.com 上にあり、(i) HTML フォーマットによる閲覧、ならびに (ii) PDF フォーマットのダウンロードと印刷に対応しております。

30日間の評価期間
この製品をダウンロードした後は、最大で30日の間無料で製品の評価を行うことができます。20日間を超えた頃から、製品がライセンス登録されていないことがソフトウェアにより表示されます。このメッセージはアプリケーションが起動されるたびに表示され、30日間を超えてプログラムを使用するには、キーコードとして送られるAltova ソフトウェア使用許諾契約書 を購入し、ソフトウェアアクティベーションダイアログにて入力し、製品をアンロックする必要があります。ライセンスの購入はAltova Web サイトのオンラインショップにてお求めいただけます。

組織内でソフトウェアの評価を行う
評価版のソフトウェアを組織内のネットワークにて配布したい場合、またはインターネットに接続されていないコンピューターにてソフトウェアを使用する場合、どのような状態でも改変されていないことを条件に、セットアッププログラムだけの配布を行うことが可能です。ソフトウェアインストーラーへアクセスした人は、例外なく30日間の評価ライセンスキーコードをリクエストして、試用期間が経過した後は、製品を使い続けるためにライセンスの購入を行う必要があります。

詳細については、このセクション最後にあるAltova ソフトウェアライセンス使用許諾契約書 を参照ください。

11.3.2 ソフトウェアのアクティベーションとライセンスの計測

Altova のソフトウェアアクティベーションの一部として、ソフトウェアにより内部ネットワークまたはインターネットへの接続を行い、インストール時、登録時、Altova により使用されるライセンスサーバーの更新やライセンスの正当性を検証することで、ソフトウェアの不正な使用を防ぎ、顧客サービスを向上するため、ライセンスに関する情報を送信することがあります。アクティベーションにより、オペレーティングシステムや IP アドレス、日付/時刻、ソフトウェアのバージョン、コンピュータ名などのライセンスに関する情報が、お使いのコンピューターとAltova ライセンスサーバー間にてやり取りされます。

お使いのAltova 製品にはライセンス計測モジュールが内蔵されており、エンドユーザー使用許諾契約書の意図しない違反を防ぎます。お使いの製品はシングルユーザーまたはマルチユーザーとしてインストールされており、ライセンス計測モジュールにより、ライセンスされている数を超えたユーザーが同時に製品を使用することが無いことが保証されます。

このライセンス計測技術により、ローカルエリア接続 (LAN) において、別々のコンピューター間で動作しているアプリケーションインスタンス間の通信が行われます。

シングルライセンス
ライセンス計測プロセスの一部としてアプリケーションが起動すると、ソフトウェアにより短いデータグラムがブロードキャストにより送信され、同一のネットワークセグメントにある他のコンピューターにてプログラムが動作していないかのチェックが行われます。応答が無い場合は、アプリケーションの他インスタンスから送信される信号に応えるため、ポートが開かれます。

マルチライセンス
同一のLAN 内にて2つ以上のアプリケーションインスタンスが使用された場合、スタートアップ時に、これらインスタンス間において通信が行われます。これらのインスタンス間にてキーコードのやりとりが行われ、購入された数のライセンスを超えてインスタンスが起動しないように保証することができます。このようなライセンス計測システムはUNIX やデータベース開発ツールにて広く使用されているもので、Altova ユーザーはリーズナブルな価格にて同時使用マルチユーザーライセンスを購入することができます。

弊社はアプリケーションのデザインも行っており、少数の小さなネットワークパケットを送信することで、ネットワークに対する負荷を最小限に抑えております。Altova により使用される2799番 TCP/IP ポートはIANA により公式登録されており(詳細は(IANA Web サイト(http://www.iana.org/) を参照ください)、弊社のライセンス計測モジュールも既にテストされたものです。

ファイヤーウォールを使用している場合、2799番ポートにてAltova 製品が動作しているコンピューター同士が通信しているのに気づかれるかも知れません。その他の手段によりライセンス使用許諾書の内容が守られることを保証できる限り、組織間の異なるグループにおいてこのようなトラフィックをブロックすることは勿論可能です。

お使いのコンピューターがオンラインの場合、ライセンス計測技術とは関係の無い数々の便利な機能を利用することができるようになります。

11.3.3 知的所有権

Altova ソフトウェアならびにAltova により認められたコピーは、Altova ならびにその供給者が知的所有権を有しています。ソフトウェアの構造や構成、コードはAltova ならびにその供給者にとって価値のある企業秘密かつ機密情報となります。このソフトウェアは米国の著作権法により保護されており、国際条項ならびにソフトウェアが使用されている国の法律が適用さ

れます。Altova はこのソフトウェアに関する特許、企業秘密、商標に関する所有権ならびにこれらに限定されないその他の知的財産権を有しており、Altova の所有権には、このソフトウェアならびにそれに付随する印刷物に抱合された画像、写真、アニメーション、映像、音声、音楽、テキスト、そして「アプレット」が含まれます。著作権侵害に関する届け出は、Altova Web Site に記されている Altova 著作権エージェントへお送りください。

Altova ソフトウェアには国際条項により保護されているサードパーティソフトウェアも含まれており、http://www.altova.com/legal_3rdparty.html にて詳細が記されている著作権法なども含め、しかしこれに限定されないかたちで知的所有権により保護されます。

その他全ての名前または商標は、対応する所有者が有するものです。

11.3.4 エンドユーザー使用許諾契約書

THIS IS A LEGAL DOCUMENT - - RETAIN FOR YOUR RECORDS

ALTOVA® END USER LICENSE AGREEMENT

Licensor:
Altova GmbH
Rudolfsplatz 13a/ 9
A- 1010 Wien
Austria

Important - Read Carefully. Notice to User:

This End User License Agreement ("Agreement") is a legal document between you and Altova GmbH ("Altova"). It is important that you read this document before using the Altova- provided software ("Software") and any accompanying documentation, including, without limitation printed materials, 'online' files, or electronic documentation ("Documentation"). By clicking the "I accept" and "Next" buttons below, or by installing, or otherwise using the Software, you agree to be bound by the terms of this Agreement as well as the Altova Privacy Policy ("Privacy Policy") including, without limitation, the warranty disclaimers, limitation of liability, data use and termination provisions below, whether or not you decide to purchase the Software. You agree that this agreement is enforceable like any written agreement negotiated and signed by you. If you do not agree, you are not licensed to use the Software, and you must destroy any downloaded copies of the Software in your possession or control. You may print a copy of this Agreement as part of the installation process at the time of acceptance. Alternatively, a copy of this Agreement may be found at http:/ / www.altova.com/ eula and a copy of the Privacy Policy may be found at http:/ / www.altova.com/ privacy.

1. SOFTWARE LICENSE

(a) License Grant.

(i) Upon your acceptance of this Agreement Altova grants you a non- exclusive, non- transferable (except as provided below), limited license, without the right to grant sublicenses, to install and use a copy of the Software on one compatible personal computer or workstation in the same local area network (LAN) up to the Permitted Number of computers. Subject to the limitations set forth in Section 1(c), you may install and use a copy of the Software on more than one of your compatible personal computers or workstations if you have purchased a Named- User license. Subject to the limitations set forth in Sections 1(d) and 1(e), users may use the software concurrently on a network. The Permitted Number of computers and/ or users and the type of license, e.g. Installed, Named- Users, and Concurrent- User, shall be determined and specified at such time as you elect to purchase the Software. Installed user licenses are intended to be fixed and not concurrent. In other words, you cannot uninstall the Software on one machine in order to reinstall that license to a different machine and then uninstall and reinstall back to the original machine. Installations should be static. Notwithstanding the foregoing, permanent uninstallations and redeployments are acceptable in limited

circumstances such as if an employee leaves the company or the machine is permanently decommissioned. During the evaluation period, hereinafter defined, only a single user may install and use the software on one (1) personal computer or workstation. If you have licensed the Software as part of a suite of Altova software products (collectively, the "Suite") and have not installed each product individually, then the Agreement governs your use of all of the software included in the Suite.

(ii) If you have licensed SchemaAgent, then the terms and conditions of this Agreement apply to your use of the SchemaAgent server software ("SchemaAgent Server") included therein, as applicable, and you are licensed to use SchemaAgent Server solely in connection with your use of Altova Software and solely for the purposes described in the accompanying documentation.

(iii) If you have licensed Software that enables users to generate source code, your license to install and use a copy of the Software as provided herein permits you to generate source code based on (i) Altova Library modules that are included in the Software (such generated code hereinafter referred to as the "Restricted Source Code") and (ii) schemas or mappings that you create or provide (such code as may be generated from your schema or mapping source materials hereinafter referred to as the "Unrestricted Source Code"). In addition to the rights granted herein, Altova grants you a non-exclusive, non-transferable, limited license to compile the complete generated code (comprised of the combination of the Restricted Source Code and the Unrestricted Source Code) into executable object code form, and to use, copy, distribute or license that executable. You may not distribute or redistribute, sublicense, sell, or transfer the Restricted Source Code to a third-party in the un-compiled form unless said third-party already has a license to the Restricted Source Code through their separate agreement with Altova. Notwithstanding anything to the contrary herein, you may not distribute, incorporate or combine with other software, or otherwise use the Altova Library modules or Restricted Source Code, or any Altova intellectual property embodied in or associated with the Altova Library modules or Restricted Source Code, in any manner that would subject the Restricted Source Code to the terms of a copyleft, free software or open source license that would require the Restricted Source Code or Altova Library modules source code to be disclosed in source code form. Notwithstanding anything to the contrary herein, you may not use the Software to develop and distribute other software programs that directly compete with any Altova software or service without prior written permission. Altova reserves all other rights in and to the Software. With respect to the feature(s) of UModel that permit reverse-engineering of your own source code or other source code that you have lawfully obtained, such use by you does not constitute a violation of this Agreement. Except as otherwise expressly permitted in Section 1(j) reverse engineering of the Software is strictly prohibited as further detailed therein.

(iv) In the event Restricted Source Code is incorporated into executable object code form, you will include the following statement in (1) introductory splash screens, or if none, within one or more screens readily accessible by the end-user, and (2) in the electronic and/or hard copy documentation: "Portions of this program were developed using Altova® [name of Altova Software, e.g. MapForce® 2016] and includes libraries owned by Altova GmbH, Copyright © 2007-2016 Altova GmbH (www.altova.com)."

(b) Server Use for Installation and Use of SchemaAgent. You may install one (1) copy of the Software on a computer file server within your internal network solely for the purpose of downloading and installing the Software onto other computers within your internal network up to the Permitted Number of computers in a commercial environment only. If you have licensed SchemaAgent, then you may install SchemaAgent Server on any server computer or workstation and use it in connection with your Software. No other network use is permitted, including without limitation using the Software either directly or through commands, data or instructions from or to a computer not part of your internal network, for Internet or Web-hosting services or by any user not licensed to use this copy of the Software through a valid license from Altova.

(c) Named-Use. If you have licensed the "Named-User" version of the software, you may install the Software on up to five (5) compatible personal computers or workstations of which you are the primary user thereby allowing you to switch from one computer to the other as necessary provided that only one (1) instance of the Software will be used by you as the Named-User at any given time. If you have purchased multiple Named-User licenses, each individual Named-User will receive a separate license key code.

(d) Concurrent Use in Same Local Area Network (LAN). If you have licensed a "Concurrent-User" version of the Software, you may install the Software on any compatible computers in a commercial environment only, up to ten (10) times the Permitted Number of users, provided that only the Permitted Number of users actually use the Software at the same time and further provided that the computers on which the Software is installed are on the same local area network (LAN). The Permitted Number of concurrent users shall be delineated at such time as you elect to purchase the Software licenses. Each separate local area network (LAN) requires its own set of separate Concurrent User Licenses for those wishing to use the Concurrent User versions of the Software in more than one location or on more than one network, all subject to the above Permitted Number limitations and based on the number of users using the Software. If a computer is not on the same local area network (LAN), then a locally installed user license or a license dedicated to concurrent use in a virtual environment is required.

(e) Concurrent Use in Virtual Environment. If you have purchased Concurrent-User Licenses, you may install a copy of the Software on a single host terminal server (Microsoft Terminal Server or Citrix Metaframe), application virtualization server (Microsoft App-V, Citrix XenApp, or VMWare ThinApp) or virtual machine environment within your internal network for the sole and exclusive purpose of permitting individual users within your organization to access and use the Software through a terminal server, application virtualization session, or virtual machine environment from another computer provided that the total number of users that access or use the Software concurrently at any given point in time on such network, virtual machine or terminal server does not exceed the Permitted Number; and provided that the total number of users authorized to use the Software through the terminal server, application virtualization session, or virtual machine environment does not exceed ten (10) times the Permitted Number of users. Key codes for concurrent users cannot be deployed to more than one host terminal server, application virtualization server or virtual machine environment. You must deploy a reliable and accurate means of preventing users from exceeding the Permitted Number of concurrent users. Altova makes no warranties or representations about the performance of Altova software in a terminal server, application virtualization session, or virtual machine environment and the foregoing are expressly excluded from the limited warranty in Section 5 hereof. Technical support is not available with respect to issues arising from use in such environments.

(f) Backup and Archival Copies. You may make one (1) backup and one (1) archival copy of the Software, provided your backup and archival copies are not installed or used on any computer and further provided that all such copies shall bear the original and unmodified copyright, patent and other intellectual property markings that appear on or in the Software. You may not transfer the rights to a backup or archival copy unless you transfer all rights in the Software as provided under Section 3.

(g) Key Codes, Upgrades and Updates. Prior to your purchase and as part of the registration for the thirty (30) day evaluation period, as applicable, you will receive an evaluation key code. You will receive a purchase key code when you elect to purchase the Software from either Altova GmbH or an authorized reseller. The purchase key code will enable you to activate the Software beyond the initial evaluation period. You may not re-license, reproduce or distribute any key code except with the express written permission of Altova. If the Software that you have licensed is an upgrade or an update, then the latest update or upgrade that you download and install replaces all or part of the Software previously licensed. The update or upgrade and the associated license keys does not constitute the granting of a second license to the Software in that you may not use the upgrade or updated copy in addition to the copy of the Software that it is replacing and whose license has terminated.

(h) Title. Title to the Software is not transferred to you. Ownership of all copies of the Software and of copies made by you is vested in Altova, subject to the rights of use granted to you in this Agreement. As between you and Altova, documents, files, stylesheets, generated program code (including the Unrestricted Source Code) and schemas that are authored or created by you via your utilization of the Software, in accordance with its Documentation and the terms of this Agreement, are your property unless they are created using Evaluation Software, as defined in Section 4 of this Agreement, in which case you have only a limited license to use any output that contains generated program code (including Unrestricted Source Code) such as Java, C++, C#, VB.NET or XSLT and associated project files and build scripts, as well as generated XML,

XML Schemas, documentation, UML diagrams, and database structures only for the thirty (30) day evaluation period.

(i) Reverse Engineering. Except and to the limited extent as may be otherwise specifically provided by applicable law in the European Union, you may not reverse engineer, decompile, disassemble or otherwise attempt to discover the source code, underlying ideas, underlying user interface techniques or algorithms of the Software by any means whatsoever, directly or indirectly, or disclose any of the foregoing, except to the extent you may be expressly permitted to decompile under applicable law in the European Union, if it is essential to do so in order to achieve operability of the Software with another software program, and you have first requested Altova to provide the information necessary to achieve such operability and Altova has not made such information available. Altova has the right to impose reasonable conditions and to request a reasonable fee before providing such information. Any information supplied by Altova or obtained by you, as permitted hereunder, may only be used by you for the purpose described herein and may not be disclosed to any third party or used to create any software which is substantially similar to the expression of the Software. Requests for information from users in the European Union with respect to the above should be directed to the Altova Customer Support Department.

(j) Other Restrictions. You may not loan, rent, lease, sublicense, distribute or otherwise transfer all or any portion of the Software to third parties except to the limited extent set forth in Section 3 or as otherwise expressly provided. You may not copy the Software except as expressly set forth above, and any copies that you are permitted to make pursuant to this Agreement must contain the same copyright, patent and other intellectual property markings that appear on or in the Software. You may not modify, adapt or translate the Software. You may not, directly or indirectly, encumber or suffer to exist any lien or security interest on the Software; knowingly take any action that would cause the Software to be placed in the public domain; or use the Software in any computer environment not specified in this Agreement. You may not permit any use of or access to the Software by any third party in connection with a commercial service offering, such as for a cloud-based or web-based SaaS offering.

You will comply with applicable law and Altova's instructions regarding the use of the Software. You agree to notify your employees and agents who may have access to the Software of the restrictions contained in this Agreement and to ensure their compliance with these restrictions.

(k) NO GUARANTEE. THE SOFTWARE IS NEITHER GUARANTEED NOR WARRANTED TO BE ERROR-FREE NOR SHALL ANY LIABILITY BE ASSUMED BY ALTOVA IN THIS RESPECT. NOTWITHSTANDING ANY SUPPORT FOR ANY TECHNICAL STANDARD, THE SOFTWARE IS NOT INTENDED FOR USE IN OR IN CONNECTION WITH, WITHOUT LIMITATION, THE OPERATION OF NUCLEAR FACILITIES, AIRCRAFT NAVIGATION, COMMUNICATION SYSTEMS, AIR TRAFFIC CONTROL EQUIPMENT, MEDICAL DEVICES OR LIFE SUPPORT SYSTEMS, MEDICAL OR HEALTH CARE APPLICATIONS, OR OTHER APPLICATIONS WHERE THE FAILURE OF THE SOFTWARE OR ERRORS IN DATA PROCESSING COULD LEAD TO DEATH, PERSONAL INJURY OR SEVERE PHYSICAL OR ENVIRONMENTAL DAMAGE. YOU AGREE THAT YOU ARE SOLELY RESPONSIBLE FOR THE ACCURACY AND ADEQUACY OF THE SOFTWARE AND ANY DATA GENERATED OR PROCESSED BY THE SOFTWARE FOR YOUR INTENDED USE AND YOU WILL DEFEND, INDEMNIFY AND HOLD ALTOVA, ITS OFFICERS AND EMPLOYEES HARMLESS FROM ANY THIRD PARTY CLAIMS, DEMANDS, OR SUITS THAT ARE BASED UPON THE ACCURACY AND ADEQUACY OF THE SOFTWARE IN YOUR USE OR ANY DATA GENERATED BY THE SOFTWARE IN YOUR USE.

2. INTELLECTUAL PROPERTY RIGHTS

You acknowledge that the Software and any copies that you are authorized by Altova to make are the intellectual property of and are owned by Altova and its suppliers. The structure, organization and code of the Software are the valuable trade secrets and confidential information of Altova and its suppliers. The Software is protected by copyright, including without limitation by United States Copyright Law, international treaty provisions and applicable laws in the country in which it is being used. You acknowledge that Altova retains the ownership of all patents, copyrights,

trade secrets, trademarks and other intellectual property rights pertaining to the Software, and that Altova's ownership rights extend to any images, photographs, animations, videos, audio, music, text and "applets" incorporated into the Software and all accompanying printed materials. You will take no actions which adversely affect Altova's intellectual property rights in the Software. Trademarks shall be used in accordance with accepted trademark practice, including identification of trademark owners' names. Trademarks may only be used to identify printed output produced by the Software, and such use of any trademark does not give you any right of ownership in that trademark. Altova®, XMLSpy®, Authentic®, StyleVision®, MapForce®, UModel®, DatabaseSpy®, DiffDog®, SchemaAgent®, SemanticWorks®, MissionKit®, Markup Your Mind®, Nanonull™, RaptorXML™, RaptorXML Server™, RaptorXML +XBRL Server™, Powered By RaptorXML™, FlowForce Server™, StyleVision Server™, and MapForce Server™ are trademarks of Altova GmbH. (pending or registered in numerous countries). Unicode and the Unicode Logo are trademarks of Unicode, Inc. Windows, Windows XP, Windows Vista, Windows 7, and Windows 8 are trademarks of Microsoft. W3C, CSS, DOM, MathML, RDF, XHTML, XML and XSL are trademarks (registered in numerous countries) of the World Wide Web Consortium (W3C); marks of the W3C are registered and held by its host institutions, MIT, INRIA and Keio. Except as expressly stated above, this Agreement does not grant you any intellectual property rights in the Software. Notifications of claimed copyright infringement should be sent to Altova's copyright agent as further provided on the Altova Web Site.

3. LIMITED TRANSFER RIGHTS

Notwithstanding the foregoing, you may transfer all your rights to use the Software to another person or legal entity provided that: (a) you also transfer this Agreement, the Software and all other software or hardware bundled or pre-installed with the Software, including all copies, updates and prior versions, and all copies of font software converted into other formats, to such person or entity; (b) you retain no copies, including backups and copies stored on a computer; (c) the receiving party secures a personalized key code from Altova; and (d) the receiving party accepts the terms and conditions of this Agreement and any other terms and conditions upon which you legally purchased a license to the Software. Notwithstanding the foregoing, you may not transfer education, pre-release, or not-for-resale copies of the Software.

4. PRE-RELEASE AND EVALUATION PRODUCT ADDITIONAL TERMS

If the product you have received with this license is pre-commercial release or beta Software ("Pre-release Software"), then this Section applies. In addition, this section applies to all evaluation and/or demonstration copies of Altova software ("Evaluation Software") and continues in effect until you purchase a license. To the extent that any provision in this section is in conflict with any other term or condition in this Agreement, this section shall supersede such other term(s) and condition(s) with respect to the Pre-release and/or Evaluation Software, but only to the extent necessary to resolve the conflict. You acknowledge that the Pre-release Software is a pre-release version, does not represent final product from Altova, and may contain bugs, errors and other problems that could cause system or other failures and data loss. CONSEQUENTLY, THE PRE-RELEASE AND/OR EVALUATION SOFTWARE IS PROVIDED TO YOU **"AS-IS" WITH NO WARRANTIES FOR USE OR PERFORMANCE,** AND ALTOVA DISCLAIMS ANY WARRANTY OR LIABILITY OBLIGATIONS TO YOU OF ANY KIND, WHETHER EXPRESS OR IMPLIED. WHERE LEGALLY LIABILITY CANNOT BE EXCLUDED FOR PRE-RELEASE AND/OR EVALUATION SOFTWARE, BUT IT MAY BE LIMITED, ALTOVA'S LIABILITY AND THAT OF ITS SUPPLIERS SHALL BE LIMITED TO THE SUM OF FIFTY DOLLARS (USD $50) IN TOTAL. If the Evaluation Software has a time-out feature, then the software will cease operation after the conclusion of the designated evaluation period. Upon such expiration date, your license will expire unless otherwise extended. Your license to use any output created with the Evaluation Software that contains generated program code (including Unrestricted Source Code) such as Java, C++, C, VB.NET or XSLT and associated project files and build scripts as well as generated XML, XML Schemas, documentation, UML diagrams, and database structures terminates automatically upon the expiration of the designated evaluation period but the license to use such output is revived upon your purchase of a license for the Software that you evaluated and used to create such output. Access to any files created with the Evaluation Software is entirely at your risk. You acknowledge that Altova has not promised or guaranteed to you that Pre-release Software will be announced or made available to anyone in the future, that Altova has no express or implied obligation to you to announce or introduce the Pre-release Software, and that Altova may not introduce a product similar to or compatible with the Pre-

release Software. Accordingly, you acknowledge that any research or development that you perform regarding the Pre-release Software or any product associated with the Pre-release Software is done entirely at your own risk. During the term of this Agreement, if requested by Altova, you will provide feedback to Altova regarding testing and use of the Pre-release Software, including error or bug reports. If you have been provided the Pre-release Software pursuant to a separate written agreement, your use of the Software is governed by such agreement. You may not sublicense, lease, loan, rent, distribute or otherwise transfer the Pre-release Software. Upon receipt of a later unreleased version of the Pre-release Software or release by Altova of a publicly released commercial version of the Software, whether as a stand-alone product or as part of a larger product, you agree to return or destroy all earlier Pre-release Software received from Altova and to abide by the terms of the license agreement for any such later versions of the Pre-release Software.

5. LIMITED WARRANTY AND LIMITATION OF LIABILITY

(a) Limited Warranty and Customer Remedies. Altova warrants to the person or entity that first purchases a license for use of the Software pursuant to the terms of this Agreement that (i) the Software will perform substantially in accordance with any accompanying Documentation for a period of ninety (90) days from the date of receipt, and (ii) any support services provided by Altova shall be substantially as described in Section 6 of this agreement. Some states and jurisdictions do not allow limitations on duration of an implied warranty, so the above limitation may not apply to you. To the extent allowed by applicable law, implied warranties on the Software, if any, are limited to ninety (90) days. Altova 's and its suppliers ' entire liability and your exclusive remedy shall be, at Altova 's option, either (i) return of the price paid, if any, or (ii) repair or replacement of the Software that does not meet Altova 's Limited Warranty and which is returned to Altova with a copy of your receipt. This Limited Warranty is void if failure of the Software has resulted from accident, abuse, misapplication, abnormal use, Trojan horse, virus, or any other malicious external code. Any replacement Software will be warranted for the remainder of the original warranty period or thirty (30) days, whichever is longer. This limited warranty does not apply to Evaluation and/ or Pre-release Software.

(b) No Other Warranties and Disclaimer. THE FOREGOING LIMITED WARRANTY AND REMEDIES STATE THE SOLE AND EXCLUSIVE REMEDIES FOR ALTOVA OR ITS SUPPLIER 'S BREACH OF WARRANTY. ALTOVA AND ITS SUPPLIERS DO NOT AND CANNOT WARRANT THE PERFORMANCE OR RESULTS YOU MAY OBTAIN BY USING THE SOFTWARE. EXCEPT FOR THE FOREGOING LIMITED WARRANTY, AND FOR ANY WARRANTY, CONDITION, REPRESENTATION OR TERM TO THE EXTENT WHICH THE SAME CANNOT OR MAY NOT BE EXCLUDED OR LIMITED BY LAW APPLICABLE TO YOU IN YOUR JURISDICTION, ALTOVA AND ITS SUPPLIERS MAKE NO WARRANTIES, CONDITIONS, REPRESENTATIONS OR TERMS, EXPRESS OR IMPLIED, WHETHER BY STATUTE, COMMON LAW, CUSTOM, USAGE OR OTHERWISE AS TO ANY OTHER MATTERS. TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, ALTOVA AND ITS SUPPLIERS DISCLAIM ALL OTHER WARRANTIES AND CONDITIONS, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, SATISFACTORY QUALITY, INFORMATIONAL CONTENT OR ACCURACY, QUIET ENJOYMENT, TITLE AND NON-INFRINGEMENT, WITH REGARD TO THE SOFTWARE, AND THE PROVISION OF OR FAILURE TO PROVIDE SUPPORT SERVICES. THIS LIMITED WARRANTY GIVES YOU SPECIFIC LEGAL RIGHTS. YOU MAY HAVE OTHERS, WHICH VARY FROM STATE/ JURISDICTION TO STATE/ JURISDICTION.

(c) Limitation of Liability. TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW EVEN IF A REMEDY FAILS ITS ESSENTIAL PURPOSE, IN NO EVENT SHALL ALTOVA OR ITS SUPPLIERS BE LIABLE FOR ANY SPECIAL, INCIDENTAL, DIRECT, INDIRECT OR CONSEQUENTIAL DAMAGES WHATSOEVER (INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOSS OF BUSINESS PROFITS, BUSINESS INTERRUPTION, LOSS OF BUSINESS INFORMATION, OR ANY OTHER PECUNIARY LOSS) ARISING OUT OF THE USE OF OR INABILITY TO USE THE SOFTWARE OR THE PROVISION OF OR FAILURE TO PROVIDE SUPPORT SERVICES, EVEN IF ALTOVA HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. IN ANY CASE, ALTOVA 'S ENTIRE LIABILITY UNDER ANY PROVISION OF THIS AGREEMENT SHALL BE LIMITED TO THE AMOUNT ACTUALLY PAID BY YOU FOR THE SOFTWARE PRODUCT. Because some states and jurisdictions do not allow the exclusion or limitation of liability, the above limitation may not apply to you. In such states and jurisdictions,

Altova 's liability shall be limited to the greatest extent permitted by law and the limitations or exclusions of warranties and liability contained herein do not prejudice applicable statutory consumer rights of person acquiring goods otherwise than in the course of business. The disclaimer and limited liability above are fundamental to this Agreement between Altova and you.

(d) Infringement Claims. Altova will indemnify and hold you harmless and will defend or settle any claim, suit or proceeding brought against you by a third party that is based upon a claim that the content contained in the Software infringes a copyright or violates an intellectual or proprietary right protected by United States or European Union law ("Claim"),but only to the extent the Claim arises directly out of the use of the Software and subject to the limitations set forth in Section 5 of this Agreement except as otherwise expressly provided. You must notify Altova in writing of any Claim within ten (10) business days after you first receive notice of the Claim, and you shall provide to Altova at no cost such assistance and cooperation as Altova may reasonably request from time to time in connection with the defense of the Claim. Altova shall have sole control over any Claim (including, without limitation, the selection of counsel and the right to settle on your behalf on any terms Altova deems desirable in the sole exercise of its discretion). You may, at your sole cost, retain separate counsel and participate in the defense or settlement negotiations. Altova shall pay actual damages, costs, and attorney fees awarded against you (or payable by you pursuant to a settlement agreement) in connection with a Claim to the extent such direct damages and costs are not reimbursed to you by insurance or a third party, to an aggregate maximum equal to the purchase price of the Software. If the Software or its use becomes the subject of a Claim or its use is enjoined, or if in the opinion of Altova 's legal counsel the Software is likely to become the subject of a Claim, Altova shall attempt to resolve the Claim by using commercially reasonable efforts to modify the Software or obtain a license to continue using the Software. If in the opinion of Altova 's legal counsel the Claim, the injunction or potential Claim cannot be resolved through reasonable modification or licensing, Altova, at its own election, may terminate this Agreement without penalty, and will refund to you on a pro rata basis any fees paid in advance by you to Altova. THE FOREGOING CONSTITUTES ALTOVA 'S SOLE AND EXCLUSIVE LIABILITY FOR INTELLECTUAL PROPERTY INFRINGEMENT. This indemnity does not apply to situations where the alleged infringement, whether patent or otherwise, is the result of a combination of the Altova software and additional elements supplied by you.

6. SUPPORT AND MAINTENANCE

Altova offers multiple optional "Support & Maintenance Package(s)" ("SMP")for the version of Software product edition that you have licensed, which you may elect to purchase in addition to your Software license. The Support Period, hereinafter defined, covered by such SMP shall be delineated at such time as you elect to purchase a SMP. Your rights with respect to support and maintenance as well as your upgrade eligibility depend on your decision to purchase SMP and the level of SMP that you have purchased:

(a) If you have not purchased SMP, you will receive the Software AS IS and will not receive any maintenance releases or updates. However, Altova, at its option and in its sole discretion on a case by case basis, may decide to offer maintenance releases to you as a courtesy, but these maintenance releases will not include any new features in excess of the feature set at the time of your purchase of the Software. In addition, Altova will provide free technical support to you for thirty (30) days after the date of your purchase (the "Support Period" for the purposes of this paragraph 6(a), and Altova, in its sole discretion on a case by case basis, may also provide free courtesy technical support during your thirty (30) day evaluation period. Technical support is provided via a Web-based support form only, and there is no guaranteed response time.

(b) If you have purchased SMP, then solely for the duration of its delineated Support Period, **you are eligible to receive the version of the Software edition** that you have licensed and all maintenance releases and updates for that edition that are released during your Support Period. For the duration of your SMP 's Support Period, you will also be eligible to receive upgrades to the comparable edition of the next version of the Software that succeeds the Software edition that you have licensed for applicable upgrades released during your Support Period. The specific upgrade edition that you are eligible to receive based on your Support Period is further detailed in the SMP that you have purchased. Software that is introduced as separate product is not included in SMP. Maintenance releases, updates and upgrades may or may not include additional features. In addition, Altova will provide Priority Technical Support to you for

the duration of the Support Period. Priority Technical Support is provided via a Web-based support form only and Altova will make commercially reasonable efforts to respond via e-mail to all requests within forty-eight (48) hours during Altova's business hours (MO-FR, 8am UTC – 10pm UTC, Austrian and US holidays excluded) and to make reasonable efforts to provide work-arounds to errors reported in the Software.

During the Support Period you may also report any Software problem or error to Altova. If Altova determines that a reported reproducible material error in the Software exists and significantly impairs the usability and utility of the Software, Altova agrees to use reasonable commercial efforts to correct or provide a usable work-around solution in an upcoming maintenance release or update, which is made available at certain times at Altova's sole discretion.

If Altova, in its discretion, requests written verification of an error or malfunction discovered by you or requests supporting example files that exhibit the Software problem, you shall promptly provide such verification or files, by email, telecopy, or overnight mail, setting forth in reasonable detail the respects in which the Software fails to perform. You shall use reasonable efforts to cooperate in diagnosis or study of errors. Altova may include error corrections in maintenance releases, updates, or new major releases of the Software. Altova is not obligated to fix errors that are immaterial. Immaterial errors are those that do not significantly impact use of the Software as determined by Altova in its sole discretion. Whether or not you have purchased the Support & Maintenance Package, technical support only covers issues or questions resulting directly out of the operation of the Software and Altova will not provide you with generic consultation, assistance, or advice under any circumstances.

Updating Software may require the updating of software not covered by this Agreement before installation. Updates of the operating system and application software not specifically covered by this Agreement are your responsibility and will not be provided by Altova under this Agreement. Altova's obligations under this Section 6 are contingent upon your proper use of the Software and your compliance with the terms and conditions of this Agreement at all times. Altova shall be under no obligation to provide the above technical support if, in Altova's opinion, the Software has failed due to the following conditions: (i) damage caused by the relocation of the Software to another location or CPU; (ii) alterations, modifications or attempts to change the Software without Altova's written approval; (iii) causes external to the Software, such as natural disasters, the failure or fluctuation of electrical power, or computer equipment failure; (iv) your failure to maintain the Software at Altova's specified release level; or (v) use of the Software with other software without Altova's prior written approval. It will be your sole responsibility to: (i) comply with all Altova-specified operating and troubleshooting procedures and then notify Altova immediately of Software malfunction and provide Altova with complete information thereof; (ii) provide for the security of your confidential information; (iii) establish and maintain backup systems and procedures necessary to reconstruct lost or altered files, data or programs.

7. SOFTWARE ACTIVATION, UPDATES AND LICENSE METERING

(a) License Metering. The Software includes a built-in license metering module that is designed to assist you with monitoring license compliance in small local area networks (LAN). The metering module attempts to communicate with other machines on your local area network (LAN). You permit Altova to use your internal network for license monitoring for this purpose. This license metering module may be used to assist with your license compliance but should not be the sole method. Should your firewall settings block said communications, you must deploy an accurate means of monitoring usage by the end user and preventing users from using the Software more than the Permitted Number.

(b) License Compliance Monitoring. You are required to utilize a process or tool to ensure that the Permitted Number is not exceeded. Without prejudice or waiver of any potential violations of the Agreement, Altova may provide you with additional compliance tools should you be unable to accurately account for license usage within your organization. If provided with such a tool by Altova, you (a) are required to use it in order to comply with the terms of this Agreement and (b) permit Altova to use your internal network for license monitoring and metering and to generate compliance reports that are communicated to Altova from time to time.

(c) Software Activation. **The Software may use your internal network and Internet connection for the purpose of transmitting license-related data at the time of**

installation, registration, use, or update to an Altova Master License Server and validating the authenticity of the license-related data in order to protect Altova against unlicensed or illegal use of the Software and to improve customer service. Activation is based on the exchange of license related data between your computer and the Altova Master License Server. You agree that Altova may use these measures and you agree to follow any applicable requirements. You further agree that use of license key codes that are not or were not generated by Altova and lawfully obtained from Altova, or an authorized reseller as part of an effort to activate or use the Software violates Altova's intellectual property rights as well as the terms of this Agreement. You agree that efforts to circumvent or disable Altova's copyright protection mechanisms, the license management mechanism, or the Altova Master License Server violate Altova's intellectual property rights as well as the terms of this Agreement. Altova expressly reserves the rights to seek all available legal and equitable remedies to prevent such actions and to recover lost profits, damages and costs.

(d) LiveUpdate. Altova provides a new LiveUpdate notification service to you, which is free of charge. Altova may use your internal network and Internet connection for the purpose of transmitting license-related data to an Altova-operated LiveUpdate server to validate your license at appropriate intervals and determine if there is any update available for you.

(e) Use of Data. The terms and conditions of the Privacy Policy are set out in full at http://www.altova.com/privacy and are incorporated by reference into this Agreement. By your acceptance of the terms of this Agreement and/or use of the Software, you authorize the collection, use and disclosure of information collected by Altova for the purposes provided for in this Agreement and/or the Privacy Policy. Altova has the right in its sole discretion to amend this provision of the Agreement and/or Privacy Policy at any time. You are encouraged to review the terms of the Privacy Policy as posted on the Altova Web site from time to time.

(f) Audit Rights. You agree that Altova may audit your use of the Software for compliance with the terms of this Agreement at any time, upon reasonable notice. In the event that such audit reveals any use of the Software by you other than in full compliance with the terms of this Agreement, you shall reimburse Altova for all reasonable expenses related to such audit in addition to any other liabilities you may incur as a result of such non-compliance.

(g) Notice to European Users. Please note that the information as described in paragraph 7(d) above may be transferred outside of the European Economic Area, for purposes of processing, analysis, and review, by Altova, Inc., a company located in Beverly, Massachusetts, U.S.A., or its subsidiaries or Altova's subsidiaries or divisions, or authorized partners, located worldwide. You are advised that the United States uses a sectoral model of privacy protection that relies on a mix of legislation, governmental regulation, and self-regulation. You are further advised that the Council of the European Union has found that this model does not provide "adequate" privacy protections as contemplated by Article 25 of the European Union's Data Directive. (Directive 95/46/EC, 1995 O.J. (L 281) 31). Article 26 of the European Union's Data Directive allows for transfer of personal data from the European Union to a third country if the individual has unambiguously given his consent to the transfer of personal information, regardless of the third country's level of protection. By agreeing to this Agreement, you consent to the transfer of all such information to the United States and the processing of that information as described in this Agreement and the Privacy Policy.

8. TERM AND TERMINATION

This Agreement may be terminated (a) by your giving Altova written notice of termination; (b) by Altova, at its option, giving you written notice of termination if you commit a breach of this Agreement and fail to cure such breach within ten (10) days after notice from Altova; or (c) at the request of an authorized Altova reseller in the event that you fail to make your license payment or other monies due and payable. In addition the Agreement governing your use of a previous version of the Software that you have upgraded or updated is terminated upon your acceptance of the terms and conditions of the Agreement accompanying such upgrade or update. Upon any termination of the Agreement, you must cease all use of the Software that this Agreement governs, destroy all copies then in your possession or control and take such other actions as Altova may reasonably request to ensure that no copies of the Software remain in

your possession or control. The terms and conditions set forth in Sections 1(h), 1(i), 1(j), 1(k), 1(l), 2, 5, 7, 9, 10, 11, and 11 survive termination as applicable.

9. RESTRICTED RIGHTS NOTICE AND EXPORT RESTRICTIONS

The Software was developed entirely at private expense and is commercial computer software provided with **RESTRICTED RIGHTS**. Use, duplication or disclosure by the U.S. Government or a U.S. Government contractor or subcontractor is subject to the restrictions set forth in this Agreement and as provided in FAR 12.211 and 12.212 (48 C.F.R. § 12.211 and 12.212) or DFARS 227. 7202 (48 C.F.R. § 227- 7202) as applicable. Consistent with the above as applicable, Commercial Computer Software and Commercial Computer Documentation licensed to U.S. government end users only as commercial items and only with those rights as are granted to all other end users under the terms and conditions set forth in this Agreement. Manufacturer is Altova GmbH, Rudolfsplatz 13a/ 9, A- 1010 Vienna, Austria/ EU. You may not use or otherwise export or re- export the Software or Documentation except as authorized by United States law and the laws of the jurisdiction in which the Software was obtained. In particular, but without limitation, the Software or Documentation may not be exported or re- exported (i) into (or to a national or resident of) any U.S. embargoed country or (ii) to anyone on the U.S. Treasury Department's list of Specially Designated Nationals or the U.S. Department of Commerce's Table of Denial Orders. By using the Software, you represent and warrant that you are not located in, under control of, or a national or resident of any such country or on any such list.

10. U.S. GOVERNMENT ENTITIES

Notwithstanding the foregoing, if you are an agency, instrumentality or department of the federal government of the United States, then this Agreement shall be governed in accordance with the laws of the United States of America, and in the absence of applicable federal law, the laws of the Commonwealth of Massachusetts will apply. Further, and notwithstanding anything to the contrary in this Agreement (including but not limited to Section 5 (Indemnification)), all claims, demands, complaints and disputes will be subject to the Contract Disputes Act (41 U.S.C. § § 7101 *et seq.*), the Tucker Act (28 U.S.C. § 1346(a) and § 1491), or the Federal Tort Claims Act (28 U.S.C. § § 1346(b), 2401- 2402, 2671- 2672, 2674- 2680), FAR 1.601(a) and 43.102 (Contract Modifications); FAR 12.302(b), as applicable, or other applicable governing authority. For the avoidance of doubt, if you are an agency, instrumentality, or department of the federal, state or local government of the U.S. or a U.S. public and accredited educational institution, then your indemnification obligations are only applicable to the extent they would not cause you to violate any applicable law (e.g., the Anti- Deficiency Act), and you have any legally required authorization or authorizing statute.

11. THIRD PARTY SOFTWARE

The Software may contain third party software which requires notices and/ or additional terms and conditions. Such required third party software notices and/ or additional terms and conditions are located at our Website at http:/ / www.altova.com/ legal_3rdparty.html and are made a part of and incorporated by reference into this Agreement. By accepting this Agreement, you are also accepting the additional terms and conditions, if any, set forth therein.

12. JURISDICTION, CHOICE OF LAW, AND VENUE

If you are located in the European Union and are using the Software in the European Union and not in the United States, then this Agreement will be governed by and construed in accordance with the laws of the Republic of Austria (excluding its conflict of laws principles and the U.N. Convention on Contracts for the International Sale of Goods) and you expressly agree that exclusive jurisdiction for any claim or dispute with Altova or relating in any way to your use of the Software resides in the Handelsgericht, Wien (Commercial Court, Vienna) and you further agree and expressly consent to the exercise of personal jurisdiction in the Handelsgericht, Wien (Commercial Court, Vienna) in connection with any such dispute or claim.

If you are located in the United States or are using the Software in the United States then this Agreement will be governed by and construed in accordance with the laws of the Commonwealth of Massachusetts, USA (excluding its conflict of laws principles and the U.N. Convention on Contracts for the International Sale of Goods) and you expressly agree that exclusive jurisdiction

for any claim or dispute with Altova or relating in any way to your use of the Software resides in the federal or state courts of the Commonwealth of Massachusetts and you further agree and expressly consent to the exercise of personal jurisdiction in the federal or state courts of the Commonwealth of Massachusetts in connection with any such dispute or claim.

If you are located outside of the European Union or the United States and are not using the Software in the United States, then this Agreement will be governed by and construed in accordance with the laws of the Republic of Austria (excluding its conflict of laws principles and the U.N. Convention on Contracts for the International Sale of Goods) and you expressly agree that exclusive jurisdiction for any claim or dispute with Altova or relating in any way to your use of the Software resides in the Handelsgericht, Wien (Commercial Court, Vienna) and you further agree and expressly consent to the exercise of personal jurisdiction in the Handelsgericht Wien (Commercial Court, Vienna) in connection with any such dispute or claim. This Agreement will not be governed by the conflict of law rules of any jurisdiction or the United Nations Convention on Contracts for the International Sale of Goods, the application of which is expressly excluded.

13. TRANSLATIONS

Where Altova has provided you with a foreign translation of the English language version, you agree that the translation is provided for your convenience only and that the English language version will control. If there is any contradiction between the English language version and a translation, then the English language version shall take precedence.

14. GENERAL PROVISIONS

This Agreement contains the entire agreement and understanding of the parties with respect to the subject matter hereof, and supersedes all prior written and oral understandings of the parties with respect to the subject matter hereof. Any notice or other communication given under this Agreement shall be in writing and shall have been properly given by either of us to the other if sent by certified or registered mail, return receipt requested, or by overnight courier to the address shown on Altova 's Web site for Altova and the address shown in Altova 's records for you, or such other address as the parties may designate by notice given in the manner set forth above. This Agreement will bind and inure to the benefit of the parties and our respective heirs, personal and legal representatives, affiliates, successors and permitted assigns. The failure of either of us at any time to require performance of any provision hereof shall in no manner affect such party 's right at a later time to enforce the same or any other term of this Agreement. This Agreement may be amended only by a document in writing signed by both of us. In the event of a breach or threatened breach of this Agreement by either party, the other shall have all applicable equitable as well as legal remedies. Each party is duly authorized and empowered to enter into and perform this Agreement. If, for any reason, any provision of this Agreement is held invalid or otherwise unenforceable, such invalidity or unenforceability shall not affect the remainder of this Agreement, and this Agreement shall continue in full force and effect to the fullest extent allowed by law. The parties knowingly and expressly consent to the foregoing terms and conditions.

Last updated: 2015/ 09/ 03

Chapter 12

用語

12 用語

この用語のセクションはMapForce に関連する用語のリストを含みます。

12.1 C

コンポーネント

MapForce 内では、"コンポーネント"という用語はデータの構造 (スキーマ)またはデータがどのように変換 (関数)されるかを視覚的に表します。コンポーネントはすべてのマッピングを構築する中心的な役割を果たします。以下は、MapForce コンポーネントの例です :

- 定数
- フィルター
- 条件
- 関数コンポーネント
- EDI ドキュメント (UN/EDIFACT、ANSI X12、HL7)
- Excel 2007+ ファイル
- 単純型 入力コンポーネント
- 単純型 出力コンポーネント
- スキーマおよびDTD

接続

接続とは2つのコネクタ間に描くことができる線です。接続を描くことにより、MapForce にデータを特定の方法で変換するように命令します。(例 :XML ドキュメントからデータを読み込み、他のXML ドキュメントに書き込みます)。

コネクタ

コンポーネントの左右に表示される小さな三角形です。コンポーネントの左に表示されるコネクタはそのコンポーネントのエントリポイントを表します。コンポーネントの右に表示されるコネクタはそのコンポーネントからのデータの終了ポイントを表します。

12.2 F

固定長フィールド (FLF)

データが習慣的に固定値を持つフィールドに分割される共通のテキストフォーマット。 (例 :行の最初の 5文字がトランザクションID を表し、次の 20文字がトランザクションの詳細を表すなど)。

FlexText

FlexText とは MapForce によりサポートされる他のフォーマットに変換するには複雑な非標準またはレガシーテキストファイルからのデータを変換するMapForce Enterprise Edition 内のモジュールです。

12.3 I

入力コンポーネント

入力コンポーネントは、単純型の値をマッピングにパスすることができるMapForce コンポーネントです。入力コンポーネントは通常ファイル名またはその他の文字列の値をランタイムにマッピングにパスするために使用されます。入力コンポーネントとソースコンポーネントを混同しないようご注意ください。

12.4 J

ジョインコンポーネント

ジョインコンポーネントは、カスタム定義条件をベースにしたマッピングの2つの構造をジョインすることができる MapForce コンポーネントです。ジョイン機能は、共通のフィールド (ID などの) を共有する2つのソース構造からデータを組み合わせる際に役にたちます。詳細に関しては、以下を参照してください:
データのジョイン。

12.5 M

MapForce

MapForce は、プログラムコードを作成することなく視覚的なドラッグアンドドロップを使用するグラフィカルなユーザーインターフェイスを用いデータを異なるデータ間、スキーマ間で変換するWindows ベースの多目的のIDE (統合された開発環境 統合開発環境) です。 また、事実上、MapForce は時際のデータ変換 (またデータマッピング)を行うプログラムコードを生成します。プログラムコードを生成しない場合、 (MapForce Professional または Enterprise Editionsで使用することのできる)MapForce 内蔵の変換言語 を実行します。

マッピング

MapForce マッピングデザイン (または、略して "マッピング")は、データがどのように1つのフォーマットから他のフォーマットに変換されるかの視覚的な表現です。マッピングは、データ変換を行うためのMapForce マッピングエリア内で1つのスキーマから他のスキーマに追加するコンポーネントから構成されています。 (例 :XML ドキュメントを1つのスキーマから他のスキーマに変換) 。有効なマッピングは、1つまたは複数のターゲットコンポーネントに接続されている、1つまたは複数の複数のソースコンポーネントから構成されています。マッピングを実行して、結果を直接 MapForce 内で確認することができます。コードを生成し、外部で実行することも可能です。MapForce 実行可能ファイルにマッピングをコンパイルし、MapForce Server または FlowForce Server を使用してマッピングの実行を自動化することもできます。 MapForce は、マッピングを .mfd の拡張子を持つファイルとして保存します。

MFF

MapForce 関数ファイルのファイル名の拡張子

MFD

MapForce デザインドキュメントのファイル名の拡張子 (マッピング)

12.6 O

出力コンポーネント

出力コンポーネント (または "単純型出力 ") とは、マッピングから文字列の値を返すことを有効化するMapForce コンポーネントです。出力 コンポーネントは、1つのターゲットコンポーネント型のみをあらわしますが、後者と混同しないように注意してください。

12.7 P

parent-context

`min`、`max`、`avg`、`count` などのMapFore 一部のコア集計関数 **parent-context** 任意の引数です。複数の階層的シーケンスを持つソースコンポーネントでは、親コンテキストは、ノードのセットがどの関数上で操作されるかを決定します。

12.8 S

ソース コンポーネント

ソースコンポーネントとは、MapForce がデータを読み込む元のコンポーネントです。マッピングを実行すると、ソースコンポーネントのコネクタにより与えられたデータを読み込み、必要とされる型に変換し、ターゲットコンポーネントのコネクタに送信します。

12.9 T

ターゲットコンポーネント

ターゲットコンポーネントとMapForce はがデータを書き込むコンポーネントです。マッピング を実行すると、ターゲットコンポーネントは、外部プログラム内で更に処理するためにMapForce にファイル (または複数のファイル)の生成、または結果を文字列の値として出力するよう命令します。ターゲットコンポーネントは、ソース コンポーネントの逆です。

Index

E

F

G

I

J

L

M

N

O

P

Q

R

S

T

U

V

W

X

Z