talend*
*open integration solutions

Talend Open Studio for MDM

Administrator Guide

5.2.1

Adapted for v5.2.1.

Copyleft

Notices

Table of Contents

Preface

1. General information

1.1. Purpose

This Administrator Guide explains how to manage Talend MDM functions in a normal operational context. It describes how to administrate and set the parameters for Talend MDM through a single console.

Information presented in this document applies to release **5.2.1** of *Talend Open Studio for MDM*.

1.2. Audience

This guide is for users and administrators involved in administrating and customizing Talend MDM.

The layout of GUI screens provided in this document may vary slightly from your actual GUI.

1.3. Typographical conventions

This guide uses the following typographical conventions:

- text in **bold:** window and dialog box buttons, field names, keyboard keys, menus and menu items,
- text in **[bold]:** titles of dialog boxes, wizards and windows,
- text in `courier`: system parameters typed in by users,
- text in *italics*: file, schema, column, row, variable names and text typed in by users,
- The  icon indicates an item that provides additional information about an important point. It is also used to add comments related to a table or a figure,
- The  icon indicates a message that gives information and recommendations about the execution requirements. It is also used to refer to situations or information the user needs to be aware of or pay special attention to.

2. Feedback and Support

Your feedback is valuable. Do not hesitate to give your input, make suggestions or requests regarding this documentation or product and find support from the **Talend** team, on **Talend**'s Forum website at:

http://talendforge.org/forum

Chapter 1. Talend MDM: Concepts & Principles

This chapter introduces Master Data Management as the means to manage shared data in an enterprise.

It also describes how the **MDM**, **Integration** and **Profiling** perspectives of *Talend Open Studio for MDM* are integrated together to provide the first complete Master Data Management solution that combines data profiling, data integration and master data functionalities on a single platform.

1.1. Introducing Master Data Management

Most large enterprises have a heterogeneous application portfolio, with fragments of often inaccurate, incomplete and inconsistent data residing in various application silos. Companies wanting a consistent view of their customer base often struggle to reconcile data across numerous operational systems. These issues cause intelligent decision making to be difficult. The heart of these problems lies in the handling of shared data.

Master Data Management (MDM) is a comprehensive method of enabling an enterprise to link all of its critical data to one "master file" that provides a common point of interest. Examples of master data include customer, product, asset, location, employee, organizational unit. Data that is not shared between functions within an organization is not master data.

MDM has emerged as a means to more efficiently manage shared data, eliminate redundancy, and create the elusive "single version of the truth”. Unlike most Enterprise Resource Planning (ERP) systems which are often monolithic, expensive and rigid, MDM is able to offer both consistency and agility, therefore providing a major competitive advantage and return on investment (ROI).

As businesses strive to dramatically reduce costs, meet compliance reporting mandates, deliver increased sales, and provide superior service to customers and suppliers, analysts have declared MDM as a solution which will significantly contribute to these business priorities.

1.2. General concepts

According to Wikipedia (http://en.wikipedia.org/wiki/Master_data_management), in computing, Master Data Management comprises a set of processes and tools that consistently defines and manages the non-transactional data entities of an organization. Its objective is to provide processes for collecting, aggregating, matching, consolidating, quality-assuring, persisting and distributing such data throughout an organization to ensure consistency and control in the ongoing maintenance and application use of this information.

1.2.1. Master data

Master data is data describing a physical or virtual object and its properties. Master data is usually described with nouns.

Typical master data can be:

- Physical: products, material, assets, customers, locations etc.
- Virtual: cost centers, planned buildings etc.

1.2.2. Transactional data

Transaction data is data describing an event (the change as a result of a transaction) and is usually described with verbs. Transaction data always has a time dimension, a numerical value and refers to one or more objects.

Typical transactions are:

- Financial: orders, invoices, payments, etc.

- Work: plans, activity records, etc.
- Logistics: deliveries, storage records, travel records, etc.

1.3. Master Data Management by Talend

Talend MDM is a model-drive, non intrusive solution easily adaptable to specific business needs and quick to implement. It has been specifically developed to address the challenges of creating and managing master data for all types of organizations where data is hosted under various formats in various systems and can be extremely volatile.

Talend MDM groups all master data of the company in a central hub. This standardized repository provides, through using Data Models, the prerequisites against which data and updates are validated.

Talend Open Studio for MDM presents a single platform that bundles data integration, data profiling and mastering and governing data in the same Studio.

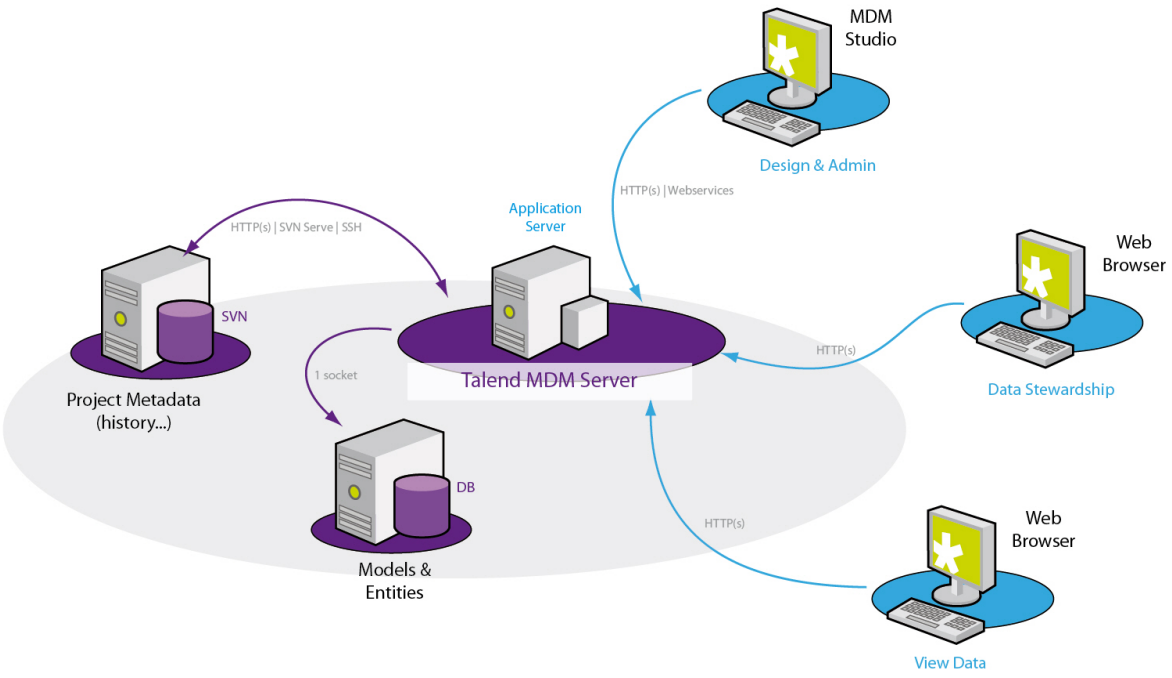
1.3.1. Overview of Talend MDM

Talend MDM has all the core features a user needs for an MDM application: advanced modeling, model-driven dynamic web interface, full-text search, event triggering, etc.

Talend MDM is a model-driven, non intrusive solution easily adaptable to specific business needs and it is quick to implement. It provides a complete set of features for mastering, governing and integrating data throughout the enterprise.

Talend MDM groups all master data of the company in a central hub. This standardized repository provides, via the use of Data Models, the prerequisites against which data and updates are validated.

The chart below illustrates the main building blocks of Talend MDM.



The list below describes the main building blocks of Talend MDM:

- MDM Studio - where administrators can set up and operate a centralized master data repository. They can build data models that employ the necessary business and data rules to create a single master copy of the master data. This master data will be propagated back to target and source systems.
- MDM Hub - where the master data is stored.
- MDM Repository - where a working copy of the data is stored before deployment to the MDM Server. The MDM Repository can be stored locally, on the same machine as the MDM Studio, or remotely, based on an SVN server.
- The web-based interface - where business users and data stewards can search, display or edit master data handled by the MDM Studio.

The building blocks you have available for use may change depending on which edition of MDM you have.

For further information about how Talend MDM fits in with the other products in the *Talend* suite, see the *Talend Installation Guide*.

1.3.2. User interfaces for data governance and data stewardship

Talend Open Studio for MDM provides the processing layers that ensure the right people have the right tools to centrally model and manipulate master data. This key capability comprises the relevant features involved in master data governance and stewardship.

Data governance is the process of defining the rules that master data has to follow. Data stewardship is the process of making sure that the data follows those rules. This means that it is necessary to have both a governance function, to demonstrate that the right controls are in place, and a stewardship function, to ensure that the controls are enforced.

In Talend MDM, the governance and stewardship processes are available through two different user interfaces. The first is an administration tool called *Talend Open Studio for MDM*. The second is a web-based interface called *Talend MDM Web User Interface*.

Users of *Talend Open Studio for MDM* can set the governance rules. For more information about available administration tasks, see chapter *Setting data governance rules*, and chapter *Advanced subjects*.

Users of *Talend MDM Web User Interface* can carry out any manual interventions necessary to make sure that the master data is clean, consistent, and accurate. For more information, see *Talend MDM Web User Interface User Guide*.

1.3.3. A comprehensive set of tools

The MDM solution offered by **Talend** is divided into various key capabilities. A comprehensive set of tools and functions are available to fulfill these key capabilities. And the available tools and functions are ALL accessible from one single interface called *Talend Open Studio for MDM*.

Inside *Talend Open Studio for MDM* the user will find:

- Data integration mechanisms: to insure that master data can be integrated from a wide range of sources,
- Data profiling mechanisms: that allows the profiling of the source data before loading it into the MDM Hub.

- Master data management mechanisms: to build data models that employ the necessary business and data rules to create one single master copy of the data which will be propagated back to the source and target systems.

Talend Open Studio for MDM bundles Data Integration, Data Profiling and Master Data Management in a single platform to enable the management of the complete data life cycle: from data integration to quality assurance, through to the identification of master data and its governance.

For more information about accessing different key capabilities perspectives, see section *Multi-perspective approach*.

1.3.3.1. Integration with *Talend Open Studio for Data Integration*

Talend Open Studio for MDM incorporates and extends the core capabilities of *Talend Open Studio for Data Integration*. The result is an innovative and powerful data integration solution with hundreds of ELT/ETL connectors enabling interaction with multiple sources and the synchronization of enterprise systems.

From the **Integration** perspective, users can design different Jobs using a dedicated component group that allows bulk loading of data.

Moreover, MDM's event management and **Talend** Jobs are tightly coupled so that users can call a Job from within *Talend Open Studio for MDM* to validate, correct or propagate data in the MDM Hub.

For detailed information about the **Talend** integration tool, see *Talend Open Studio for Data Integration User Guide*.

For detailed information about integration components, see the *Talend Open Studio Components Reference Guide*.

1.3.3.2. Integration with *Talend Open Studio for Data Quality*

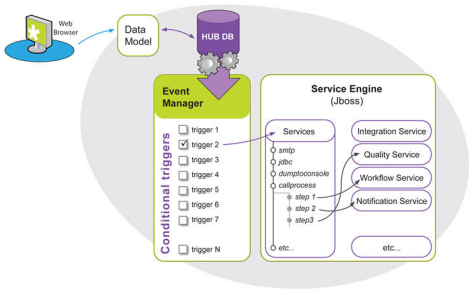
Talend Open Studio for MDM also incorporates *Talend Open Studio for Data Quality*, a powerful data profiling tool, to guarantee high standards of master data quality in your company.

From the **Profiling** perspective, users can profile data from various sources before loading it into the MDM Hub.

For detailed information about the **Talend** data quality tool, see *Talend Open Studio for Data Quality User Guide*.

1.4. Example of a functional workflow through Talend MDM

The below schema illustrates how data is handled in Talend MDM.



- A business user creates or modifies a master data record from *Talend MDM Web User Interface*,
- A Trigger that matches the conditions set on the data validated against a data model kicks in to initialize a service. Various services can be initialized including Processes (*callprocess*) such as data validation or a human validation process, an enrichment process, a data integration process, etc.
- A Process may use one or more plug-ins. Each plug-in performs certain tasks on master data. The most important plug-ins used by a given Process are *CallJob* and *XSLT*.

Chapter 2. Getting started with *Talend Open Studio for MDM*

This chapter introduces *Talend Open Studio for MDM*. It provides a short description of the Graphical User Interface (GUI) and gives definitions of basic terms.

This chapter guides you through the basics for launching *Talend Open Studio for MDM* and emphasizes the use of multiple perspectives, those of data integration, data quality and Master Data Management, in the same Studio.

2.1. Important terms in *Talend Open Studio for MDM*

When working with *Talend Open Studio for MDM* and in order to understand its functional mechanism, it is important to understand some basic vocabulary.

The following table defines some of the most essential terminology.

Term	Description
Annotation	Gives a description about the metadata that the administrator “attached” to an Entity in the data model.
Consumer	Consumes data FROM the MDM Hub. A consumer may also be a provider.
Data Container	Holds data of one or several business entities. Data containers are typically used to separate master data domains.
Data governance	The process of defining the rules that data has to follow within an organization.
Data Model	Defines the attributes, user access rights and relationships of entities mastered by the MDM Hub. The data model is the central component of Talend MDM. A data model maps to a single entity that can be explicitly defined. Any concept can be defined by a data model.
Data stewardship	The process of validating master data against the rules (data models) that are set in the MDM Studio.
Domain	A collection of data models that define a particular concept. For instance, the customer domain may be defined by the organization, account, contact and opportunity data models. A product domain may be defined by a product, product family and price list. Ultimately, the domain is the collection of all entities (data models) that relate to a concept. Talend MDM can model any and many domains within a single hub. It is a generic multi-domain MDM solution.
Entity	Describes the actual data, its nature, its structure and its relationships. A data model can have multiple entities.
Event Manager	A service of the MDM Hub responsible for routing events thrown by the MDM Hub to trigger, evaluate their conditions, execute Processes, and trace active / completed / failed actions for monitoring purpose.
MDM Hub	Defines a complete Talend MDM implementation. It consists of components for Integration, Quality, Master Data Model, an XML DB interface and operational database, Web Services, MDM Web Interface. The MDM Hub is configured to meet different business needs.
Process	A Process is executed when the condition specified by the corresponding Trigger is verified. A Process may have several "steps", each step performs a specific task such as: update a record in the hub, run a **Talend** Job, etc.
Provider	Feeds data IN to the MDM Hub.
Record	An instance of data defined by a data model in the MDM Hub. Two records may be compared and considered similar or a close match, in which case the records may be linked and one may or may not survive.
Talend Studio	The administration user interface built from Eclipse. It allows the administrator of the system to manage and maintain the MDM Hub and all associated Data Integration Jobs through a single console.
Triggers	Condition for a Process to be executed, based on events thrown by the MDM Hub. Example of a Trigger condition: *Agency created and Agency/Revenue > 100*. An event may cause more than one Trigger conditions to be true, which will result in several Processes to be executed. Triggers are used to specify when specific Processes such as notifications, duplicate checking, records enrichment, propagation to back end systems, etc. should be executed.
Version	Partition of the MDM Hub where part, or all, objects may be modified separately. The MDM Hub has at least one Version shared by all users, which is considered as the "main" Version. Users typically create additional Versions for sandboxing modifications before they are made available in the main Version. This may also be used to isolate a derived Version of the MDM Hub for a specific group of users. A Version can have its own copy of models, entities, views, etc., or it may share objects with other Versions - for instance it may have a specific Version of the Product entity but still share the same model and views of the main Version.
View	A complete or a subset view of a record. A complete view shows all elements or columns in an entity, while a subset view shows some of the elements or columns in an entity. A View may restrict access to attributes of a record depending on who or what is asking for the data.

2.2. Launching *Talend Open Studio for MDM* and connecting to the MDM server

Talend Open Studio for MDM is the administration and parameter setting tool for the MDM platform. It is the Graphical User Interface for administrators and it is run as an Eclipse plug-in.

Prerequisite(s): Make sure to install JBoss and the MDM server before launching *Talend Open Studio for MDM*. For further information on the installation procedure using the .jar file, see *Talend MDM Installation Guide*. For further information on the installation procedure using the .exe file, see *Talend MDM Installation Guide*.

2.2.1. Launching *Talend Open Studio for MDM*

To launch *Talend Open Studio for MDM* for the first time, do the following:

1. Launch the MDM server: in the JBoss folder, double-click *run.bat* if you are on Windows or *run.sh* if you are on Linux.

   Talend MDM server is a J2EE application, it comprises numerous J2EE artifacts. Allow several minutes for JBoss to deploy the MDM Server when it starts up.

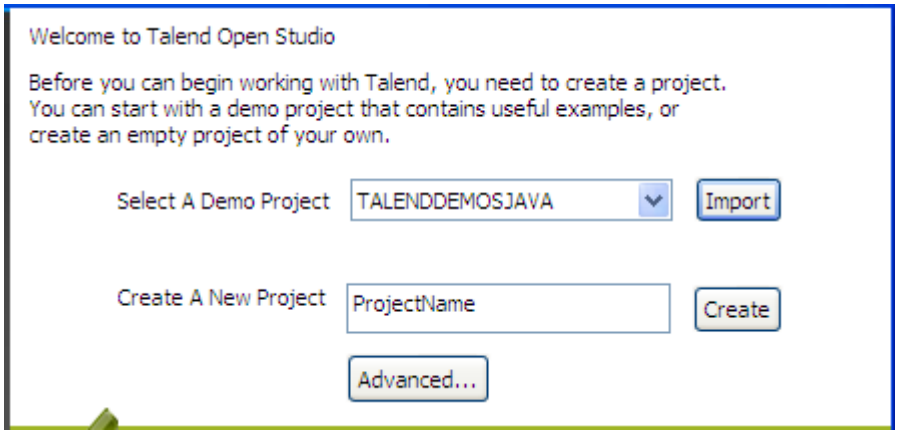
   If you set a PATH environment variable to specify where executable programs such as run.bat files are located, make sure that this variable DOES NOT end with a backslash, otherwise the execution of the server will fail.

2. Unzip the Studio zip folder and open it.

3. Double-click the executable file that corresponds to your platform, for example *TMDMCE-win32-x86.exe* on Windows or *TMDMCE-linux-gtk-x86* on Linux 32 bits Intel, etc.

4. In the **[License]** window that appears, read and accept the terms of the license agreement to proceed to the next step.

   The startup window opens.

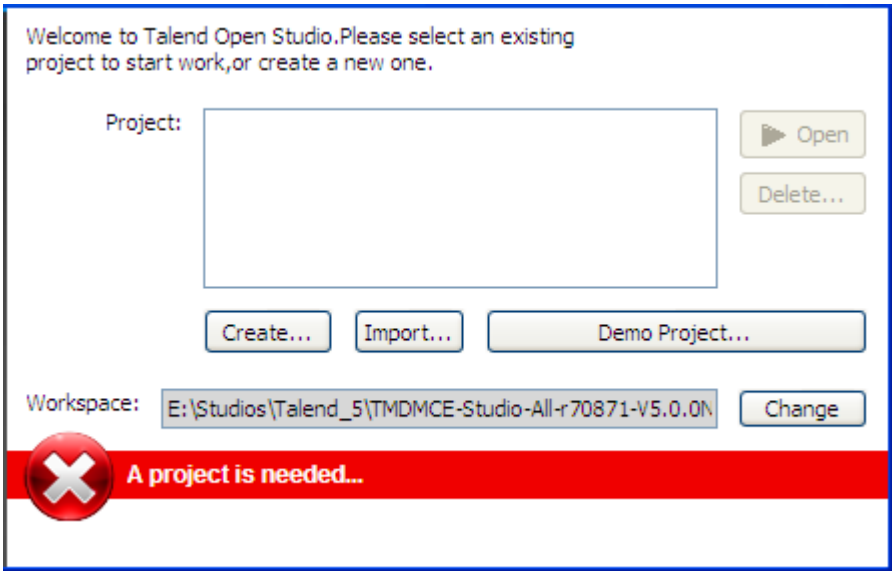
   

   This screen appears only when you launch the *Talend Open Studio for MDM* for the first time or if all existing projects have been deleted.

   The launching procedures for *Talend Open Studio for MDM* are the same as those for *Talend Open Studio for Data Integration*. For more detailed information about launching the Studio, check *Talend Open Studio for Data Integration User Guide*.

5. Click the **Import** button to import the selected demo project, or type in a project name in the **Create A New Project** field and click the **Create** button to create a new project, or click the **Advanced...** button to go to the Studio login window.

In this procedure, click **Advanced...** to go to the Studio login widow.



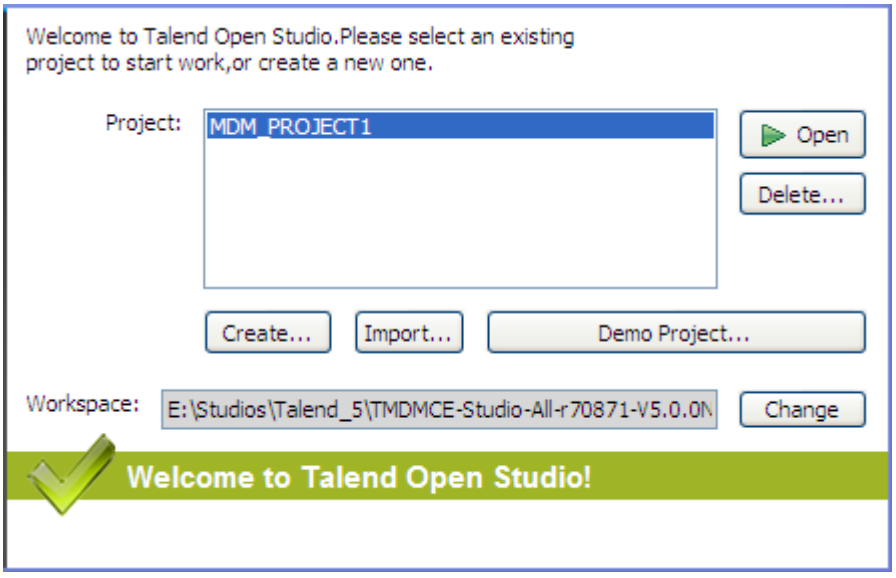
6. From the login window:

Click...	**To...**
Create...	create a project where you can store all the Jobs and Business Models you create in the Studio.
Import...	import any project you have.
Demo Project...	import the Demo project that includes numerous samples of ready to use Jobs. This Demo project can help you understand the functionalities of different Talend components.
Open	open the selected project.
Delete...	open a dialog box where you can delete any of the listed projects that you already created or imported and that you do not need any more.

As the purpose of this procedure is to create a new project, click **Create...** to open the **[New project]** dialog box.

7. In the **[New project]** dialog box, enter a name for the project, and then click **Finish** to close the dialog box.

The name of the new project displays in the **Project** field.



8. Select the project, and click **Open**.

   The **Connect to TalendForge** page appears, inviting you to connect to the **Talend** Community so that you can check, download, and install external components, and upload your own components to the **Talend** Community to share with other **Talend** users directly in the **Exchange** view of your Job designer in the Studio.

   To learn more about the **Talend** Community, click the **read more** link. For more information on using and sharing community components, see *Talend Open Studio for Data Integration User Guide*.

9. If you want to connect to the **Talend** Community later, click **Skip** to continue.

10. If you are working behind a proxy, click **Proxy setting** and fill in the **Proxy Host** and **Proxy Port** fields of the **Network setting** dialog box.

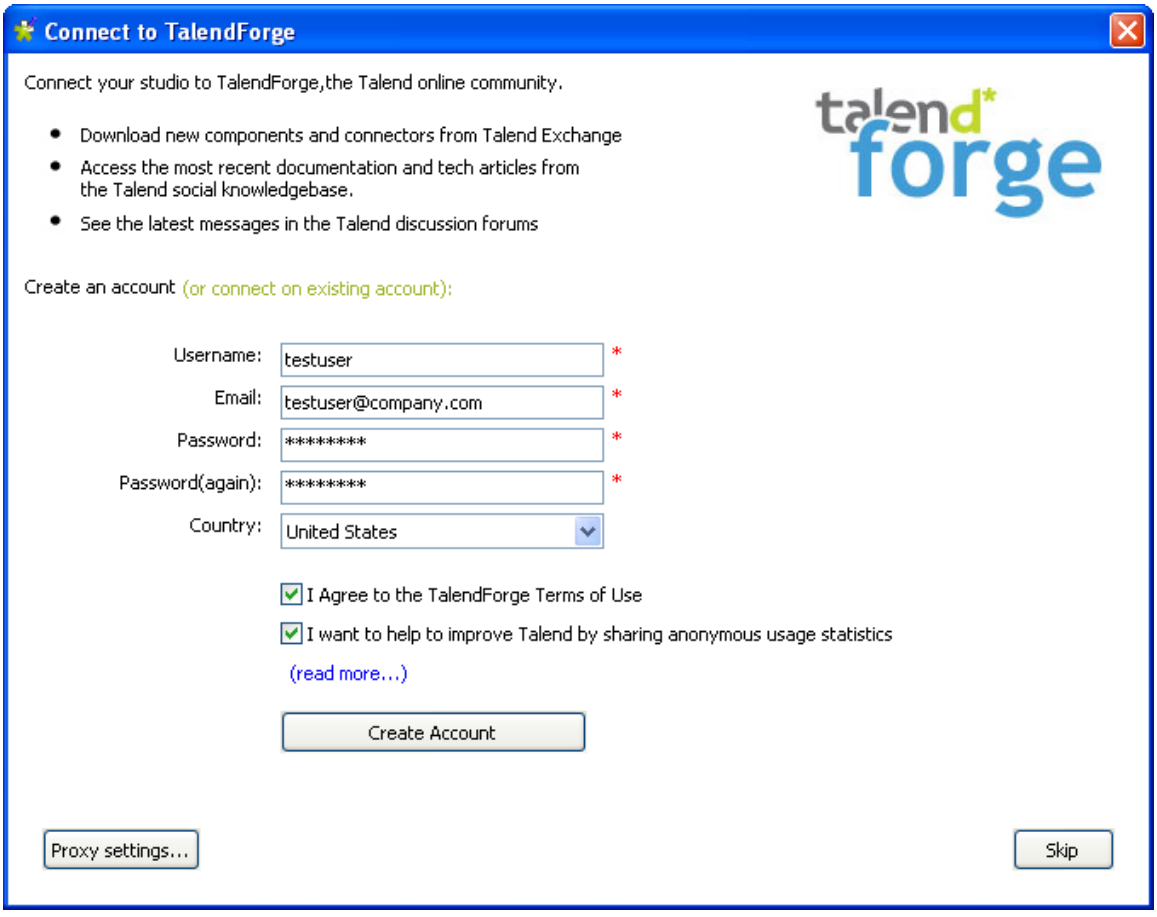
11. By default, the Studio will automatically collect product usage data and send the data periodically to servers hosted by **Talend** for product usage analysis ans sharing purposes only. If you do not want the Studio to do so, clear the **I want to help to improve Talend by sharing anonymous usage statistics** check box.

    You can also turn on or off usage data collection in the Usage Data Collector preferences settings. For more information the Usage Data Collector, see *Talend Open Studio for Data Integration User Guide*.

12. Fill in the required information, select the **I Agree to the TalendForge Terms of Use** check box, and click **Create Account** to create your account and connect to the **Talend** Community automatically. If you already have created an account at http://www.talendforge.org, click the **or connect on existing account** link to sign in.

    Be assured that any personal information you may provide to **Talend** will never be transmitted to third parties nor used for any purpose other than joining and logging in to the **Talend** Community and being informed of **Talend** latest updates.



This page will not appear again at Studio startup once you successfully connect to the **Talend** Community or if you click **Skip** too many times. You can show this page again from the Exchange preferences settings page. For more information on Exchange preferences, see *Talend Open Studio for Data Integration User Guide*.

A progress information bar displays. A welcome page displays from which you have direct links to user documentation, tutorials, **Talend** forum, **Talend** exchange and **Talend** latest news.

13. Click **Start now** to close the welcome page and open the main window of *Talend Open Studio for MDM*.

    Talend Open Studio for MDM main window opens on the **MDM** perspective. For further information, see section *Multi-perspective approach* and section *Switching between different perspectives*.

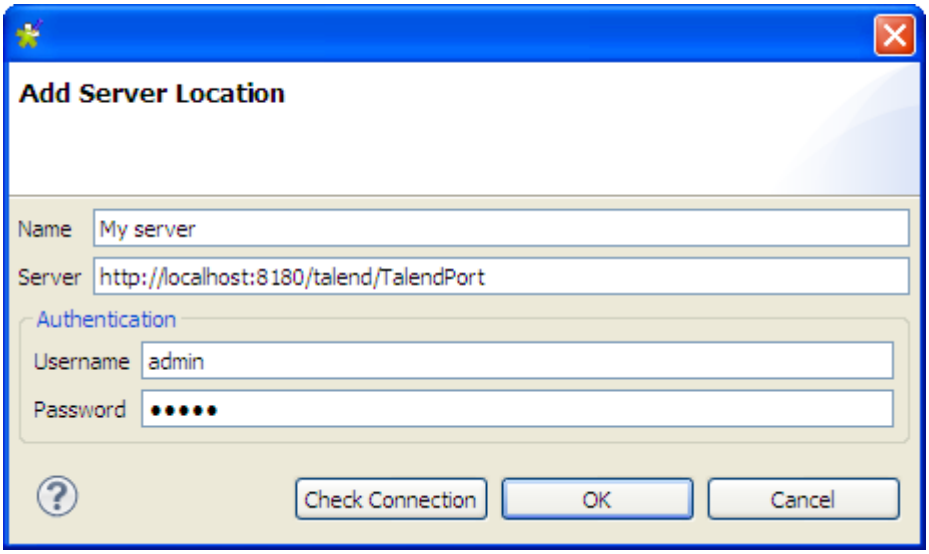
2.2.2. Connecting to the MDM server

Before you begin working with *Talend Open Studio for MDM*, you need to create a connection to at least one MDM server.

To create a connection to an MDM server, do the following:

1. In the Studio main window, in the Server Explorer panel, click the button to specify a new MDM server location.

   The **[Add Server Location]** dialog box appears.



   You can have more than one active connection to the MDM server using different authentication information.

2. In the **Name** field, enter a name or description for the MDM server to which you want to connect.

   You can then choose this alias to refer to the server from different points within the Studio. This simplifies deployment of MDM related integration and quality jobs.

3. In the **Server** field, type in the address of the MDM server to which you want to connect.

   The server address is structured as follows:

   ```
   -http://[server address]:[server port]/talend/TalendPort
   ```

4. Enter *admin* as the user name and *talend* as the password.

5. Click **OK** to close the dialog box and create the connection to the MDM server.

2.3. Working with the MDM Repository

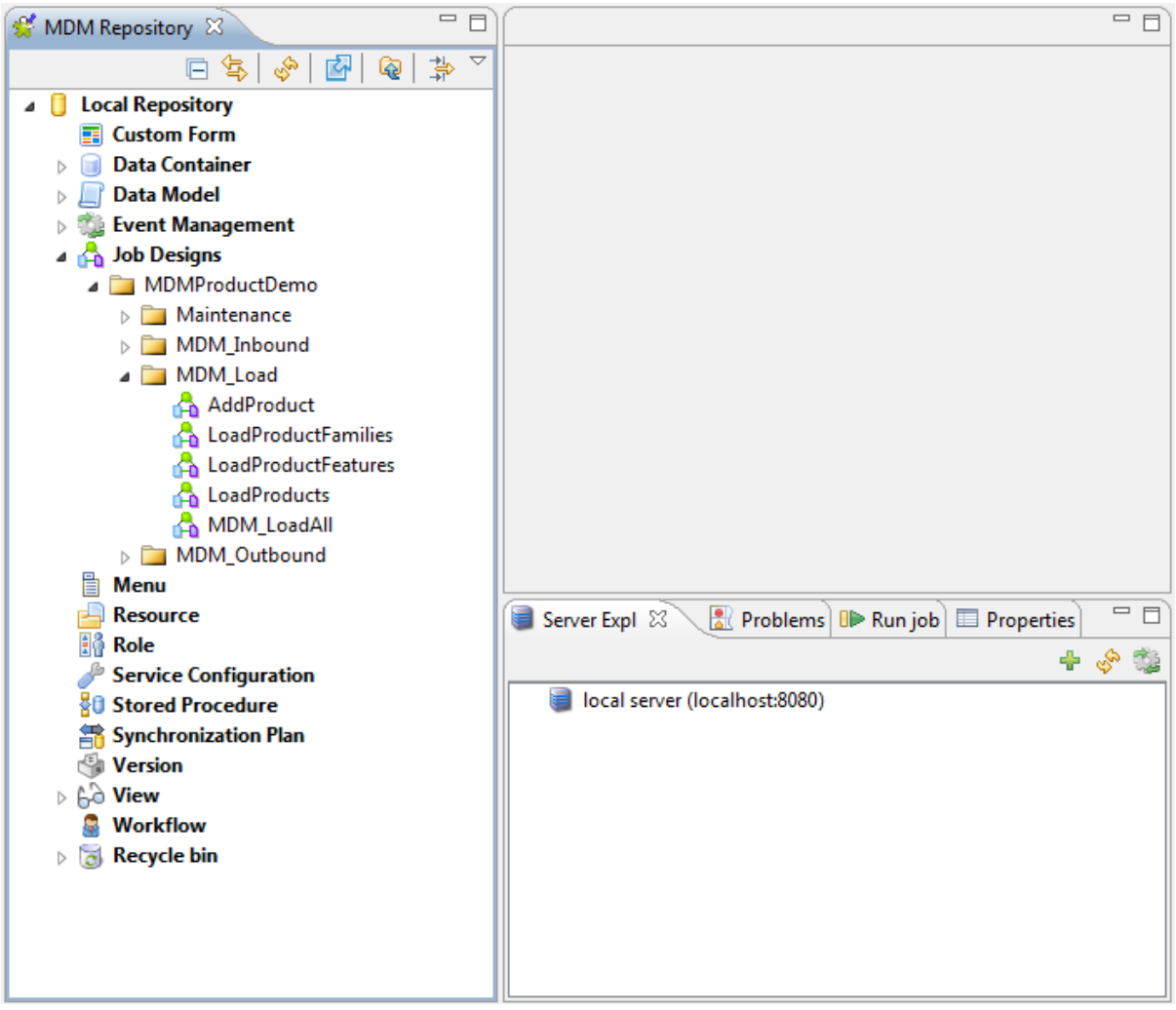
The MDM Repository enables you to work with your system objects, including data models, jobs, processes, and stored procedures, in a repository that is separate from the MDM Server database.

The MDM Repository can be stored locally or it can be stored on a remote server. If the repository is stored on a remote server, it can be shared among many users, and in this case the locking of objects is managed by Subversion.

When you make changes to the objects in the MDM Repository, you must deploy these objects to the MDM Server for your changes to be taken into account at runtime. In versions of *Talend Open Studio for MDM* prior to version 5.0, users were only able to edit objects directly on the MDM Server. For more information on setting a connection to an MDM Server, see section *Connecting to the MDM server*.

2.3.1. Displaying the MDM Repository view

To display the MDM Repository view, click **Window > Show View** to open the **[Show View]** dialog box, and then select **Talend > MDM Repository**.



When you open *Talend Open Studio for MDM* for the first time, the MDM Repository view is displayed by default.

2.3.2. Deploying system objects to the MDM Server

You must always deploy your system objects to the MDM Server for any changes you make to be taken into account at runtime.

You have a choice of how you deploy objects to the MDM Server.

- You can click the Update Server button  in the repository icon bar to deploy some or all of the objects that have changed in the MDM Repository since your last deployment.

  In the specific case of Jobs, only those Jobs that have previously been deployed using the **Deploy to...** option are proposed for update.

- You can manually select which objects you want to deploy from the **MDM Repository** tree view, either by specifying the destination server explicitly or by choosing to deploy to the last server used.

- You can edit the Preferences so that, when you save an object, it is automatically deployed to the last server used.

The steps involved in each of these different deployment methods are described in more detail in the following procedures. For further information on deploying Jobs, see section *How to deploy Jobs from the MDM perspective*

To deploy objects that have changed in the MDM Repository since your last deployment, do the following:

1. In the **MDM Repository** tree view, click the Update Server button  in the repository icon bar.

2. In the window that opens, select the server where you want to deploy the object or objects from the drop-down list, select the object or objects to deploy, and then click **OK** to perform the deployment.

To manually select the objects you want to deploy to the MDM Server, do the following:

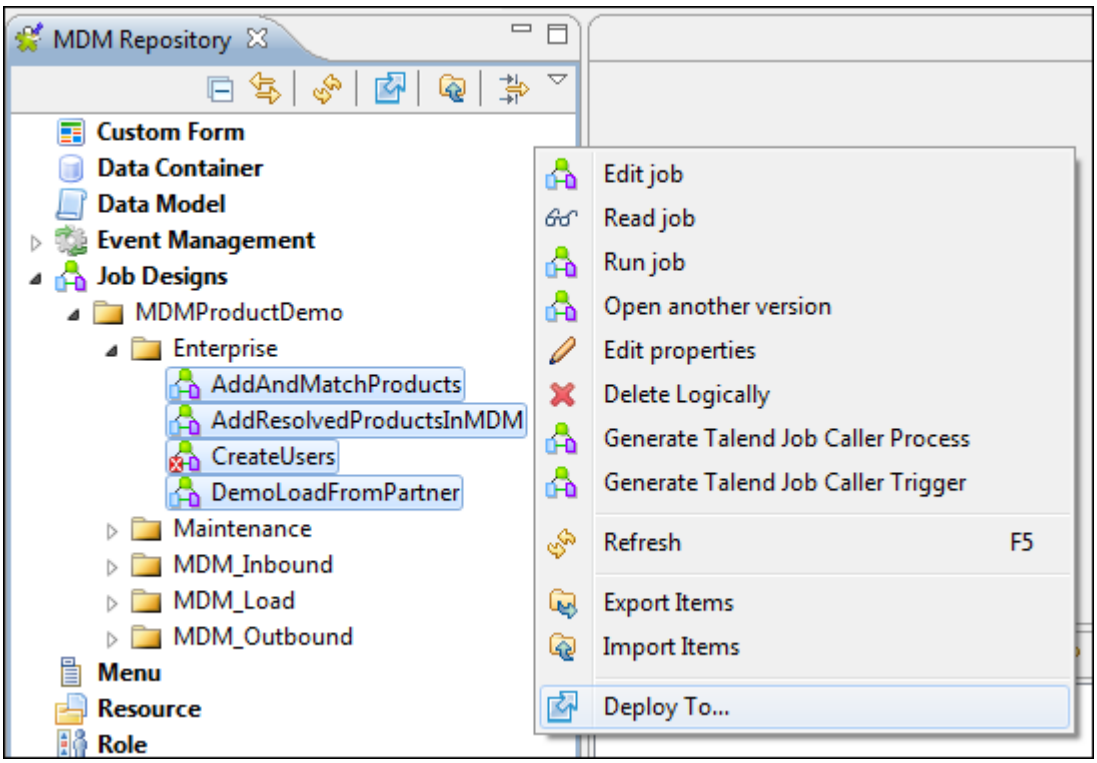
1. In the **MDM Repository** tree view, right-click the object you want to deploy, and then click one of the following deployment options:

   - **Deploy To...** to select the MDM Server where you want to deploy the object from the list of available servers

   - **Deploy To Last Server** to reuse the server you used for the most recent deployment action

   - **Update Server** to open the server update dialog box, as described in the previous procedure.

   The deployment options available may vary depending on the type of object you want to deploy, and if you are deploying the object for the first time or if it has already been deployed to a server.



If you want to deploy multiple objects at the same time, press and hold down the **Ctrl** key while you click each of the objects you want to deploy.

2. If appropriate, select the server where you want to deploy the objects in the **[Select server location definition]** window.

3. In the **[Deploy to Talend MDM]** window, specify the Settings for the deployment:

    1. Export type: Choose between Distributed (War) or Hosted (Zip).

        Deploy the Job as a Zip file if you want it to be preloaded and run directly in the MDM Server, which reduces latency. Deploy the Job as a War file if you want to embed it in a web service, for instance as a way of deploying the Jobs across different servers for the purposes of load balancing.

    2. Leave the Context scripts checkbox selected.

4. Click **Finish** to deploy your objects.

To automatically deploy objects to the MDM Server on saving, do the following:

1. Click **Windows > Preferences**.

    The **[Preferences]** window opens.

2. Expand **Talend**, and then click **MDM**.

3. Select the **Automatically deploy items onto last server when saving** checkbox, and then click **OK** to enable this option.

    If this option is enabled, whenever you save any changes you make to an object, the object is automatically deployed to the MDM Server.

2.3.3. Importing server objects from the MDM Server

You can import system objects, including data models, jobs, processes, and stored procedures, from the MDM Server database to the MDM Repository.

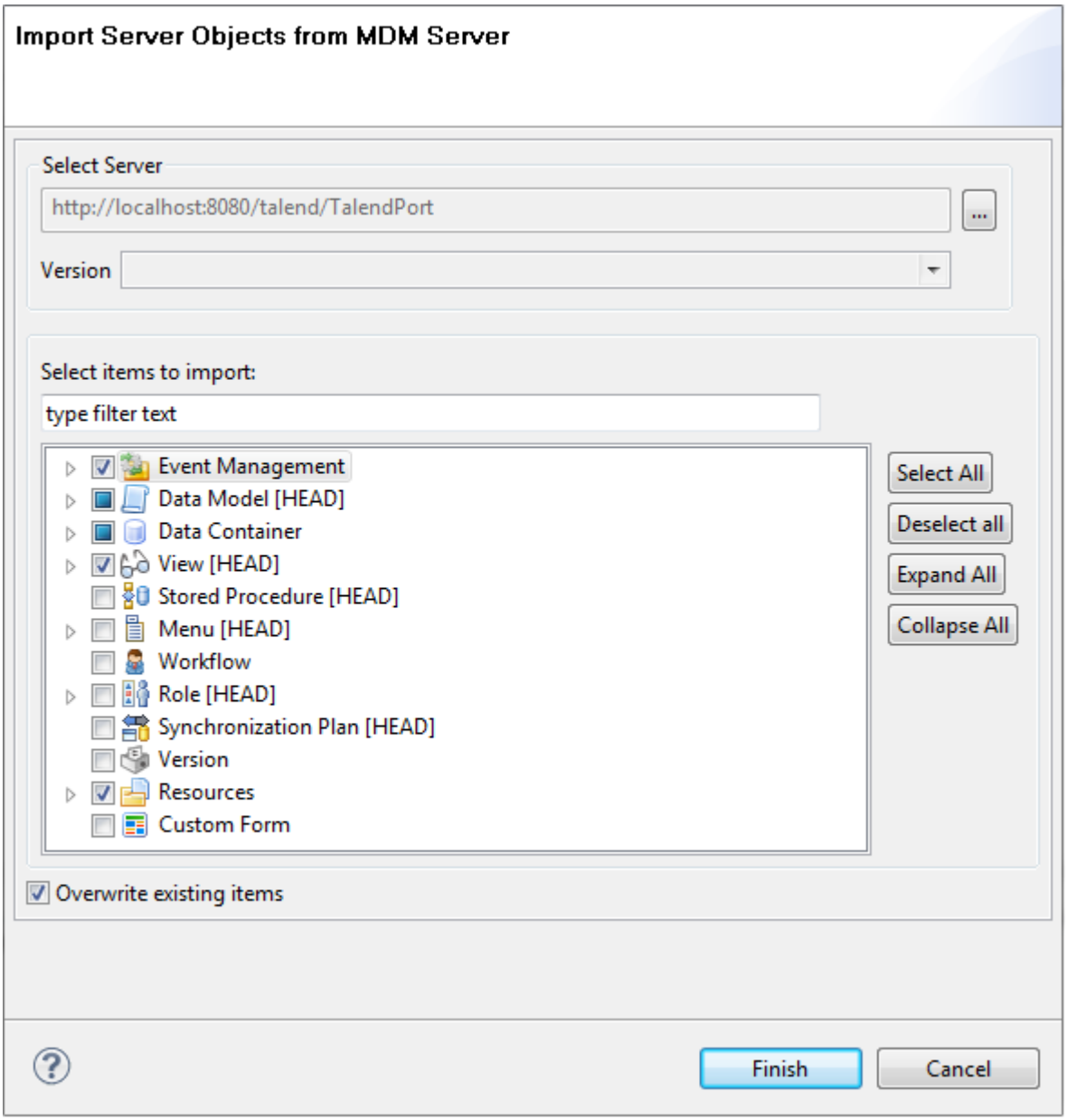
To import system objects from the MDM server, do the following:

1. In the **MDM Repository** tree view, right-click any object, and then click **Import Server Objects from MDM Server**.

   The **[Import Server Objects from MDM Server]** window opens.

2. Click the **[...]** button, select the server from which you want to import the objects in the **[Select a server location definition]** window that opens, and then click **OK**.

   The window is populated with a list of all available system objects.



3. Select the objects you want to import, and then click **Finish** to import the objects to your MDM Repository.

2.4. Migrating MDM projects

MDM master-records are stored in a database. Depending on your installation, this may be an XML or a relational database. You can migrate from one XML database version to another, or you can export your data from an XML database and reimport it in a relational database.

Note that the MDM Repository is always stored in an XML database.

Since not everything is stored in a database, you must also manually import and redeploy the following items:

- Jobs,
- pictures,
- web resources.

You must delete your web browser cache and cookies whenever you change the version, or the Studio edition (CE / EE) of Talend MDM. Unpredictable behavior or display errors will occur if you do not.

For detailed information about how to carry out a complete migration operation, see the installation documentation.

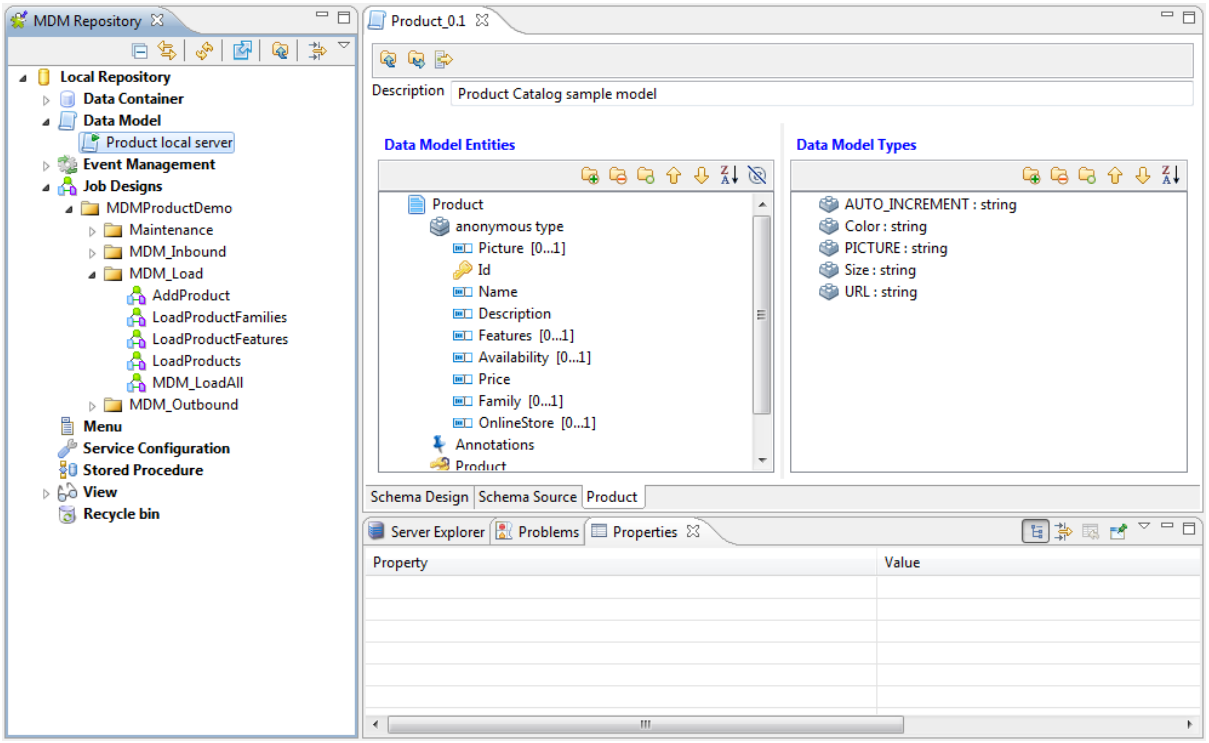
You can also move your data models, processes and triggers, etc. from the old server onto the new server through simple export/import operations available for each data object on the MDM server. For further information, see section *Managing data models* and section *Managing Processes*, section *Managing Triggers*.

2.5. Main window and navigation principles

Talend Open Studio for MDM is a user interface for designing and administrating Talend MDM.

In the *Talend Open Studio for MDM* main window, all data objects in the MDM Repository are listed in a tree view on the left hand side of the window.

Below is an example of a *Talend Open Studio for MDM* main window.



From the tree view:

- Double-click any of the MDM objects to open an editor on the right hand side of the main window. Here, you can view the object metadata or set the object parameters.
- Right-click any of the MDM objects or any item under an MDM object to display a contextual menu with multiple options.

For more information about *Talend Open Studio for MDM* Graphical User Interface, see appendix *Talend Open Studio for MDM management GUI*.

2.6. Multi-perspective approach

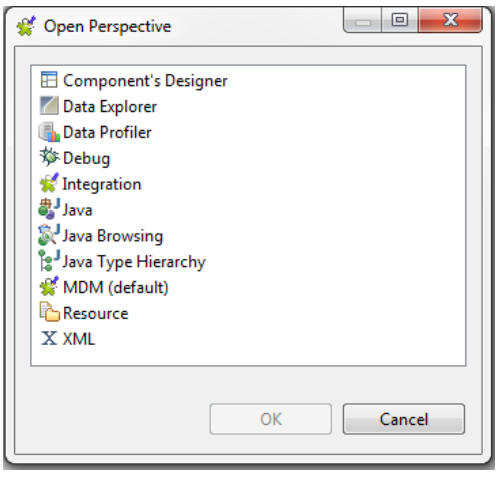
Talend Open Studio for MDM offers a comprehensive set of tools and functions for all its key capabilities including data integration, data profiling and master data management. These tools are ALL accessible in different perspectives from one single platform.

2.6.1. Switching between different perspectives

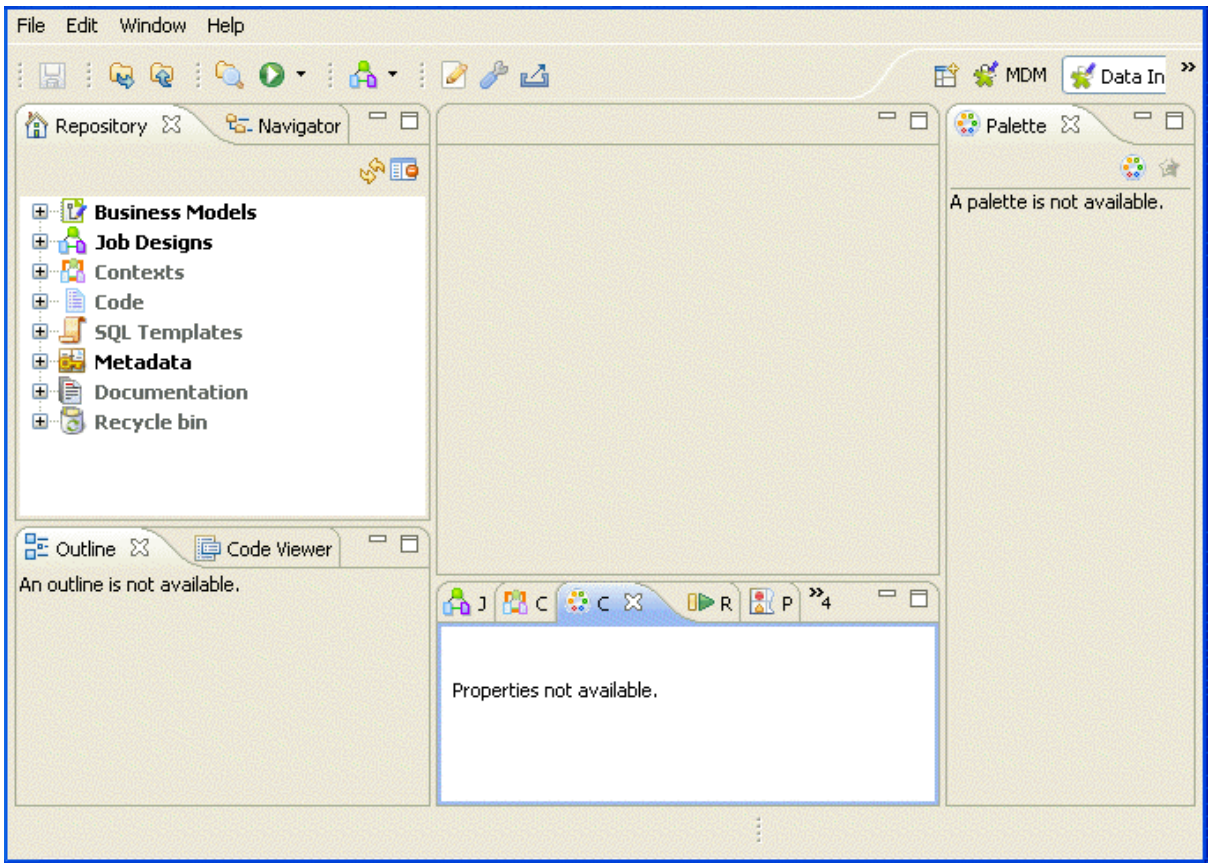
There are different ways to switch between the data integration, data profiling or master data management perspectives.

The options are as follows:

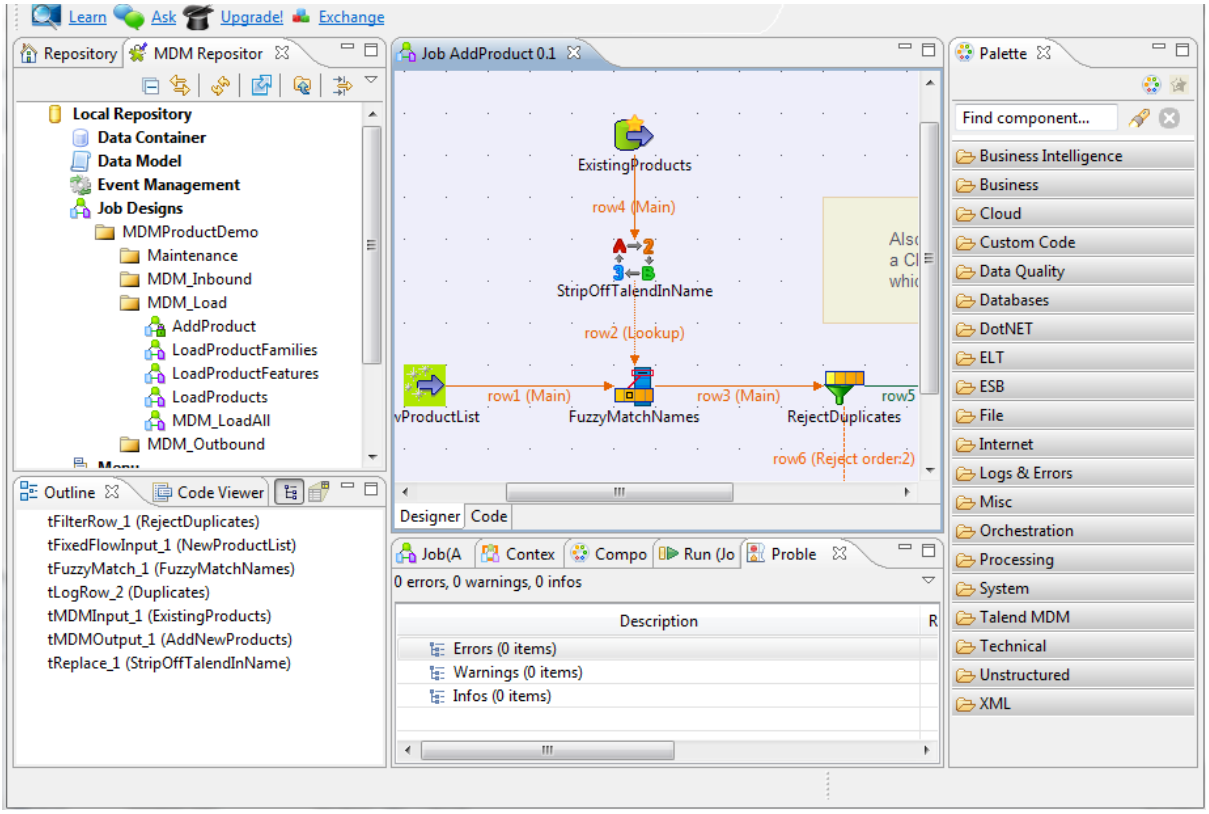
- In the upper right corner of the *Talend Open Studio for MDM* main window, click the icon and then select **Other...** to display the **[Open Perspective]** dialog box.



- Select the perspective you want to access, **Integration** for example, and then click **OK** to close the dialog box. The **Integration** perspective of your studio displays.

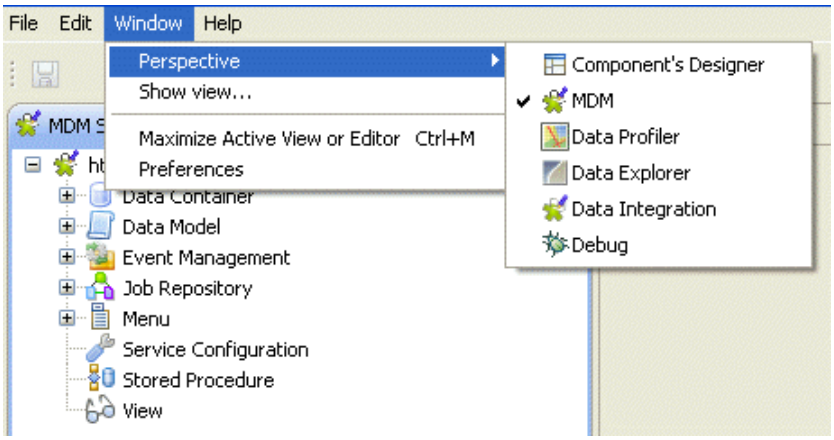


It is also possible, using the **Window - Show view...** combination, to show views from other perspectives in the open perspective.

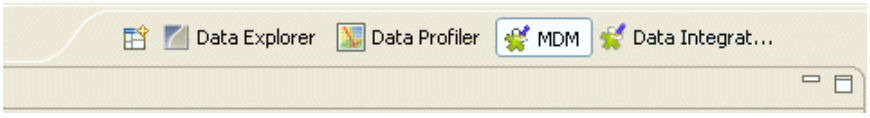


Alternatively, you may also switch between different perspectives by doing the following:

- On the menu bar, click **Window > Perspective** to display a list from which you can select the perspective to open.



Finally, you can also switch between available perspectives by simply clicking the corresponding icon on the upper right corner of the main window.



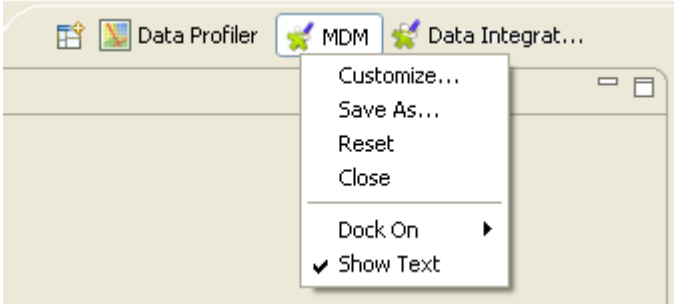
For more information on how to manage the display of these icons, see section *Managing quick access icons for different perspectives*.

2.6.2. Managing quick access icons for different perspectives

Every time you open a perspective in *Talend Open Studio for MDM*, a corresponding icon is docked on the upper right corner of the main window.

To manage the display of these quick access icons, do the following:

1. Right-click the icon of the open perspective to display a contextual menu.



2. Select the needed management option from the menu.

   The quick access icon or the corresponding perspective is changed accordingly.

The table below lists all available management options and their indications.

Option	Description
Customize	Opens a dialog box where you can

Option	Description
	-customize shortcuts in the current perspective -add command groups to the current perspective
Save As...	Changes the text that displays next to the icon as a title for the current perspective
Reset	Returns the current perspective to its default state
Close	Closes the current perspective
Dock on	Places the icons of the open perspectives: **Top right**: in the upper right corner of the Studio **Top left**: in the upper left corner of the Studio **Left**: to the left of the Studio
Show text	Displays/hides text next to the icon

Chapter 3. Setting data governance rules

This chapter provides the information you need to define the rules master data has to follow. These rules include how to partition data; what validity models are required for data; who can read, create, update, and delete data and what services to be taken on data, to name a few.

Before starting any of the MDM management procedures included in this chapter, you need to be familiar with the *Talend Open Studio for MDM* Graphical User Interface (GUI). For more information, see chapter *Talend MDM: Concepts & Principles* and appendix *Talend Open Studio for MDM management GUI*.

3.1. MDM working principles

From *Talend Open Studio for MDM*, you can define the rules that master data has to follow.

A typical sequence of setting governance rules in *Talend Open Studio for MDM* involves the following steps:

1. Create a data model containing one or multiple business entities. A business entity can be a supplier, a country, a customer, etc. For more information, see section *Setting up a data model*.
2. Define a data container in the MDM Hub where the master data is to be persist. For more information, see section *Creating a data container*.
3. Design one or more Processes that perform transformations, data validation and other tasks. Processes usually call a sequence of available plug-ins to project data from the source system to a defined data container (MDM Hub). During projection, the data is transformed, cross-referenced if needed and validated against the entity model. For more information on Processes, see section *How to create a Process from scratch*.
4. Design one or more Triggers, rules for transforming data, that enable the event-based Processes to be undertaken on specific data records. For more information, see section *Creating a Trigger*.

After projection, the MDM Hub holds valid data (i.e. model compliant data) that a data steward or a business user can query and/or extract via *Talend MDM Web User Interface*. For more information, see the *Talend MDM Web User Interface User Guide*.

Before a given business user is able to extract and query valid master data via the web-based User Interface, you need to create one or more Views in *Talend Open Studio for MDM* that specify:

- which records of an entity a business user has the right to search,
- which records of an entity a business user has the right to view,
- optionally which conditions should the content meet to be delivered as the result of a search (content filters).

For more information on Views, see section *Creating a View*.

3.2. Data Models

Data models are the central component of *Talend Open Studio for MDM*. They only define the master entities you want to manage while master data records themselves are stored in data containers. For further information about data containers, see section *Data Containers*.

Data models contain entities which are master data objects (or business entities) such as a *Person*, a *Product*, an *Organization*, etc. Each entity is defined by its attributes which are the entity characteristics, for example, for the *Person* entity, the attributes could be *name*, *date of birth*, *address*, etc.

When creating a data model from *Talend Open Studio for MDM*, you compartmentalize records and data to create the most efficient data model possible. In addition, you can define the attributes,user access rights and relationships of entities mastered by the MDM Hub.

Authorized users can create, import/export, edit, copy/paste and delete data models.

The following sections detail the procedural steps to carry out each of the above management options.

3.2.1. Setting up a data model

If you have the appropriate user authorization, you can create one or multiple data models for any domain and store them in the MDM Hub.

Once created, an authorized user of *Talend MDM Web User Interface* can validate hub data against these data models. For more information, see the *Talend MDM Web User Interface User Guide*.

A few steps are necessary to complete the creation of a data model:

1. Create business entities in the model.
2. Adding attributes in each of the created business entities.
3. Set annotations to entities or attributes regarding display issues, foreign key, etc.

You can also define reusable types for any data model. These reusable types are sets of attributes that can be used in one or more entities of the data model. Using reusable types is a way of factoring the data model and managing the impact of changes since any modification on a reusable type will have an impact on every entity that uses this type.

Once the data model is created, you can easily create Views over one or multiple business entities held in this data model. Creating a View on a business entity allows a business user inside *Talend MDM Web User Interface* to visualize specific records in this entity according to the defined criteria.

Once the data model is created, you can right-click any of the entities held in this data model and automatically generate a default View for the selected entity.

For more information, see section *Views*.

Once the creation of the data model is complete, you may need to design one or more Processes and create a View to define search parameters on business entities and data records in the created data model.

For more information on Processes and Triggers, see section *Processes* and section *Triggers* respectively.

For more information on Views, see section *Views*.

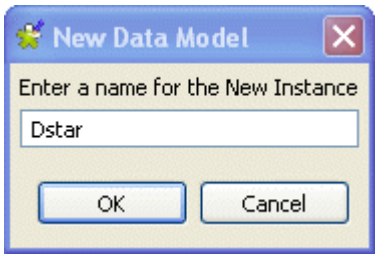
3.2.1.1. How to create a data model

Prerequisite(s): You have already connected to the MDM server from *Talend Open Studio for MDM*. .

To create a new data model :

1. In the **MDM Repository** tree view, right-click **Data Models** and select **New** from the contextual menu.

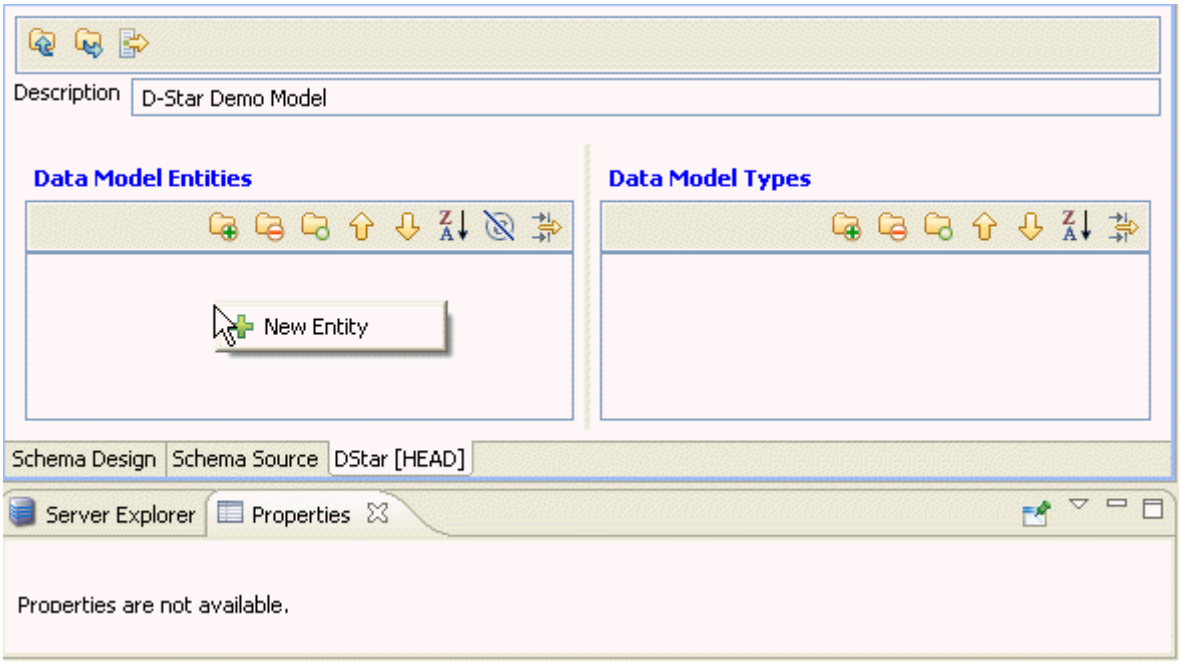
   A dialog box displays.



2. Enter a name for the new model and click **OK** to close the dialog box.

   An editor displays in the Studio workspace where you can set parameters for the new model.

No spaces are allowed in the data model name. The OK button in the dialog box will be unavailable as long as there is a space in the instance name.



You can define your business entities, attributes, annotations, etc. in the upper part of the editor; while the lower part will display the properties of any of the elements you select in the editor.

The editor also enables you to switch to the XML schema design and XML schema source through clicking the **Schema Design** and **Schema Source** tabs respectively.

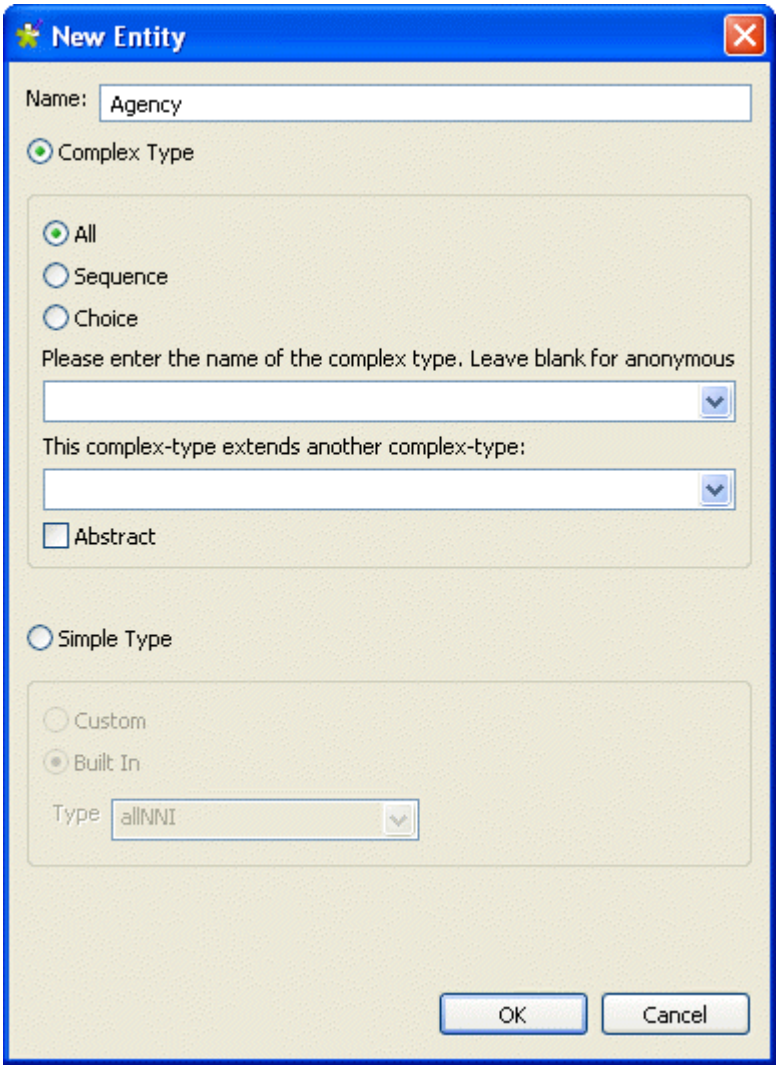
You can create the data model directly in the schema source, XML document, if you click the **Schema Source** tab at the bottom of the editor.

3.2.1.2. How to create business entities in the data model

Prerequisite(s): You have already created a data model. For further information, see section *How to create a data model*.

To create business entities in the model:

1. Right-click in the **Data Model Entities** panel and select **New Entity** to open the **[New Entity]** dialog box.



2. In the **Name** field, enter a name for the new entity.
3. Select the **Complex type** option if you want to define a complete structure, such as an address. The steps that follow differ according to the selected type.

   You cannot create two complex types where one is a base type and the other is a type that derives from the base type, and then create entities based on these types.

   Instead, you should first define a higher level entity which implements its own complex type (for example, *Media*) to define all common elements for the entity. Next, define another complex-type element (for example, *MediaSpecifics*) which can extend *Media* and then be extended in turn. In this example, *MediaSpecifics* could be extended with *Books* to add elements that are specfic to books and *Audio* to add elements that are specific to audio media.

   At this point you can create a **Browse_item_Media#Books** View that filters (*where* clause) on **@xsi:type - equals - Books** and a **Browse_item_Media#Audio** View that filters on **@xsi:type - equals - Audio**. The standard **Browse_item_Media** View would return all media.

   For more information on creating Views, see section *Views*.
4. Select the **Simple type** option if you want to define a single element type such as a phone number, an email, etc. The steps that follow differ according to the selected type.

   You can still change the entity type after you create it if you right-click the entity in the **Data Model Entities** panel and select **Change to a Complex Type** or **Change to a Simple Type**.
5. If you select **Complex type**, select the group type among:

   -**All**: to list the elements in any sequence,

-**Sequence**: to list the elements according to the defined sequence,

-**Choice**: to have a choice on the elements.

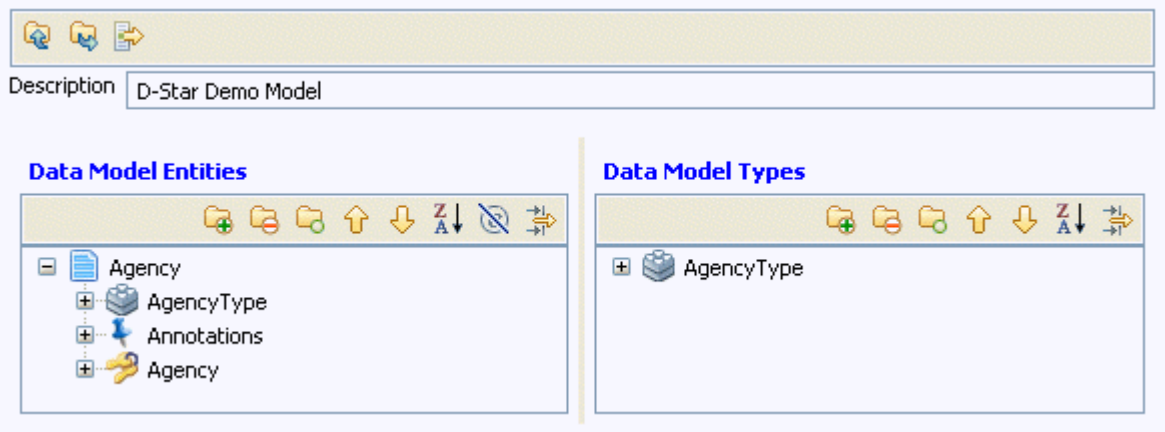
6. Enter a name for the complex type in the corresponding field, if you want to create a reusable type of this entity.

   Reusable types are sets of attributes that can be used in one or more entities.

   The complex type list gives you access to all comples types you create in your data model. When defining a new entity, you can select from this list the complex type from which you want to inherit elements in the new business entity. Each complex type you define can serve as the basis for another complex type. For further information, see section *Data model inheritance and polymorphism*.

7. Click **OK** to validate and close the dialog box.

   The created business entity is listed in the **Data Model Entities** panel with a by-default record called **subelement**, and the complex type, if any, displays in the **Data Model Types** panel.



   Each time you create a new business entity, a default Primary Key record, called **subelement**, and a Unique Key record that holds the entity's name are automatically created.

   A Primary Key can be an integer but a Foreign Key must always be a string. The server surrounds Foreign Keys with square brackets to support compound keys.

8. Repeat the above steps to create as many business entities as you need in your data model.

   All business entities you create are listed in the **Data Model Entities** panel.

To finish defining the business entities in your data model, you must create attributes in these business entities. These attributes represent the characteristics of each of the business entities. For further information, see section *How to add attributes to the business entity*.

Displaying user data differently based on locale

You can also set up the elements you create so that the *Talend MDM Web User Interface* displays the data for a record differently depending on the user's locale.

Prerequisite(s): You have already created business entities in the data model. For further information, see section *How to create business entities in the data model*.

1. Right-click the element, and then click **Change to a Simple Type** in the contextual menu.

   The **[Make Simple Type]** dialog box opens.

2. Select the **Custom** radio button and set the **Type** as **MULTI_LINGUAL** by selecting this option in the drop-down list, and then click **OK**.

Users of *Talend MDM Web User Interface* will now be able to create localized instances of the data for this field. For further information, see the *Talend MDM Web User Interface User Guide*.

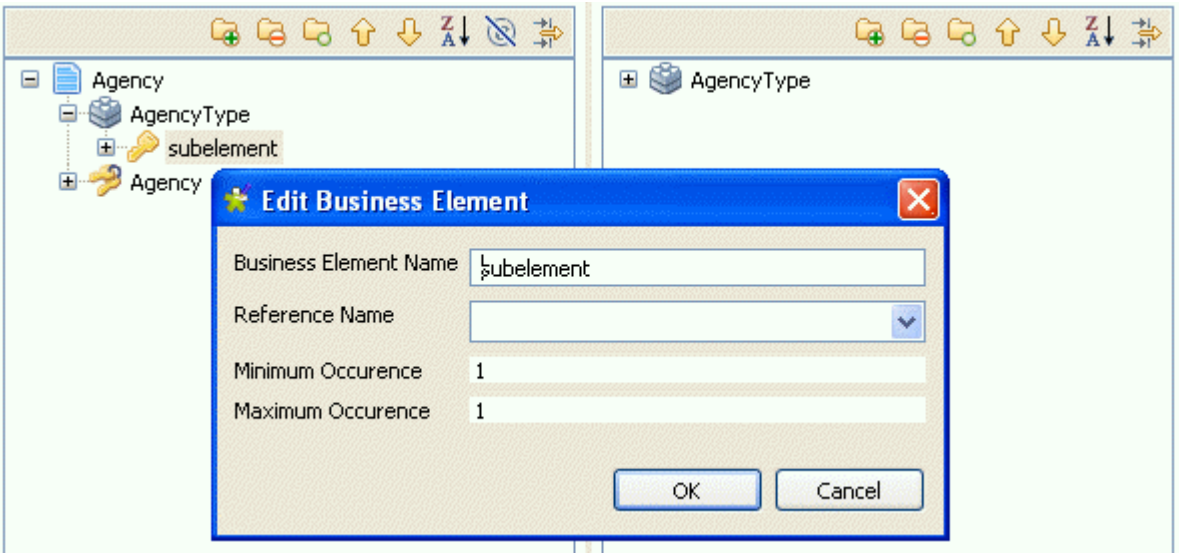
3.2.1.3. How to add attributes to the business entity

Prerequisite(s): You have already created business entities in the data model. For further information, see section *How to create business entities in the data model*.

To add attributes to the business entity, do the following:

1. Expand the business entity you created and the group type in succession and right-click **subelement** then select **Edit Element** from the contextual menu.

   A dialog box displays.



2. Change the name to *Id* and set the minimum and maximum occurrences to *1* and then click **OK** to close the dialog box.

3. Right-click *Id*, select **Add Element (after)** from the contextual menu, and then select **Add string Element**.

   A dialog box displays.

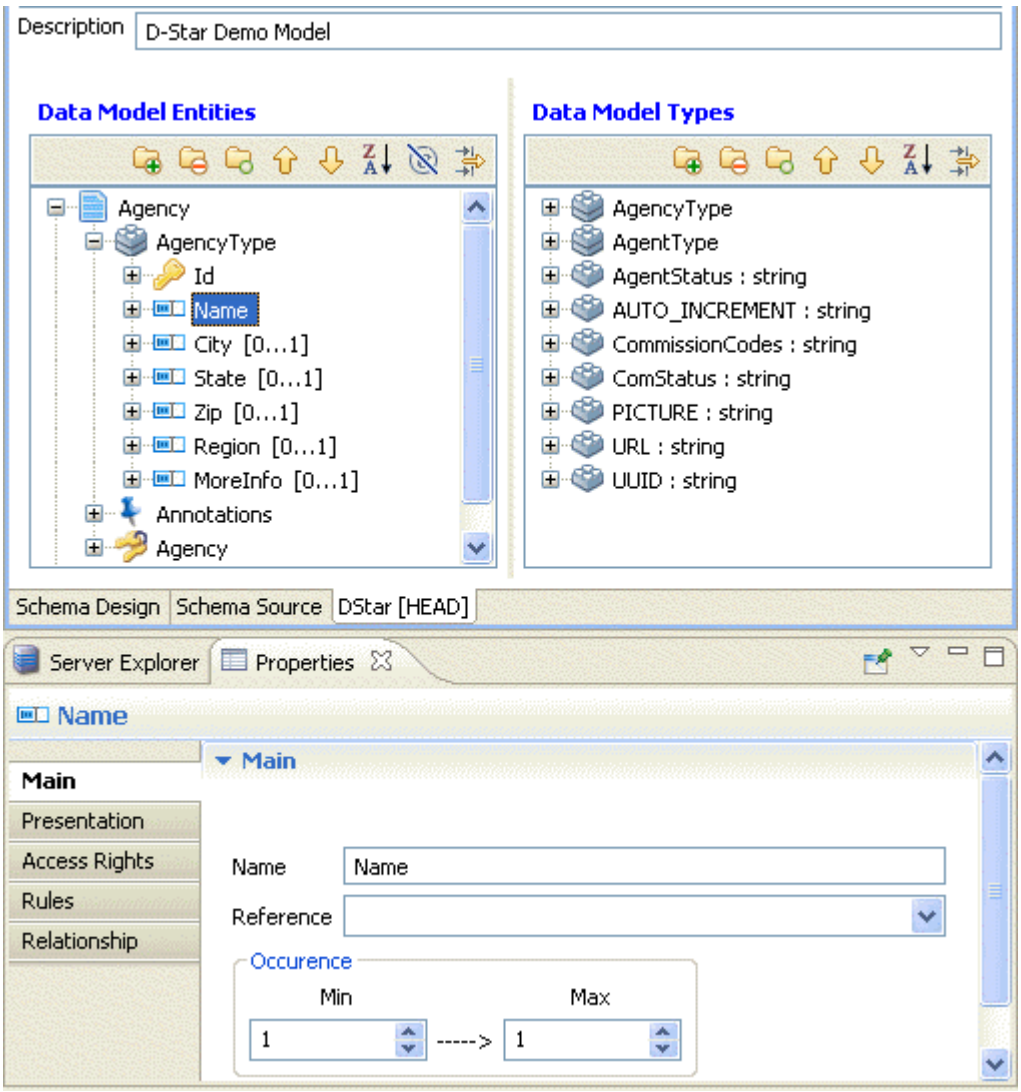
4. Enter a name and a minimum and maximum number of occurrences for the new attribute in the corresponding fields.

   Talend Open Studio for MDM supports the definition of multi-occurrence attributes. You can set the **Maximum Occurrence** value to a specific number or leave it blank, which means the maximum number of occurrences is not bounded: in this case, the cardinality will be *[1..many]* or *[0..many]*.

5. Click **OK** to validate your changes and close the dialog box.

6. Follow the same steps to create as many attributes as needed in the business entity.

   All attributes you create are listed under the business entity in the **Data Model Entities** panel, and all complex types are listed in the **Data Model Types** panel.

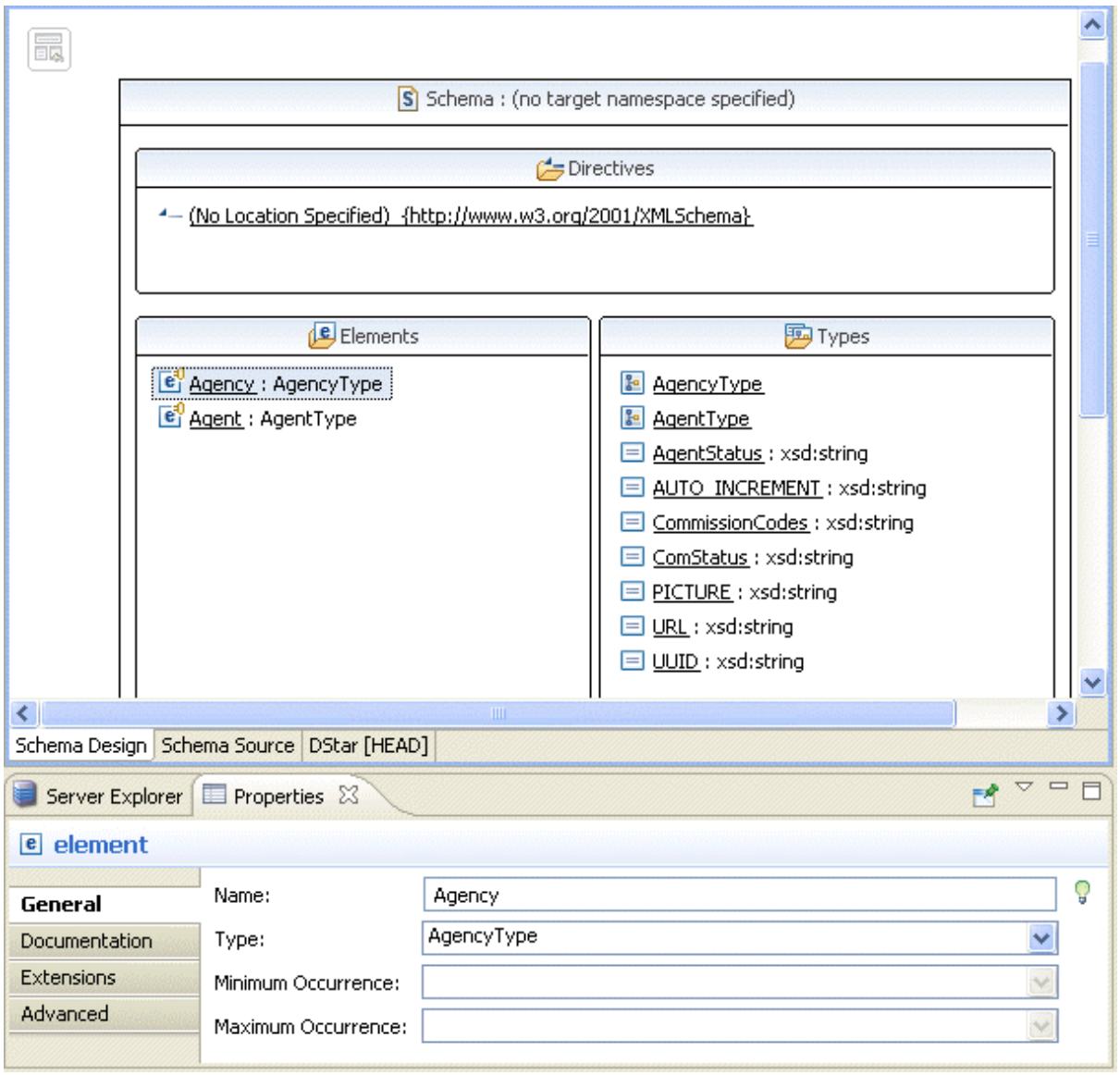


The data model editor includes a **Properties** view that displays all the properties of the selected entity/attribute in the data model. You can use this detail view to edit dynamically any of the entity or attribute listed in the data model. For further information, see section *How to use the Properties view in the data model editor*.

7. If required, right-click any of the listed element and select **Edit element** to open a dialog box where you can change the parameters for the selected element.

You can switch to the XML schema design or XML schema source through clicking the **Schema Design** and **Schema Source** tabs respectively.

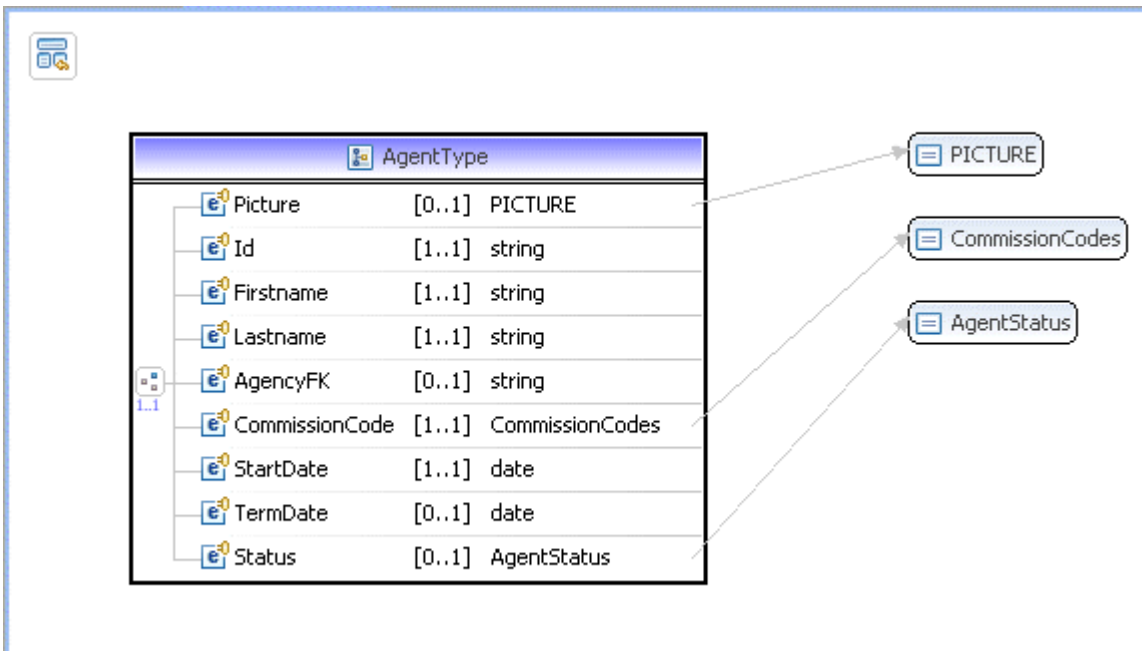
Below is an example of an XML schema design:



The XML schema design presents the data model in a graphical format. This format shows related entities in a graphical format and also implements an XSD editor.

1. Double-click any of the items in the **Types** panel in the XML schema design.

   A graphical format of the selected data model type displays.



2. Click the  icon in the upper left corner of the editor to go back to the XML schema index view.

Because Talend MDM uses XSD as the language to support data modeling and to make it easier to import existing XML schemas, you can use any of the XSD capabilities in any of the XML schema views. Some management options are accessible from the XML schema design or XML schema source if you right-click any of the entities/attributes listed in the view, or if you right-click in the view itself.

You can also select any of the elements or types in the XML schema design to display its properties in the **Properties** view. Then you can modify any of the listed properties in the view and reflect the modification directly in the XML schema design.

Below is an example of an XML schema source:

```
1  <xsd:schema attributeFormDefault="unqualified" blockDefault="" elem
2    <xsd:import namespace="http://www.w3.org/2001/XMLSchema"/>
3    <xsd:element abstract="false" name="Agent" nillable="false" type=
4      <xsd:annotation>
5        <xsd:appinfo source="X_Label_EN">D* Agent</xsd:appinfo>
6        <xsd:appinfo source="X_Label_FR">Agent D*</xsd:appinfo>
7        <xsd:appinfo source="X_Schematron">&lt;pattern name="Check id
8
9
10
11     <xsd:appinfo source="X_Write">General_Manager</xsd:appinfo>
12 <xsd:appinfo source="X_Write">Manager_MWest</xsd:appinfo>
13 <xsd:appinfo source="X_Write">System_Admin</xsd:appinfo>
14 </xsd:annotation>
```

Schema Design | Schema Source | DStar [HEAD]

You can carry out any modifications on the business entities or attributes directly in the XML schema source. This enables you to take advantage of such features as code-assist and syntax highlighting.

After defining business entities and attributes in your data model, you must set annotations to these entities or attributes regarding display issues, foreign keys, etc. The following sections explain this in detail.

3.2.1.4. How to set up annotations to business entities

Prerequisite(s): You have already created business entities and attributes in the data model. For further information, see section *How to create business entities in the data model* and section *How to add attributes to the business entity*.

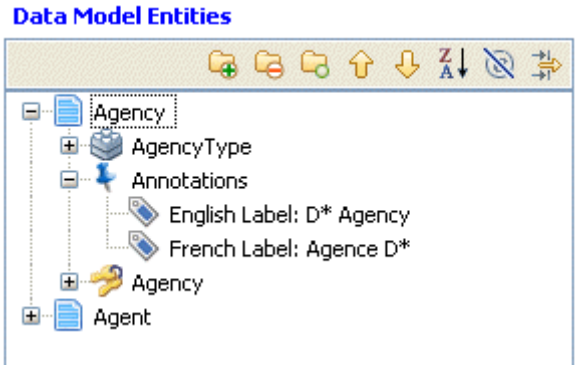
The data model editor makes it very easy to add specific annotations to any business entity in the data model.

Through annotations, you can define display issues related to language specific labels, foreign and primary keys, multilingual description etc.

The below table describes the annotations you can add to business entities.

Option	Description
Add Key	To add a unique or a simple key to the selected entity.
Set the Labels	To have labels in different languages for any business entity in the data model. You can automatically generate the labels for new entities in any of the listed languages. To generate a label automatically, select a language from the **Language** list in the upper right corner of the data model editor and then click the plus button.
Set the Descriptions	To have multilingual descriptions for different business entities in the data model.
Set Lookup Fields	To look up fields in the source system through a Process attached to a specific view in order to enrich data on the fly. For example, when an interaction is done on a business entity that is not persisted or not totally persisted in the MDM Hub. For further information, see section *Running the view result through a Process (registry style lookup)*.For example,
Set the Primary Key Infos	To display specific data related to the business entity when accessing any data record in the business entity through *Talend MDM Web User Interface*.

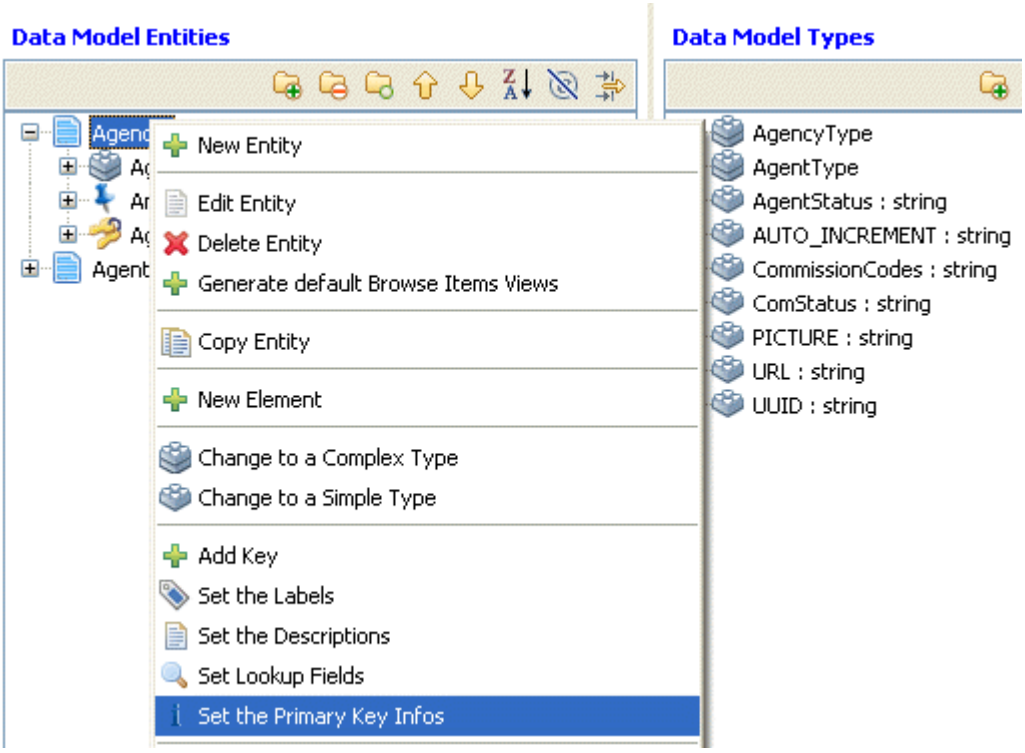
When you add any of the above annotation to a business entity, an **Annotations** node is added to the business entity in the **Data Model Entities** panel. This node lists all the annotations added to the business entity.



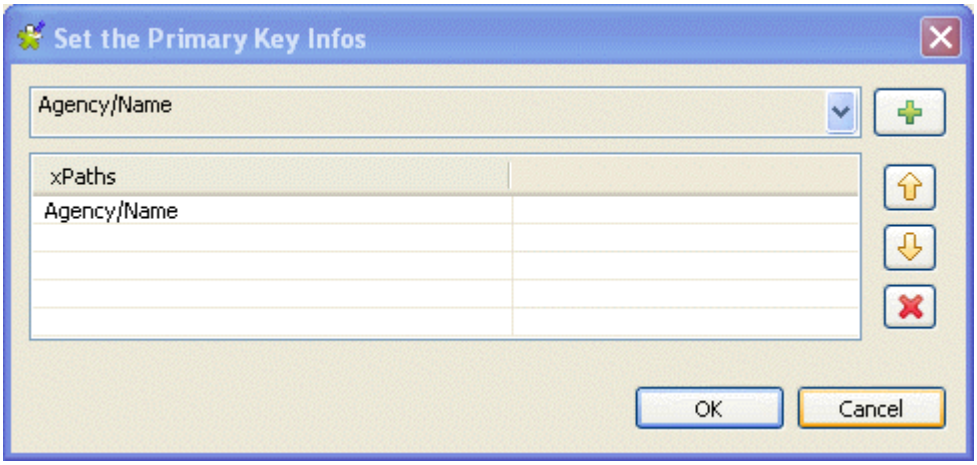
As an example of how to set up an annotation to a business entity, the below procedure gives detail information about how to add the **Set the Primary Key Infos** annotation.

1. Right-click the business entity to which you want to add the annotation.

2. In the contextual menu, select the annotation you want to apply to the selected business entity, **Set the Primary Key Infos** in this example.



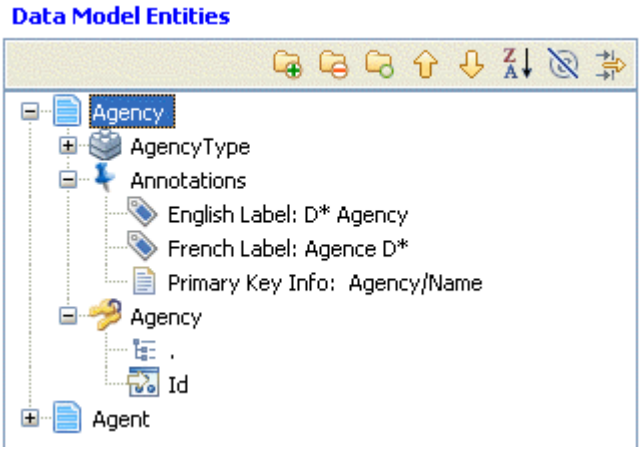
A dialog box corresponding to the annotation option you have selected displays.



3. Click  and select from the list the primary key information you want to display in *Talend MDM Web User Interface* when a user access any of the records of the selected business entity. In this example, you want each agency record to be headed with the agency name.

4. Click the  button in the upper right corner to add the selected xpath to the **xpaths** list and then click **OK** to validate your changes and close the dialog box.

   The set annotation displays under the selected business entity.



5. Click the save icon on the toolbar or press **Ctrl + S** on your keyboard to save your modifications.

When a business user access the data model holding this entity through *Talend MDM Web User Interface* and browse for any of the data records in this entity, the detailed view of the data record will have the agency name as its heading as defined in the above procedure.

3.2.1.5. How to set up annotations to attributes

Prerequisite(s): You have already created business entities and attributes in the data model. For further information, see section *How to create business entities in the data model* and section *How to add attributes to the business entity*.

The data model editor makes it very easy to add specific annotations to any attribute within the business entity.

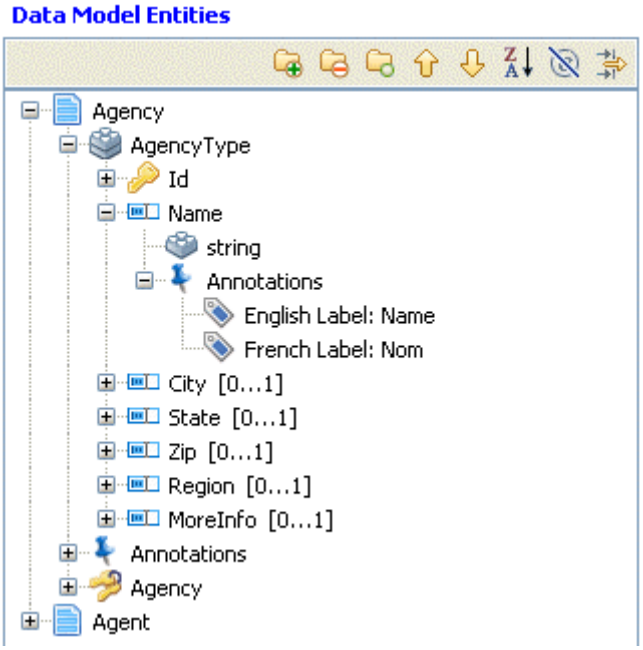
Through annotations, you candefine display issues related to language specific labels, foreign and primary keys, multilingual description etc.

The below table describes the annotations you can add to attributes within business entities.

Option	Description
Set the Descriptions	To have multilingual descriptions for different attributes in business entities.
Set the Labels	To have labels in different languages for any attribute within a business entity in the data model. You can automatically generate the labels for new attributes in any of the listed languages. To generate a label automatically, select a language from the **Language** list in the upper right corner of the data model editor and then click the plus button.
Set the Foreign Key	To link one attribute to another attribute or to a business entity. For further information, see section *How to add a foreign key: linking entities together*. A Primary Key can be an *integer* but a Foreign key must always be a *string*. The server surrounds Foreign Keys with square brackets to support compound keys.
Set the Foreign Key Filter	To filter foreign keys by values extracted from the current record through using the standard xpath predicate of the relevant element. For further information, see section *How to add a foreign key filter*.
Set the Foreign Key Infos	To display the data related to the foreign key.
Set the Facet Message	To set multilingual facet error messages.
Set the display format	To decide the format according to which you want to display/insert, in a specific language, dates or numbers in master data records in *Talend MDM Web User Interface*. To define the date/number format, you must use a syntax that is based on the *java.util.Formatter* class. For further information on this class, see *http://download.oracle.com/javase/6/docs/api/java/util/Formatter.html*.

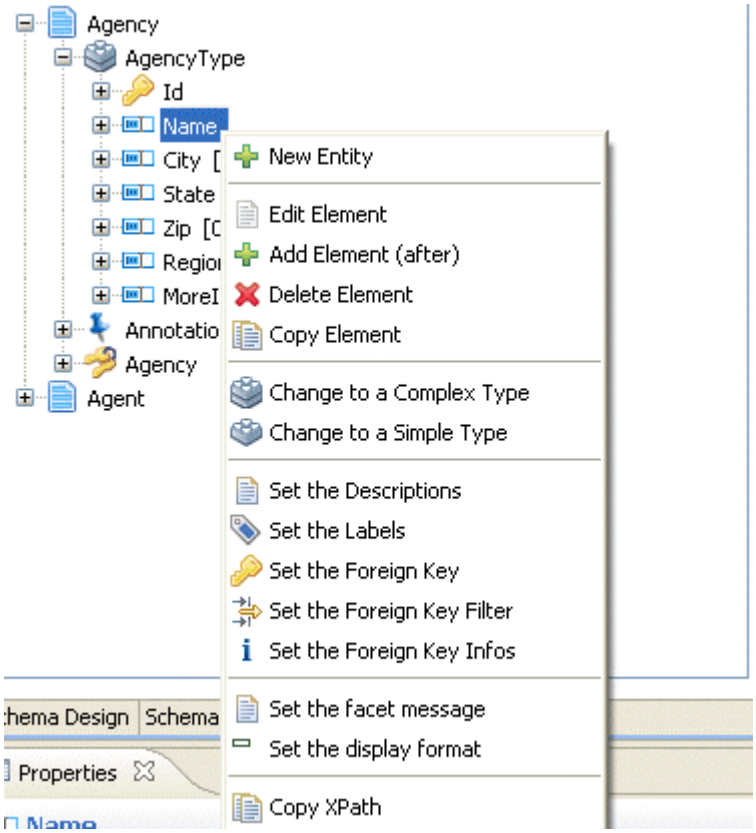
Option	Description
	For an example on how to set the display format, see section *How to set the display format of dates and numbers*.

When you add any of the above annotation to an attribute in a business entity, an **Annotations** node is added to the attribute in the **Data Model Entities** panel. This node lists all the annotations added to the selected attribute.



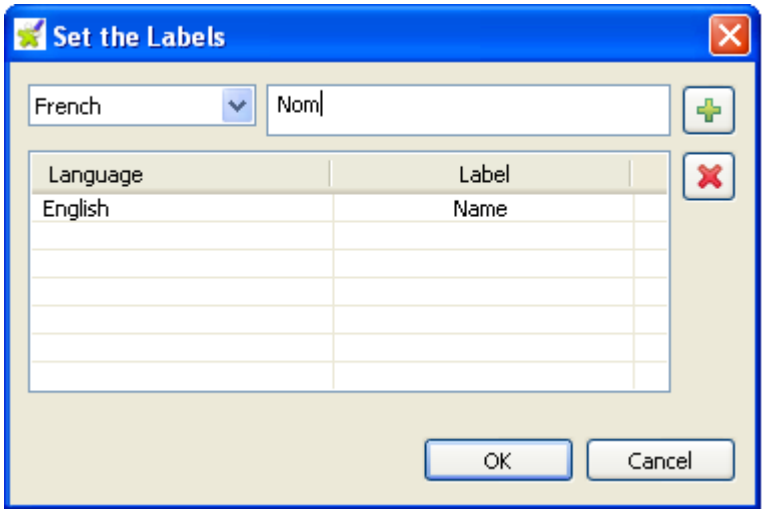
To set up an annotation to any of the attributes you create in a business entity, do the following:

1. In the **Data Model Entities** panel, right-click the attribute to which you want to add the annotation.
2. In the contextual menu, select the annotation you want to apply to the selected attribute.



A dialog box corresponding to the annotation option you have selected displays.

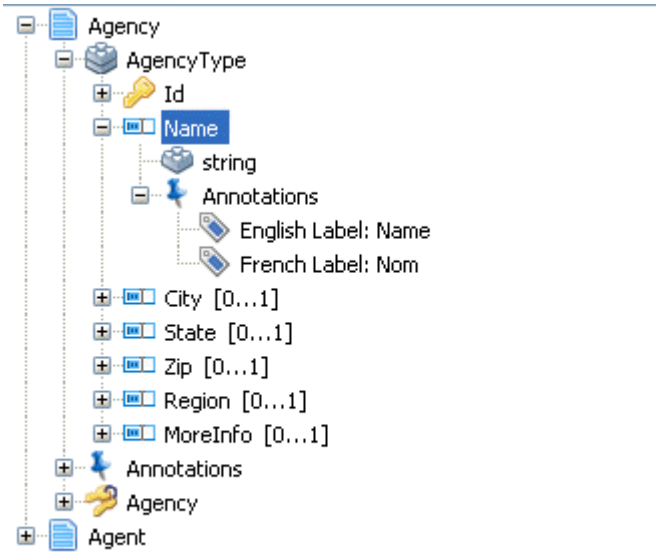
For example, if you select **Set the Labels** to have labels in different languages for a selected attribute, the following dialog box displays.



3. From the list to the left, select a language and then enter a label in the field to the right.

4. Click the  icon in the upper right corner to add the defined label to the **Language/Label** list and then click **OK** to validate your changes and close the dialog box.

   The set annotation displays under the selected attribute.



5. Click the save icon on the toolbar or press **Ctrl + S** on your keyboard to save your modifications.

You can automatically generate the labels for new attributes in any of the listed languages. To generate a label automatically, select a language from the **Language** list in the upper right corner of the data model editor and then click the plus button.

3.2.1.6. How to add business rules

Talend Open Studio for MDM enables you to enrich data models by adding simple business rules.

How to add simple rules

In any business entity, you can set rules against which you validate the entity attributes (minimum and maximum length, list of values, etc.).

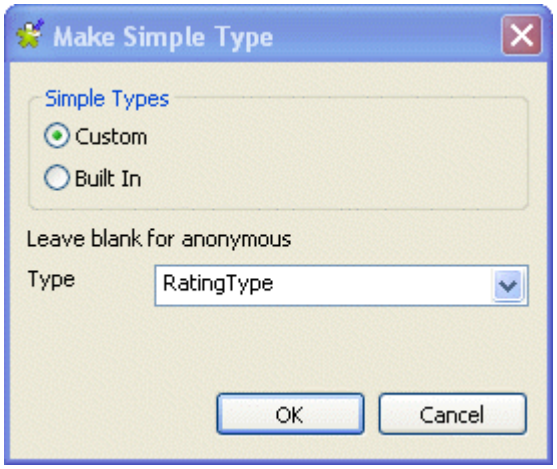
Prerequisite(s): You have already created a data model and the business entities and attributes in the data model. For further information, see section *How to create a data model*, section *How to create business entities in the data model* and section *How to add attributes to the business entity*.

Consider as an example that your data model holds the following entities: *Agency*, *Agent* and *Region*. You have an attribute called *Rating* in the *Agency* entity. This attribute will be used as a reference to the agency rating (list of values) in a workflow process.

To add a simple rule to a business entity, do the following:

1. In the data model editor, right-click the *Rating* attribute and select **Change to a Simple Type**.

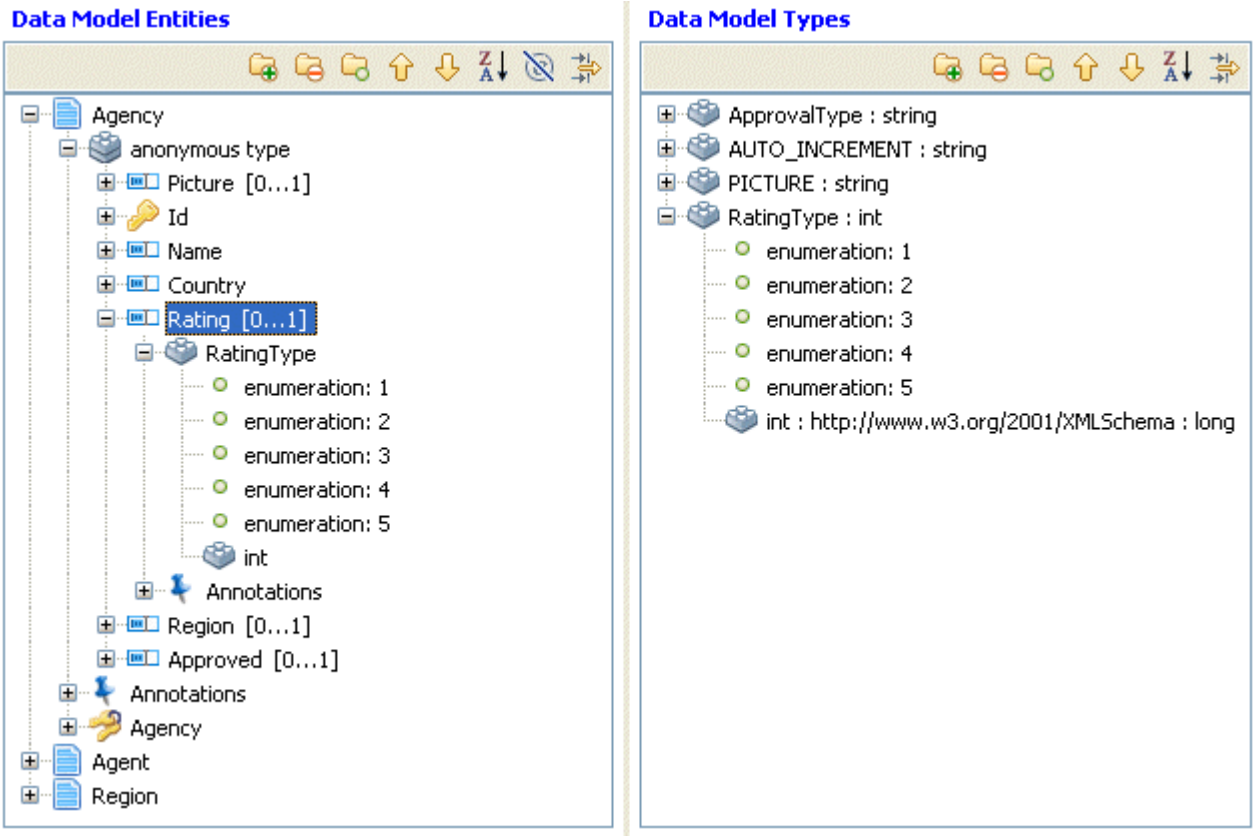
   A dialog box displays.



2. Select the **Custom** option and then enter *RatingType* in the **Type** field.

3. Click **OK** to validate your changes and close the dialog box.

   The *RatingType* simple rule displays under the selected attribute in the data model editor and a reusable type of the same simple rule is created in the **Data Model Types** list.



4. Right-click the simple rule in the **Data Model Types** panel and select **Edit Enumeration**.

   A dialog box displays.

5. In the **New Item** field, enter the first agency rating value and then click the plus button to clear the text field and add the value to the list.

6. Repeat the operation to add as many values as needed and then click **OK** to close the dialog box.

   The added values display under the simple rule in the data model editor.

3.2.1.7. How to add a foreign key: linking entities together

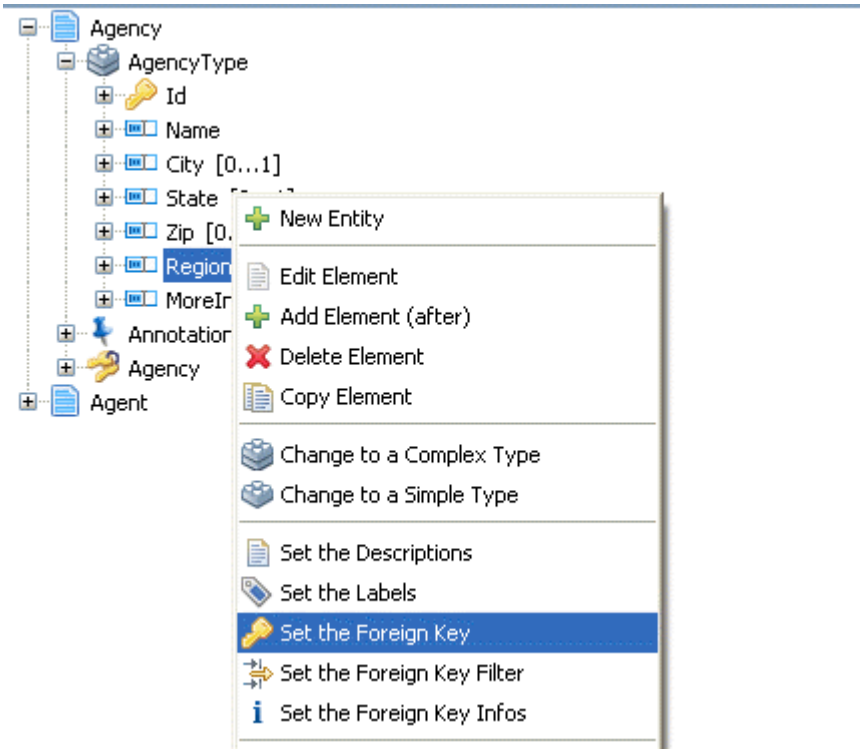
Talend Open Studio for MDM enables you to set relationship between different entities using Foreign Keys.

Prerequisite(s): You have already created data models, business entities and attributes in the data model. For further information, see section *How to create a data model*, section *How to create business entities in the data model* and section *How to add attributes to the business entity*.

Consider as an example that your data model hold the following entities: *Agency*, *Agent* and *Region*. You want to link the two entities *Agency* and *Region* together in order to represent that an *Agency* belongs to a *Region*. Consequently, *Agency* will have a new attribute that points to a *Region*.

To set a foreign key, do the following:

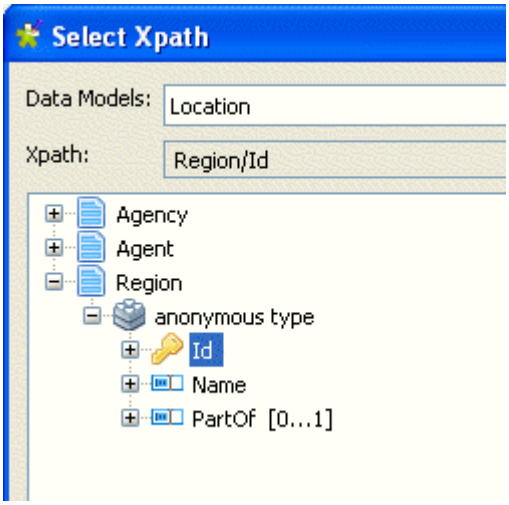
1. Add a new *Region* attribute to the *Agency* entity as outlined in section *How to add attributes to the business entity*



2. Right-click *Region* and select **Set the Foreign Key** from the list.

   [Set the Foreign Key] dialog box displays.

3. Click the three-dot button to open the **[Select Xpath]** dialog box.



4. Browse to the *Region* primary key *Id* and then click **Add** in the bottom of the dialog box to proceed to the next step.

   The Xpath to the foreign key displays in the **[Set the Foreign Key]** dialog box.

5. If you want to render the Foreign Key in the main tab, select the **Render Foreign Key in the main tab?** checkbox. By default, Talend Web UI renders the Foreign Key in a separate tab.

6. Click **OK** to close the dialog box.

   The foreign key is set to *Region/Id* under the **Annotations** node of the *Region* attribute in the data model editor.

   Entities which are linked by a foreign key are identified by a key icon with a green arrow . To jump directly to the related entity, right-click the entity in the data model editor and click **Jump to Foreign Key Entity**.

3.2.1.8. How to add a foreign key filter

One of the most useful annotations that you can set up on any of the attributes (elements) of the business entities in a data model is the foreign key filter.

Through this annotation, you can filter foreign keys by:

- values extracted from the current record through using the standard xpath predicate of the relevant attribute,
- complex expressions: literal and functions,
- current record values and complex expressions.

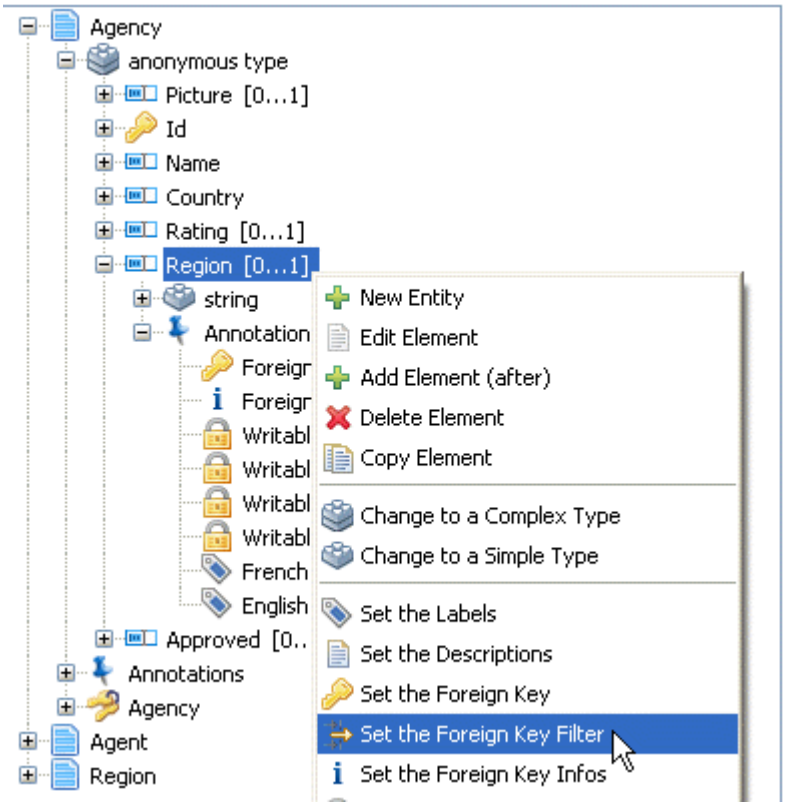
Custom filters for foreign keys are not supported when you use a SQL database to store your data records.

Prerequisite(s): You have already created data models, business entities and attributes in the data model. For further information, see section *How to create a data model*, section *How to create business entities in the data model* and section *How to add attributes to the business entity*.

Consider as an example that your data records hold the following entities: *Agency*, *Agent* and *Region*. The *Agency* and *Region* entities hold the *Country* attribute. In *Talend MDM Web User Interface*, you want to filter the foreign key values by the *Country* attribute.

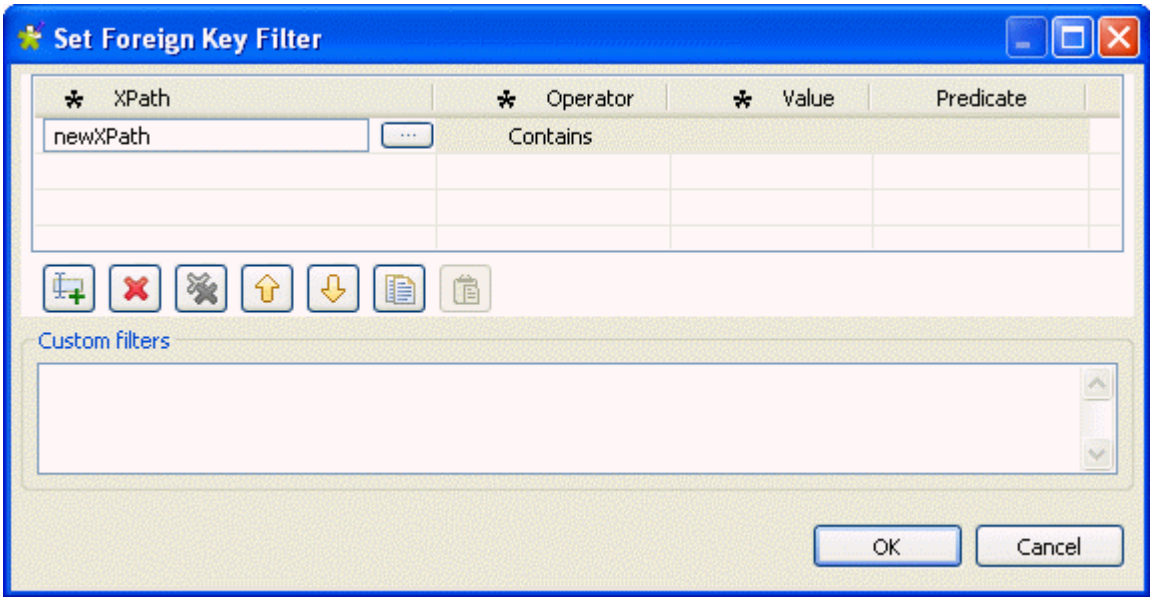
To set a foreign key filter, do the following:

1. Right-click the attribute to which you want to add the annotation, *Agency* > *Region* in this example.



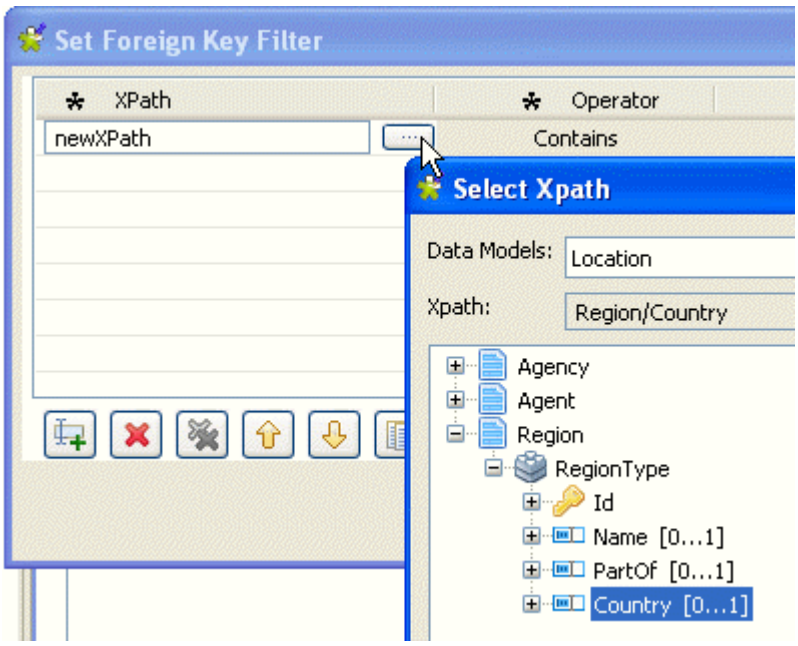
2. In the contextual menu, select **Set the Foreign Key Filter**.

   A dialog box displays.



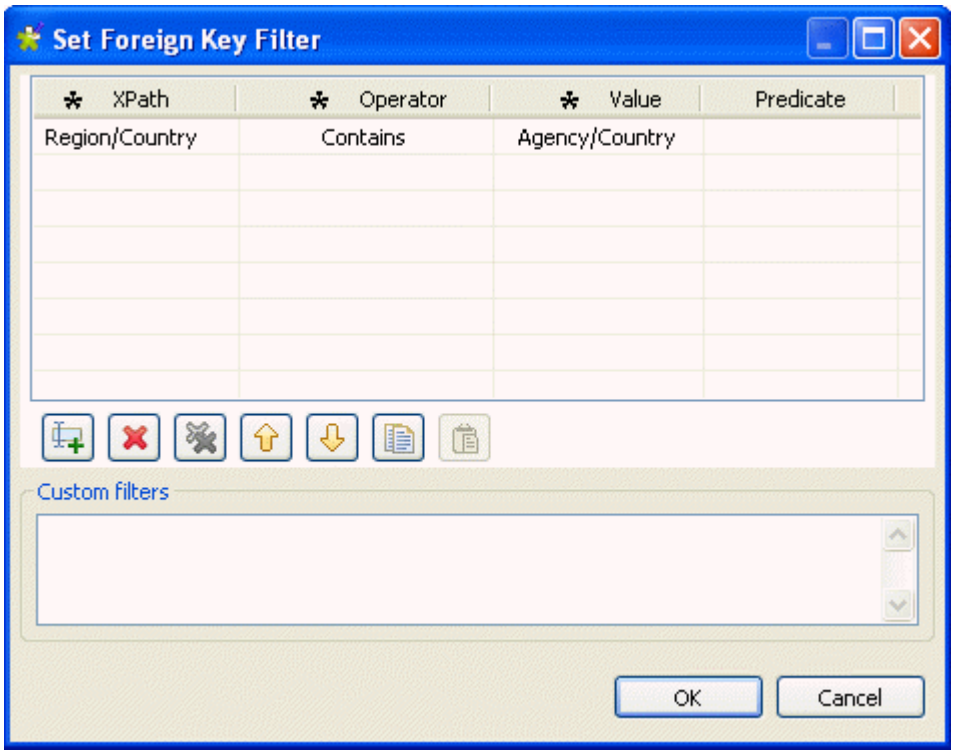
3. Click the  icon to add a line to the table where you can define the foreign key filter.

4. Click in the **XPath** column and then click the three-dot button to open a dialog box where you can select the XPath on which to set the foreign key filter.



5. Select *Region* > *Country* and then click **Add** in the dialog box.

   The XPath is set to *Region/Coutry*.



6. Click in the **Operator** column and select an operator from the list, **Contains** in this example.

7. Click in the **Value** column and then click the three-dot button to open a dialog box where you can select the value on which to set the foreign key filter.

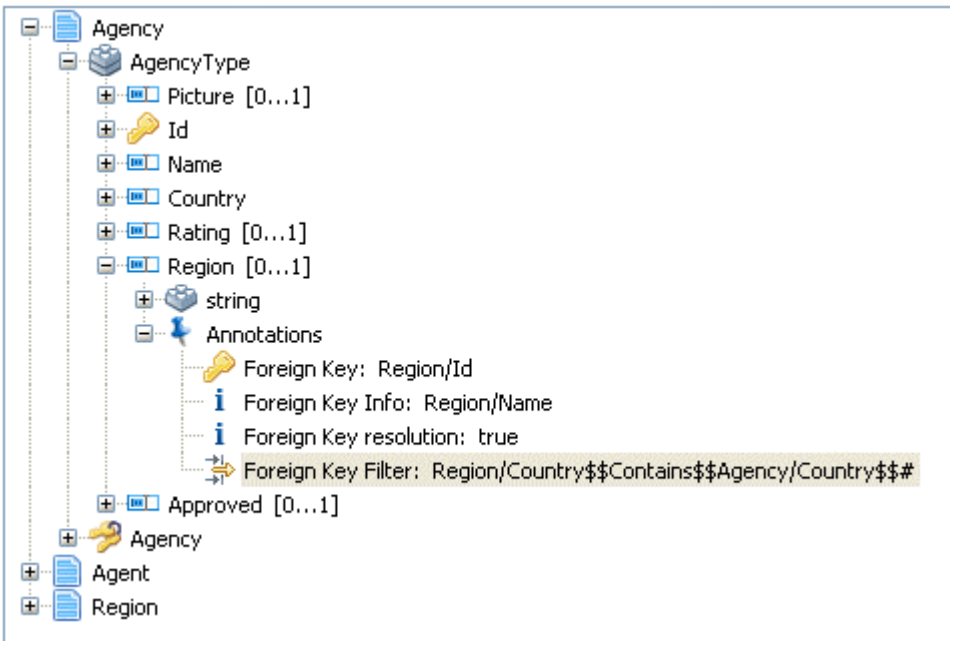
8. Select *Agency* > *Country* and then click **Add** in the dialog box.

   The foreign key filter value is set to *Agency/Coutry*.

9. In this example, you want to filter foreign keys based on the *Country* attribute:

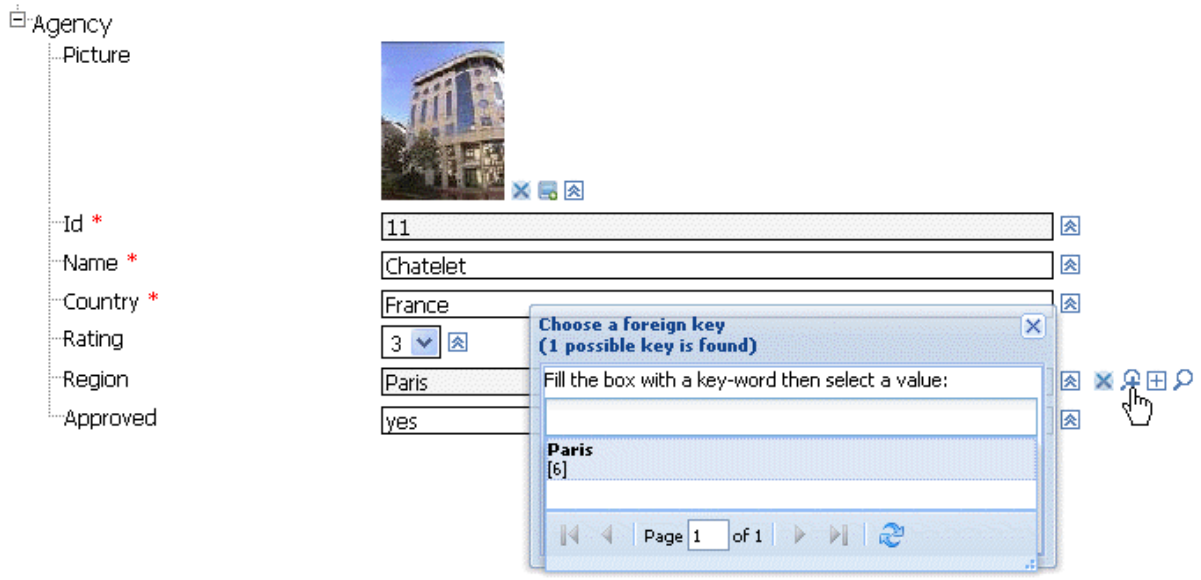
Click **OK** to close the dialog box.

The defined foreign key filter displays under the **Annotation** node of the *Region* element.



10. Click the save icon on the toolbar or press **Ctrl + S** on your keyboard to save your modifications.

Now in any agency record in *Talend MDM Web User Interface*, every time a user clicks on the foreign key icon next to the **Region** field, the list will only include the foreign keys based on the *Country* attribute.



3.2.1.9. How to set the display format of dates and numbers

Talend Open Studio for MDM enables you to decide the format according to which you want to display/insert dates or numbers in different languages in master data records in *Talend MDM Web User Interface*.

You must use a syntax based on the java.util.Formatter class to define the display of dates and numbers in a specific language. For further information about this class, see http://download.oracle.com/javase/6/docs/api/java/util/Formatter.html .

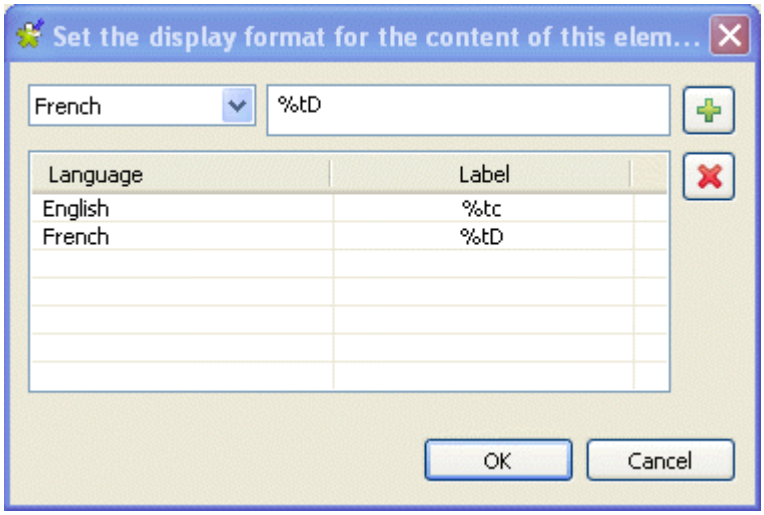
Prerequisite(s): You have already created data models, business entities and attributes in the data model. For further information, see section *How to create a data model*, section *How to create business entities in the data model* and section *How to add attributes to the business entity*.

Consider as an example that your data model holds the *Agent* entity and that you have created an attribute in this entity, *StartDate*, to display the start date of the agent.

To set the display format of dates in the English and French languages, do the following:

1. In the data model editor, expand the *Agent* entity and browse to the *StartDate* attribute.
2. Right-click *StartDate* and select **Set the display format**.

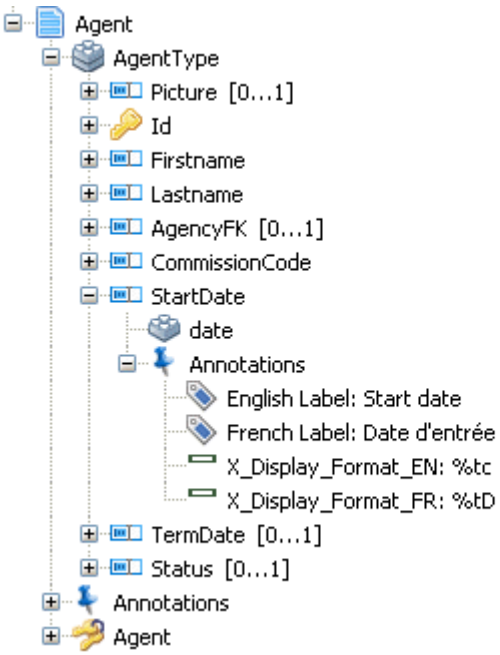
   A dialog box displays.



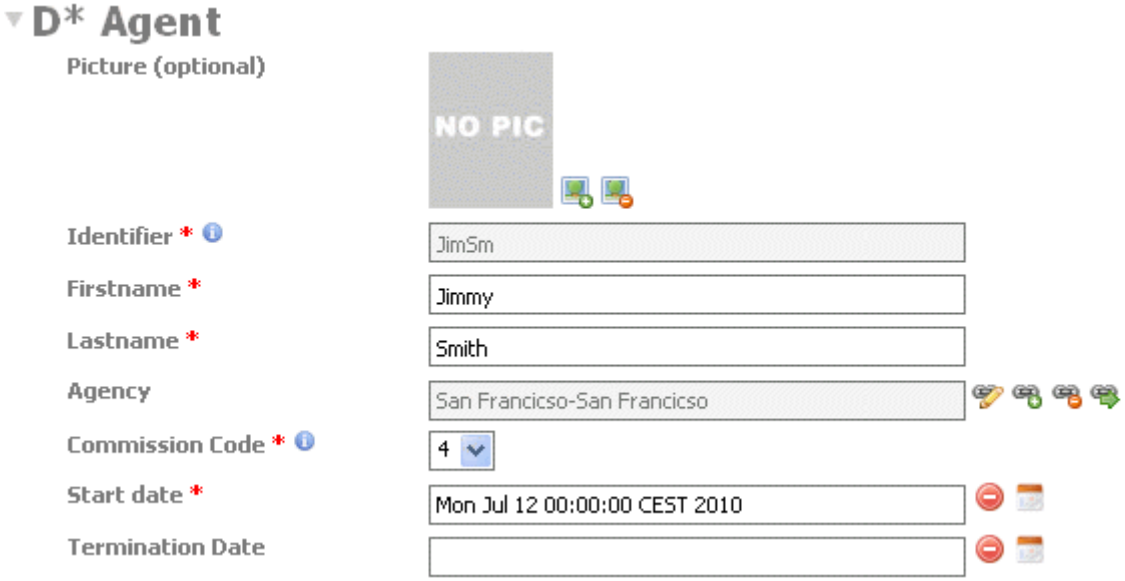
3. From the language list, select the language for which you want to define the date format.
4. In the field to the right, enter the syntax for the date format according to the *java.util.Formatter* class.

   In this example, enter `%tc` for the English language and `%TD` for the French language.
5. Click the plus button to add each of the two lines to the list below the fields, and then click **OK** to close the dialog box.

   The two annotations display below the selected attribute.



When a business user accesses the *Agent* record through *Talend MDM Web User Interface*, the start date will display as the following in English *Mon Jul 12 00:00:00 CST 2010*:



And as the following in French *07/12/10*:

You can follow the same procedure to set the display format of numbers. For example, assume that the actual value in the database is 2.5 (XSD standard); if you define the format in English as the following `%-10.4f`, the result in the web application will be *2.5000*. And if you define the format in French as the following `%-10.4f`, the result in the web application will be *2,5000*.

Java automatically substitutes the local-specific decimal separator.

3.2.1.10. How to use the Properties view in the data model editor

The data model editor has a **Properties** view that details all the properties of the currently selected element (entity or attribute) in the editor. The properties in the view are arranged in different categories in order to represent the element linguistic presentation, validation rules, etc.

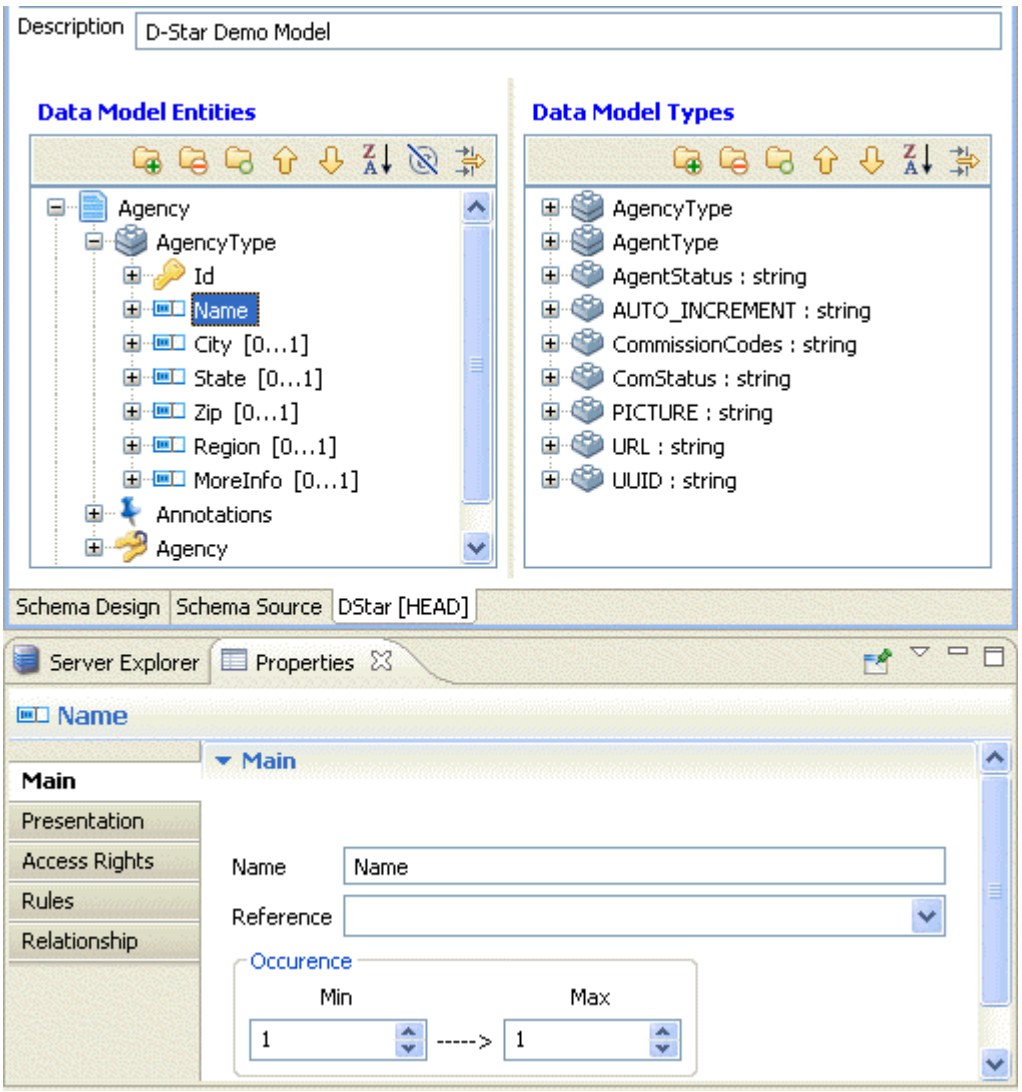
From this view, you can easily edit any of these properties and see the change in the data model on the fly.

Prerequisite(s): You have already created a data model, with the corresponding business entities and attributes. For further information, see section *How to create a data model*, section *How to create business entities in the data model* and section *How to add attributes to the business entity*.

To edit an element in a data model from the **Properties** view, do the following:

1. Open the data model that holds the elements you want to modify.

2. In the data model editor, select an element (entity or attribute) to display its main properties in the view.



3. Click, any of the tabs in the view to display the corresponding properties and modify them according to your needs.

   All your modifications will reflect in the data model editor on the fly.

3.2.2. Data model inheritance and polymorphism

Talend MDM introduces a true object-oriented data model that allows you to use inheritance and polymorphic characteristics when defining your data models. This object-oriented approach enables you to define inheritance hierarchies (inheritance trees) in the data models you create in *Talend Open Studio for MDM*. For further information on setting up a data model, see section *Data Models*.

Inheritance allows you to extend an existing type to add or override specific elements while inheriting the attributes from the main entity. This summarizes the two main concepts included in inheritance: generalization and specialization. Generalization (or abstraction) is the process of sharing attributes from the main type and inheriting them automatically in other subtypes (inheritance types). In contrast to generalization, specialization means creating new subtypes from an existing main type by adding attributes specific to each subtype.

Polymorphism allows you to set the type of an element to an abstract generic type and the concrete type is determined at run time. For instance you can have an address established and define it at runtime as a US or European address.

Inheritance and polymorphism are supported for both attributes and entities. The sections below gives examples for both cases.

3.2.2.1. How to use inheritance and polymorphism with attributes

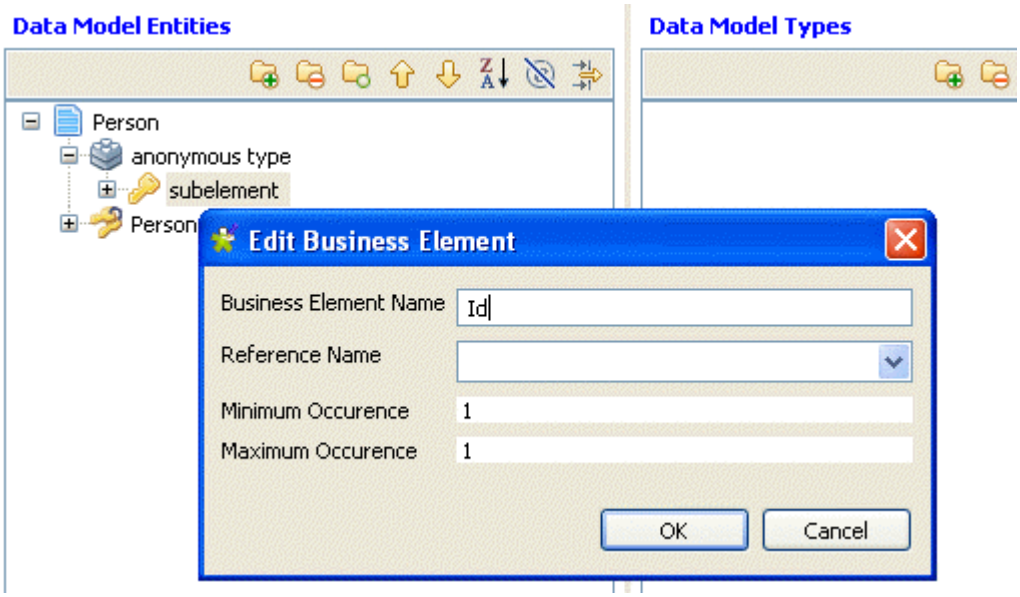
Consider as an example that your data model could contain a *Person* entity. This business entity has several attributes including *Address*. This attribute has one main reusable type, *AddressType*. *Talend Open Studio for MDM* enables you to create inheritance reusable types that return a collection of new elements while inheriting the attributes from the *AddressType* in order to return one address format specific to Europe and another specific to the USA. So in this example, the Europe and US address types both extend the generic address type.

Prerequisite(s): You have already created the data model and business entities in the data model. For further information, see section *How to create a data model* and section *How to create business entities in the data model* respectively.

To create inheritance and polymorphism for attributes, do the following:

1. Expand the business entity you created and the group type in succession and right-click **subelement** then select **Edit Element** from the contextual menu.

   A dialog box displays.

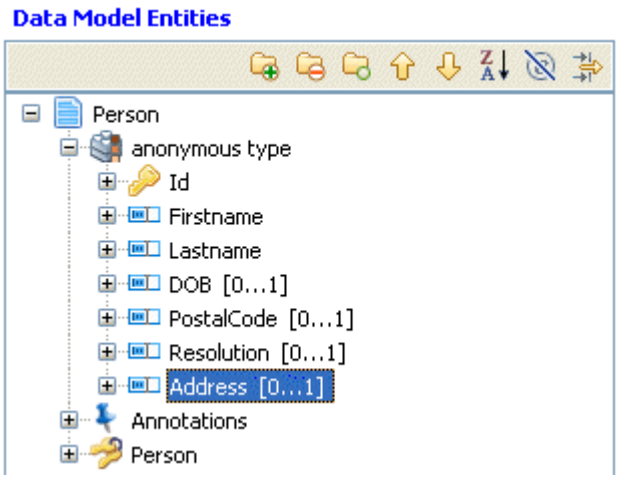


2. Change the name to *Id* and set the minimum and maximum occurrences to *1* and then click **OK** to close the dialog box.

3. Right-click *Id*, select **Add Element (after)** from the contextual menu, and then select **Add string Element**.

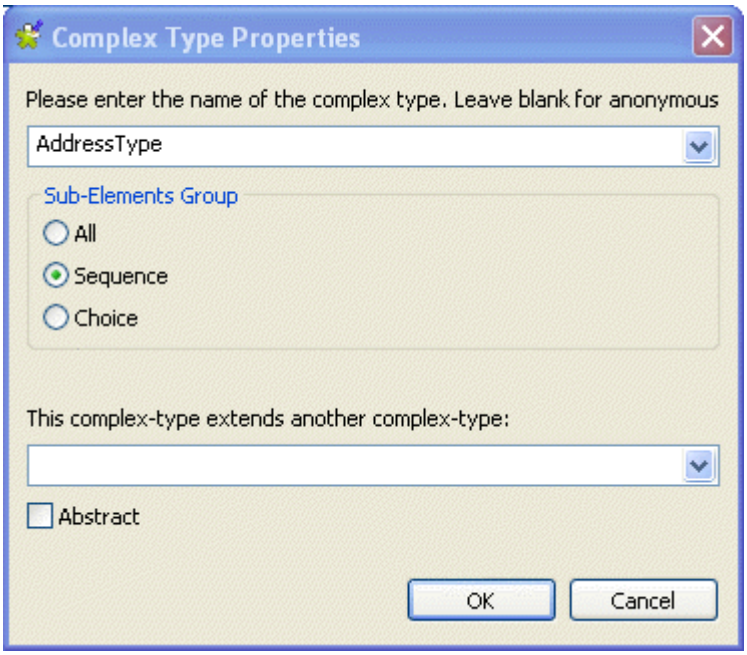
   Add *Firstname*, *Lastname*, *DOB*, *PostalCode*, *Resolution* and *Address* as attributes of the *Person* entity as outlined in section *How to add attributes to the business entity*.

   Your data model should look like the one shown below.



4. Right-click *Address* and select from the list **Change to a complex type**.

   A dialog box displays.



   The complex type enables you to define a complete structure, as you need for an address, and not only a single element.

5. Enter a name for the complex type and then select the **Sequence** option in order to list the address elements in the defined order.

   Click **OK** to validate your changes and close the dialog box.

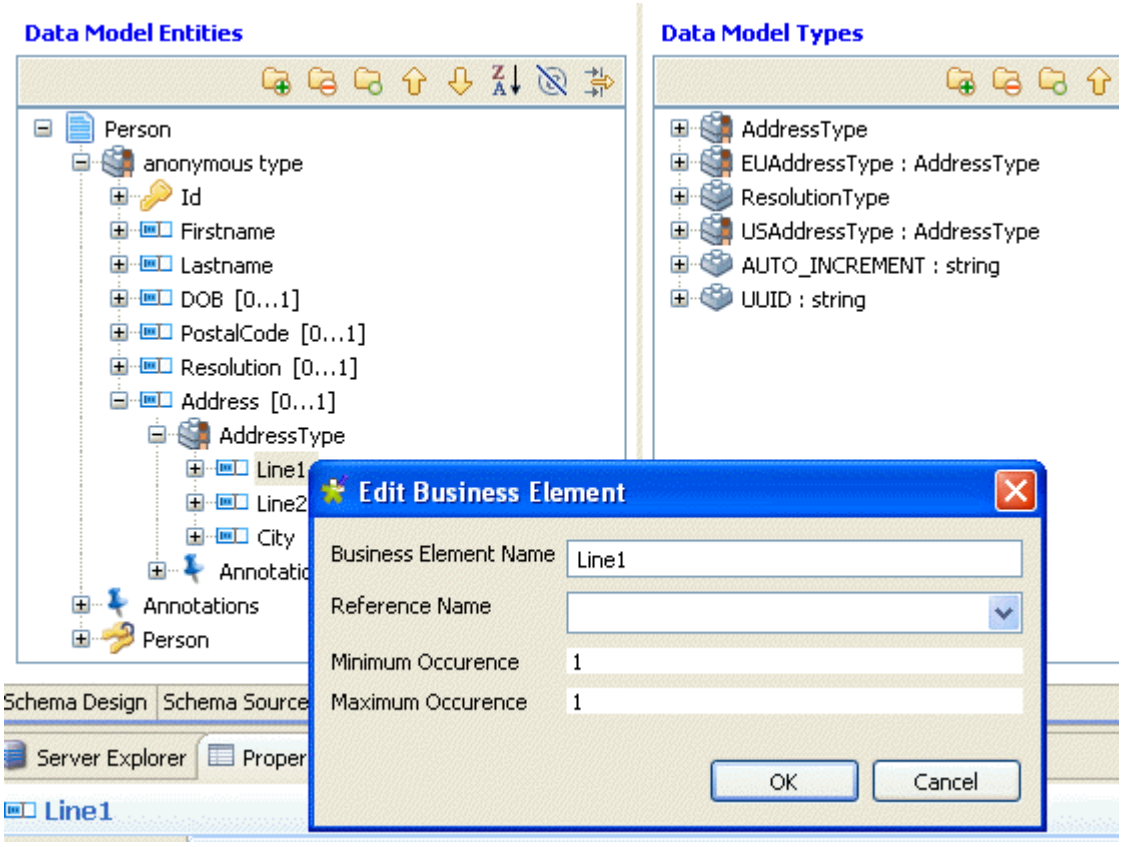
   The complex type is listed under the *Address* attribute and also in the **Data Model Types** panel.

6. You now need to create the elements in your address:

   Expand *AddressType* and rename **subelement** to *Line1*.

7. Right-click *Line1* , select **Add Element (after)** from the contextual menu, and then select **Add string Element**. Add two other elements to the address: *Line2* and *City*.

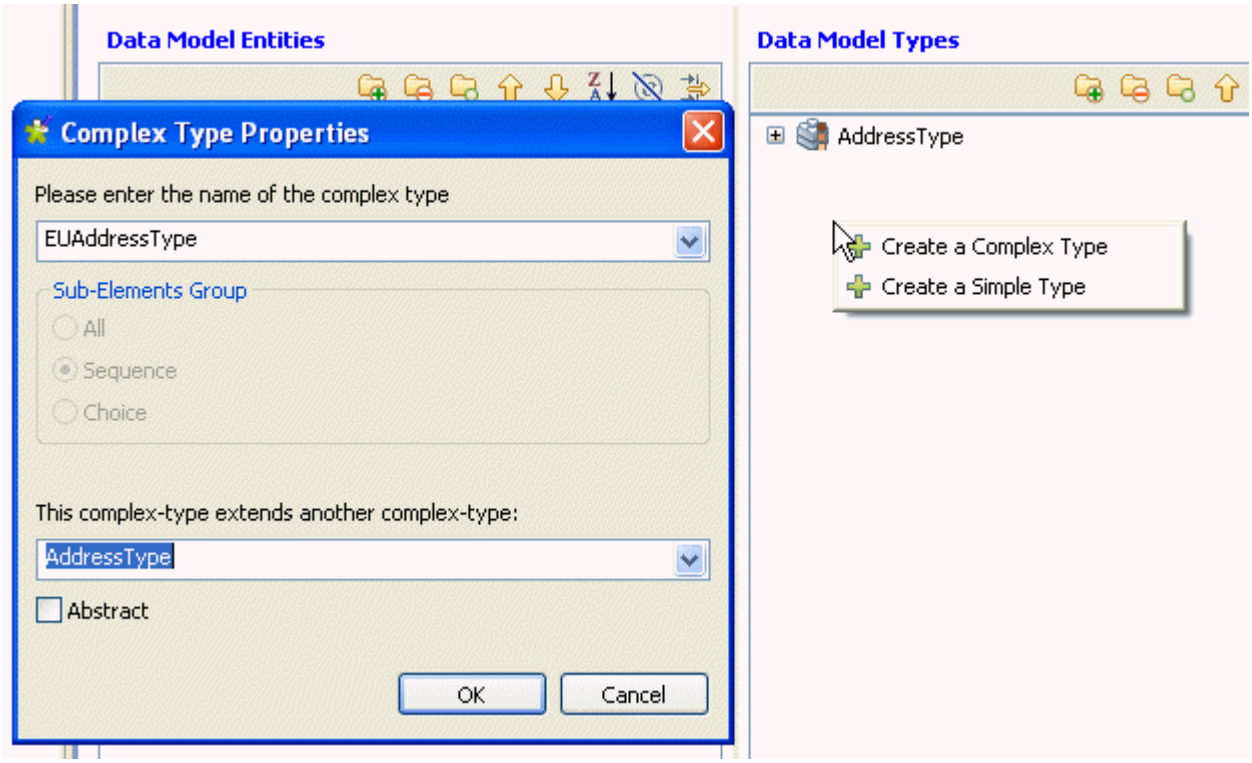
   The elements in the address types are now created below the *Address* attribute and also in the **Data Model Types** panel.



Now you need to create the inheritance type(s) for the US and EU addresses. The *EUAddressType* extends the main *AddressType* by adding a postal code and a country, whereas the*USAddressType* extends the main *AddressType* by adding a zip code and a state.

8. Right-click in the **Data Model Types** panel and select **Create a Complex Type**.

   A dialog box displays.

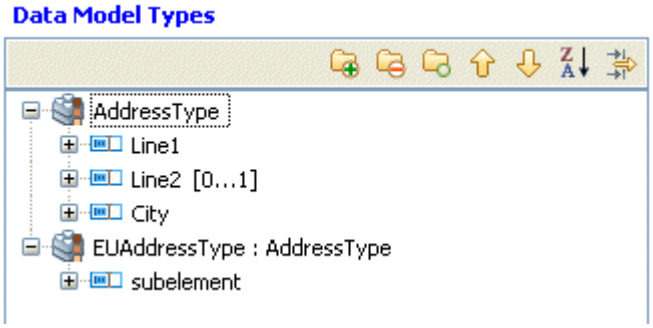


9. Enter a name for this inheritance type, *EUAddressType* in this example.

   From the complex type list, select the main type from which you want to inherit existing elements, *AddressType* in this example.

10. Click **OK** to validate your changes and close the dialog box.

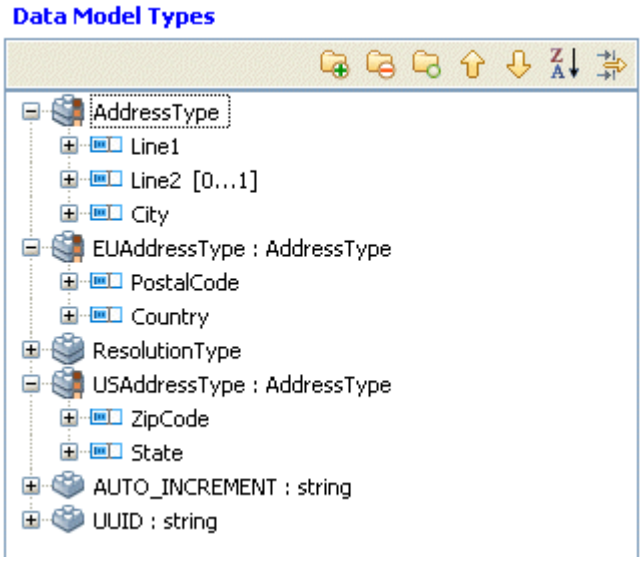
    The inheritance type is created in the **Data Model Types** panel with a by-default **subelement** element.



    This inheritance type derives the three elements in the main *AddressType*. You can now add elements specific to this type in order to have a complete address format for Europe.

11. Change **subelement** to *PostalCode* and then add a *Country* element.

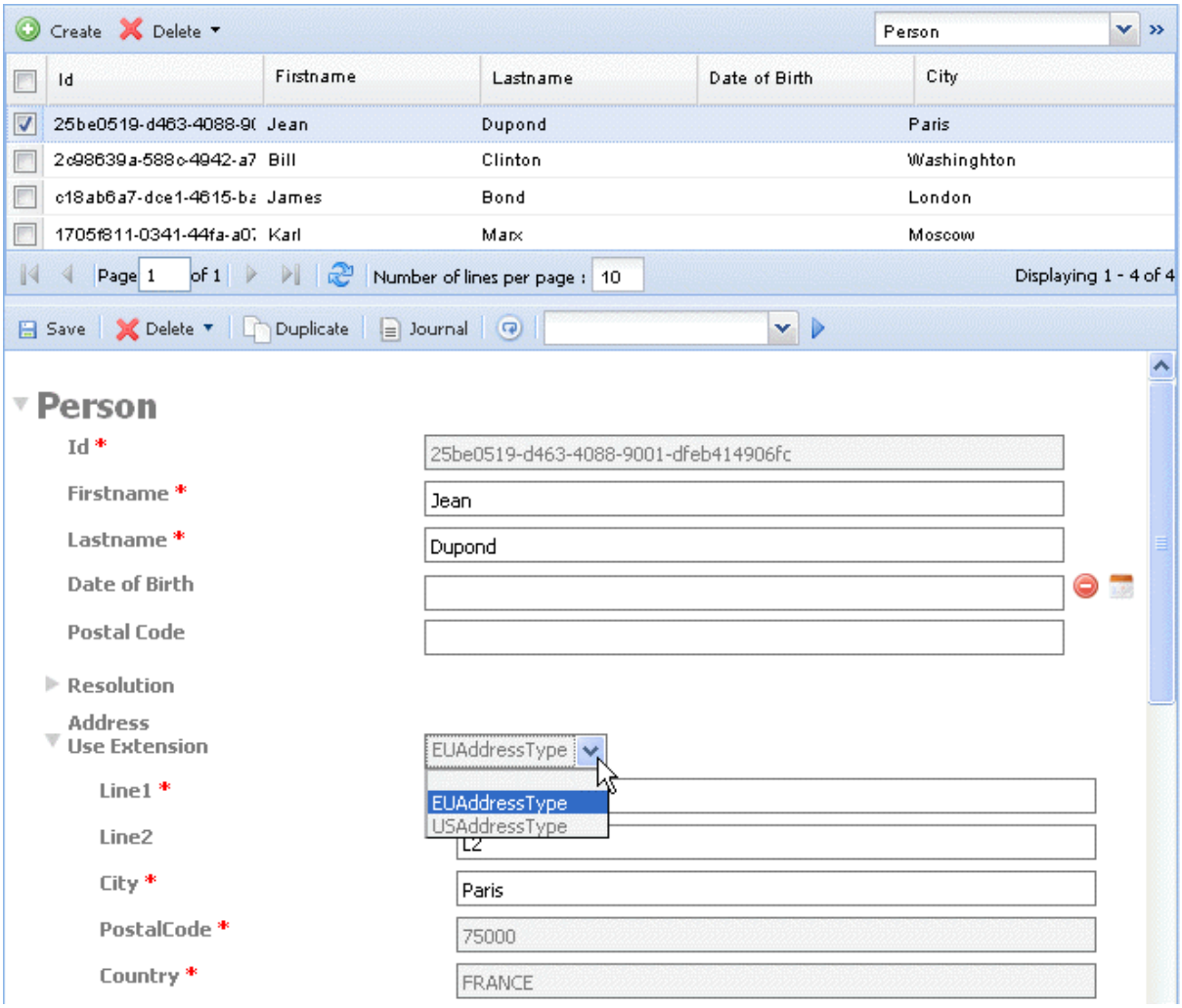
    Follow the same steps in the **Data Model Types** panel to create another inheritance type that you call *USAddressType* that inherits the main *AddressType*. Add a *ZipCode* element and a *State* element in order to complete the address format for the USA.



12. Save your data model.

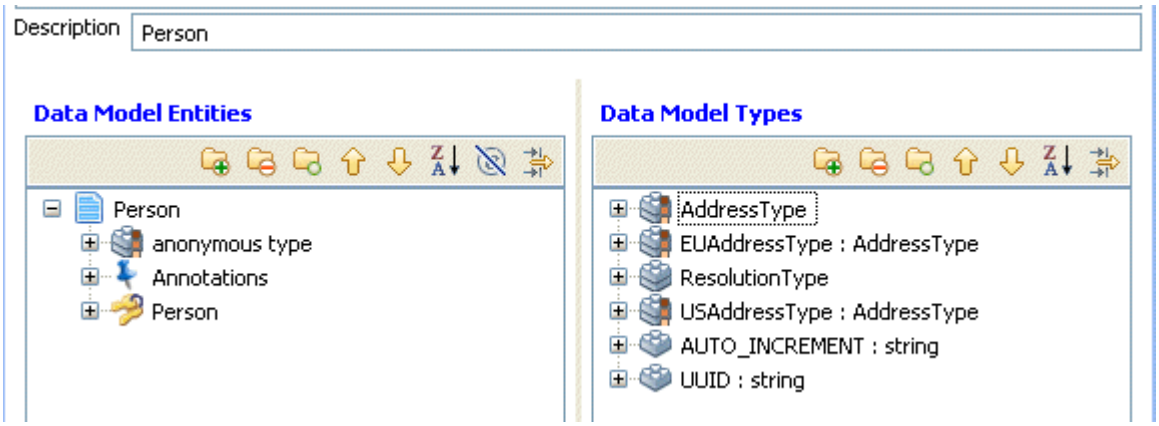
You have achieved inheritance here at the attribute level by extending an existing type (*AddressType*) to add specific elements to complete the US and EU addresses while inheriting the attributes from the main address type.

Polymorphism is also achieved when, at runtime, the actual address instance of a person can be an *EUAddressType* or a *USAddressType*. When business users browse master data records through *Talend MDM Web User Interface*, the concrete address type for the open record (person) is automatically selected. Business users will also have in an open record a **Use Extension** list for the *Address* attribute. From this list they can choose any of the extended address types you defined in *Talend Open Studio for MDM*.



By default, the extended types list displays the code of the types, but *Talend Open Studio for MDM* also lets you define labels for the extended types in order to display these labels in the **Use Extension** list. You can also decide the order in which you want to display these types. To do so, proceed as follows:

1. Open the data model where you have defined the complex types.



2. Click **Schema Source** at the bottom of the editor to open the source XML schema of the data model.

```
<xsd:complexType name="EUAddressType">
  <xsd:complexContent>
    <xsd:extension base="AddressType">
      <xsd:sequence maxOccurs="1" minOccurs="1">
        <xsd:element maxOccurs="1" minOccurs="1" name="PostalCode" type="xsd
        <xsd:element maxOccurs="1" minOccurs="1" name="Country" type="xsd:st.
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
<xsd:complexType name="USAddressType">
  <xsd:complexContent>
    <xsd:extension base="AddressType">
      <xsd:sequence maxOccurs="1" minOccurs="1">
        <xsd:element maxOccurs="1" minOccurs="1" name="ZipCode" type="xsd:st.
        <xsd:element maxOccurs="1" minOccurs="1" name="State" type="xsd:stri.
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
```

3. Browse to the schema corresponding to the extended complex type to which you want to add a label and which you want to display in a certain order for the business user.

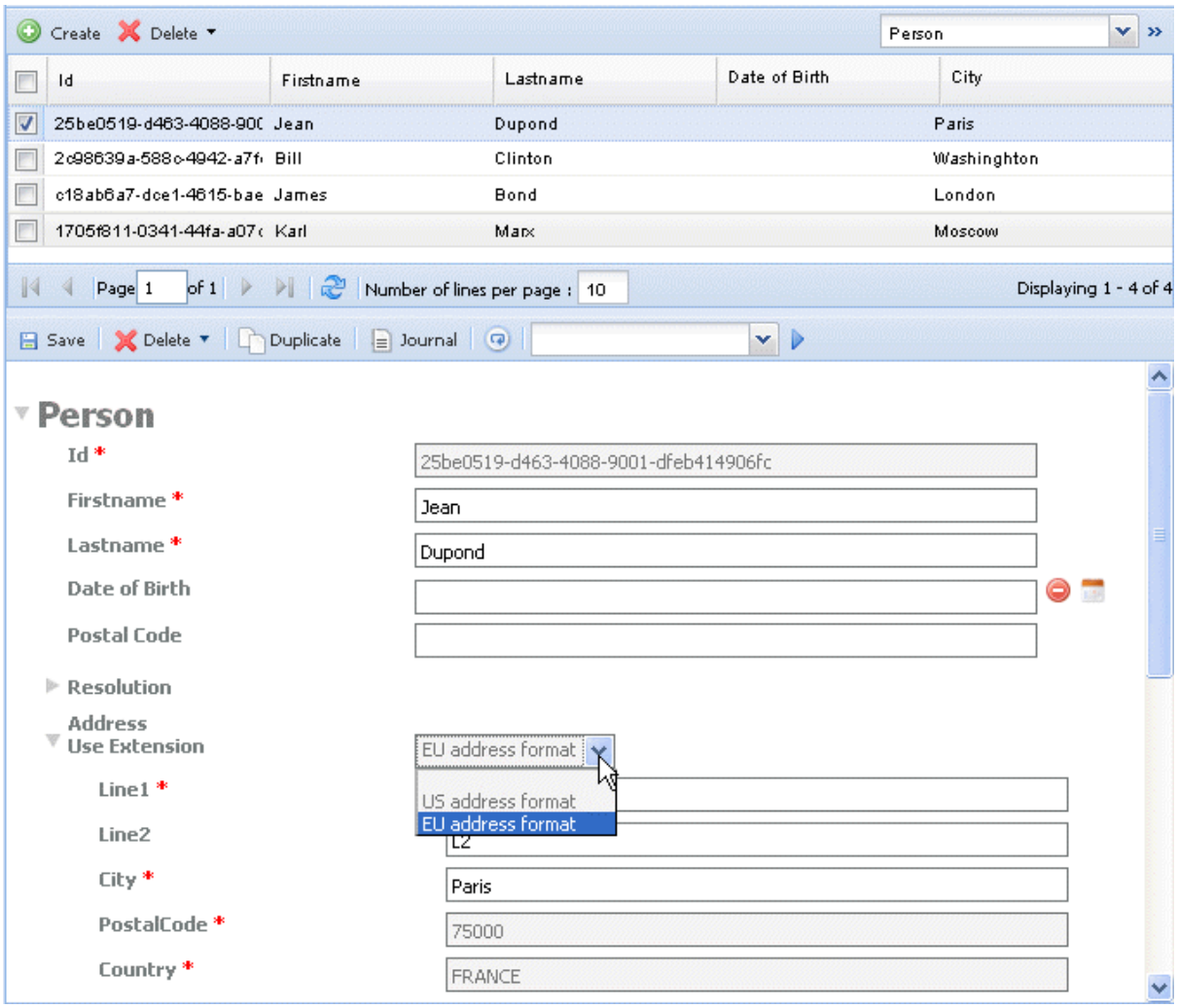
4. Inside the root element of the complex type, use the `X_Label_<ISO>` annotation to add a label for the extended type and the `X_Order_Value` annotation to define the order of the extended type, as follows:

```
...
<xsd:complexType name="EUAddressType">
 <xsd:annotation>
  <xsd:appinfo source="X_Label_EN">EU address format</xsd:appinfo>
    <xsd:appinfo source="X_Label_FR">format d'addresse UE</xsd:appinfo>
    <xsd:appinfo source="X_Order_Value">2</xsd:appinfo>
 </xsd:annotation>
  <xsd:complexContent>
    <xsd:extension base="AddressType">
      <xsd:sequence maxOccurs="1" minOccurs="1">
        <xsd:element maxOccurs="1" minOccurs="1" name="PostalCode"
 type="xsd:string"/>
        <xsd:element maxOccurs="1" minOccurs="1" name="Country" type="xsd:string"/>
      </xsd:sequence>
    </xsd:extension>
 </xsd:complexContent>
</xsd:complexType>
<xsd:complexType name="USAddressType">
<xsd:annotation>
   <xsd:appinfo source="X_Label_EN">US address format</xsd:appinfo>
   <xsd:appinfo source="X_Label_FR">format d'addresse US</xsd:appinfo>
   <xsd:appinfo source="X_Order_Value">1</xsd:appinfo>
</xsd:annotation>
<xsd:complexContent>
  <xsd:extension base="AddressType">
    <xsd:sequence maxOccurs="1" minOccurs="1">
    <xsd:element maxOccurs="1" minOccurs="1" name="ZipCode"   type="xsd:string"/>
    <xsd:element maxOccurs="1" minOccurs="1" name="State" type="xsd:string"/>
   </xsd:sequence>
  </xsd:extension>
 </xsd:complexContent>
</xsd:complexType>
...
```

In the above schema, you want to name the *EUAddressType* as *EU address format* and you want it to be the second in the list of extended types. You want also to name the *USAddressType* as *US address format* and you want it to be the first in the list of extended types.

5. Save your modifications in the model editor.

When business users browse master data records in the updated data model through *Talend MDM Web User Interface*, the **Use Extension** list for the *Address* attribute shows the extended type labels you defined in the annotations in the order you defined as well.



3.2.2.2. How to use inheritance and polymorphism with entities

Consider as an example that your data model could contain the following entities: *Company*, *Individual*, *Party* and *Product*. With *Talend Open Studio for MDM*, you can define inheritance types where *Individual* and *Company* both extend *Party*. *Party* becomes a polymorphic element that holds individuals and companies.

You can also define in the *Product* entity a foreign key in the *supplier* attribute that points at the *Party* entity. So, this foreign key can also point at *Individual* or *Company* since both extend *Party*, so it acts as a polymorphic foreign key.

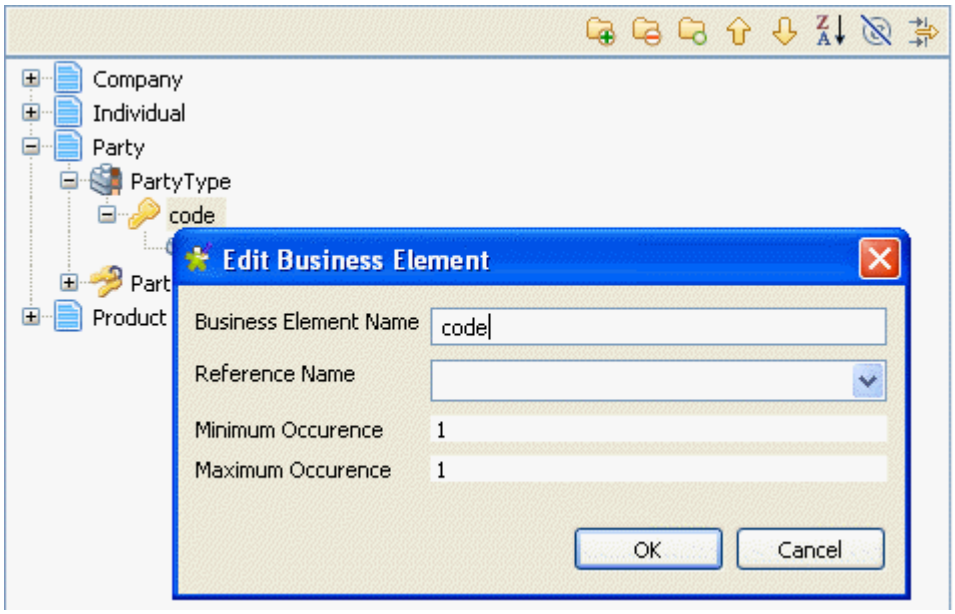
Prerequisite(s): You have already created your data model. For further information, see section *How to create a data model*.

To create inheritance and polymorphism for entities, proceed as follows:

1. In your data model, create the business entities *Company*, *Individual* and *Party* selecting the **Complex type** and **Sequence** options. Create a *Product* business entity selecting the **Complex type** and **All** options. For further information, see section *How to create business entities in the data model*.

2. Expand *Party* and its group type, right-click **subelement** then select **Edit Element** from the contextual menu.

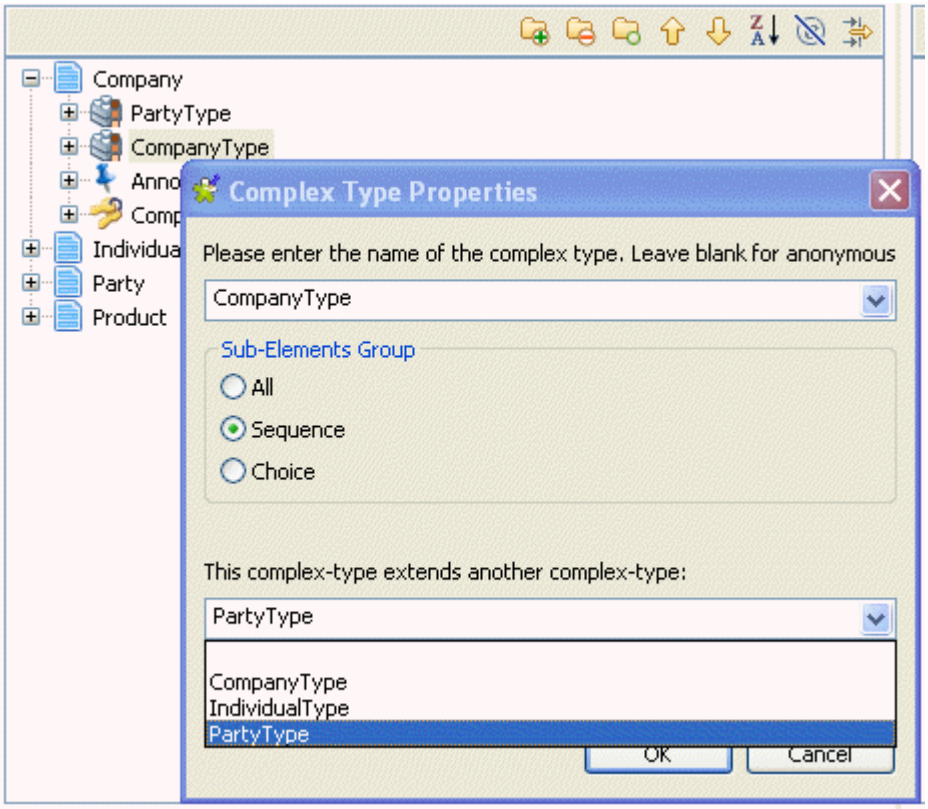
   A dialog box displays.



3. Change the name to *code* and set the minimum and maximum occurrences to *1* and then click **OK** to close the dialog box.

4. Expand *Company* and its group type, and then change **subelement** to *name*. You will only need a name for the company.

5. Double-click *Companytype* to open the **[Complex Type Properties]** dialog box.



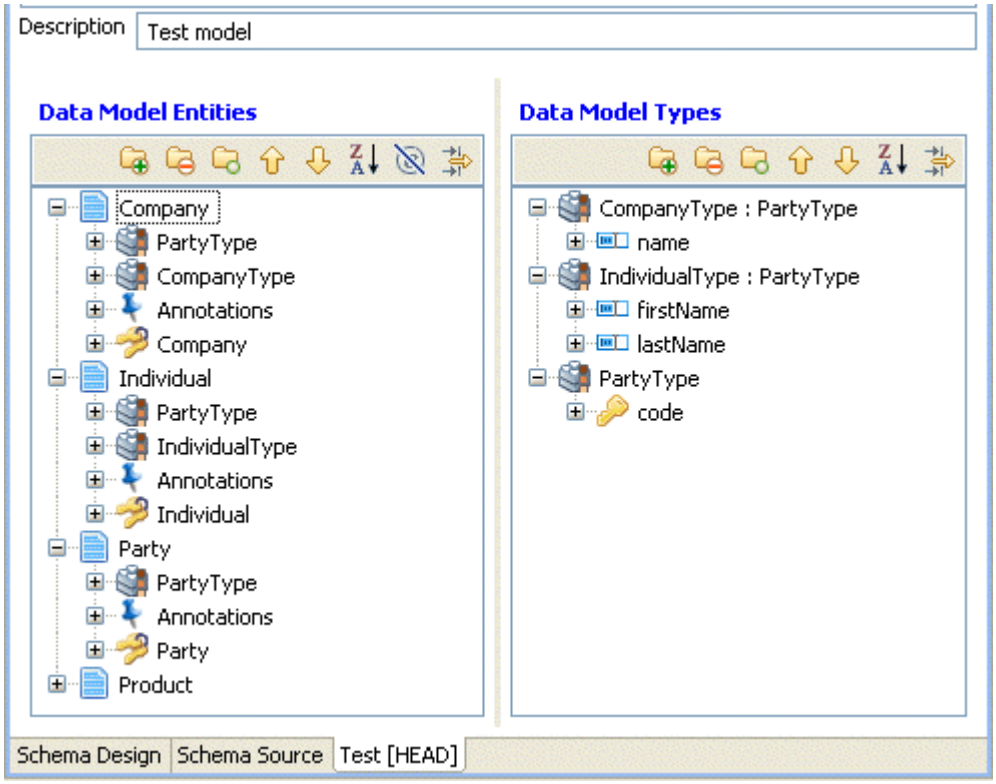
6. From the complex type list, select *PartyType* and then click **OK** to close the dialog box.

This will add the *PartyType* as a group type below *Company*. Here, you have defined inheritance between the *Company* and *Party* entities where *Company* extends *Party*.

7. Do the same to define inheritance between the *Individual* and *Party* entities where *Individual* extends *Party*.

8. Define two attributes in the group type of the *Individual* entity: *firstName* and *lastName*. You will need these two elements for an individual.

   Your data model should look like the one shown below:



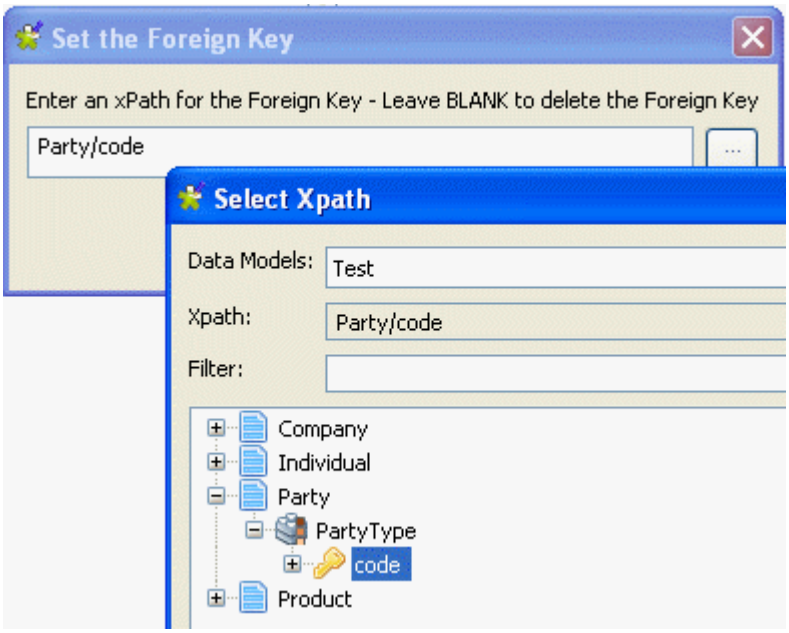
The *Company* and *Individual* entities extend the *Party* entity using the *PartType* group. All complex types are also listed in the **Data Model Types** panel.

Now, you can create a polymorphic foreign key which, by pointing to the *Party* entity, can also point to the *Individual* or *Company* entities, since both extend *Party*.

9. Create a *supplier* attribute in the *Product* entity.

   Right-click *supplier* and select **Set the Foreign Key** from the list.

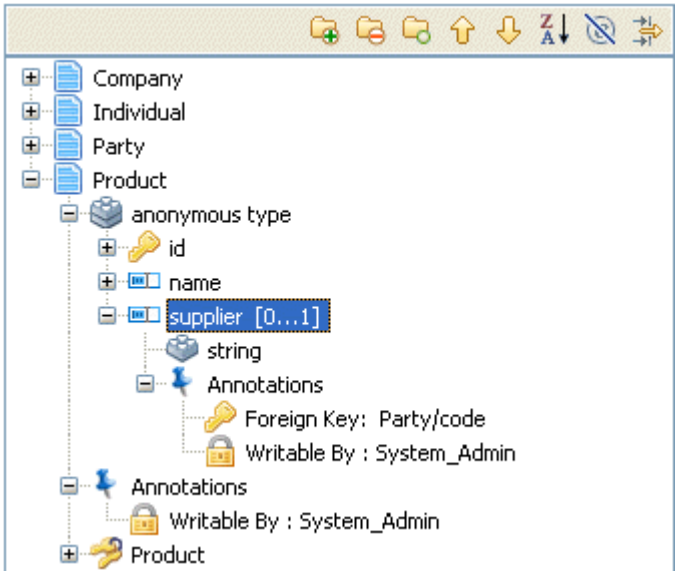
   A dialog box displays.



10. Click the three-dot button and browse to set an Xpath to the entity to which you want the foreign key to point.

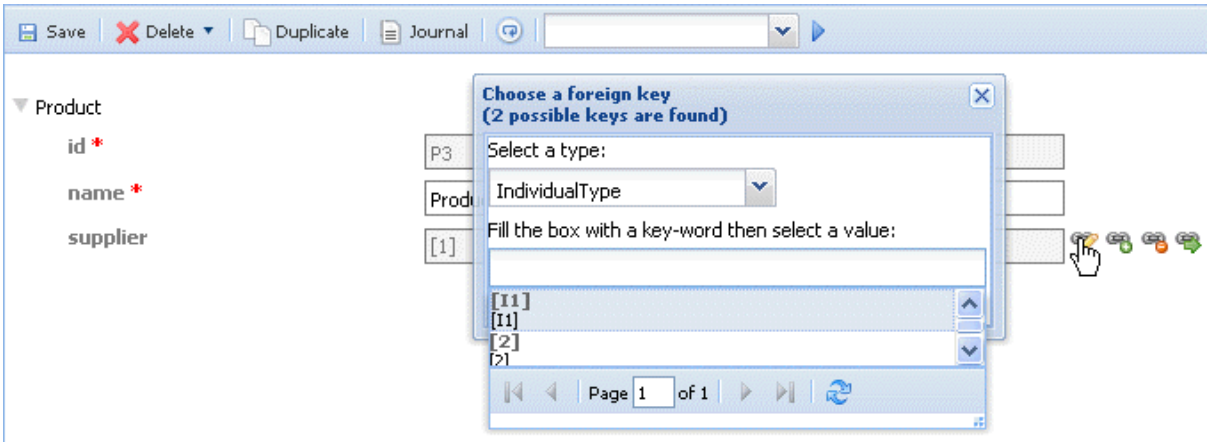
11. Click **Add** and then **OK** to close the dialog boxes.

    The foreign key path is listed below the *supplier* attribute.



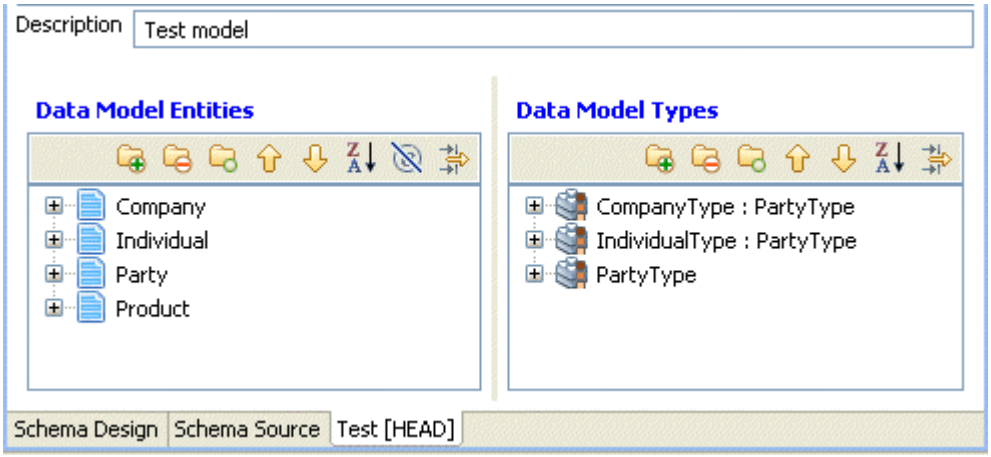
You have achieved inheritance here at the entities level by adding the *Party* group type to the *Company* and *Individual* entities.

Polymorphism is also achieved when, at runtime, the actual *Party* element can be expanded into *Individual* or *Company* when a business user tries to create a data record through *Talend MDM Web User Interface*. Business users will also have access to a foreign key list that displays *Individualtype* and *CompanyType* when they click the icon. From this list they can choose any of the extended types you defined in *Talend Open Studio for MDM*.



By default, the extended types list displays the code of the types, but *Talend Open Studio for MDM* also lets you define labels for the extended types in order to display these labels in the **Select a type** list. You can also decide the order in which you want to display these types. To do so, proceed as follows:

1. Open the data model where you have defined the complex types.



2. Click **Schema Source** at the bottom of the editor to open the source XML schema of the data model.

```
  <xsd:complexType name="IndividualType">
    <xsd:complexContent>
      <xsd:extension base="PartyType">
        <xsd:sequence>
          <xsd:element maxOccurs="1" minOccurs="1" name="firstName" type="xsd:string">
<xsd:annotation>


<xsd:appinfo source="X_Write">System_Admin</xsd:appinfo>
</xsd:annotation>
</xsd:element>
          <xsd:element maxOccurs="1" minOccurs="1" name="lastName" type="xsd:string">
<xsd:annotation>


<xsd:appinfo source="X_Write">System_Admin</xsd:appinfo>
</xsd:annotation>
</xsd:element>
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
```

3. Browse to the schema corresponding to the extended complex type to which you want to add a label and which you want to display in a certain order for the business user.

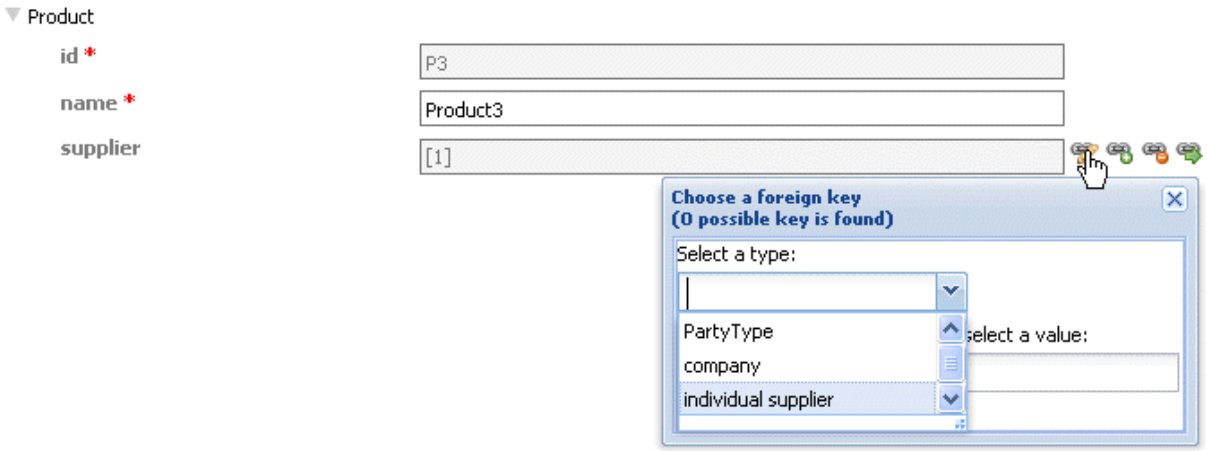
4. Inside the root element of the complex type, use the `X_Label_<ISO>` annotation to add a label for the extended type and the `X_Order_Value` annotation to define the order of the extended type, as follows:

```
...
<xsd:complexType name="IndividualType">
 <xsd:annotation>
  <xsd:appinfo source="X_Label_EN">individual supplier</xsd:appinfo>
    <xsd:appinfo source="X_Label_FR">fournisseur individuel</xsd:appinfo>
    <xsd:appinfo source="X_Order_Value">3</xsd:appinfo>
 </xsd:annotation>
  <xsd:complexContent>
...
```

   In the above schema, you want to name the *IndividualType* as *individual supplier* and you want it to be the third in the list of extended types.

5. Save your modifications in the model editor.

When business users browse master data records in the updated data model through *Talend MDM Web User Interface*, the  icon gives them access to the extended type list that shows the labels you defined in the annotation elements in the defined order.



3.2.3. Managing data models

An authorized user can also import/export, copy/paste and delete created data models from *Talend Open Studio for MDM*.

It is also possible to import and share MDM complete projects or only a data model or part of the data model from the community web page, **Talend Exchange**. For further information, see section *Projects/objects on Talend Exchange*.

3.2.3.1. How to export data models

From *Talend Open Studio for MDM* you can export one or multiple data models in order to exchange them between:

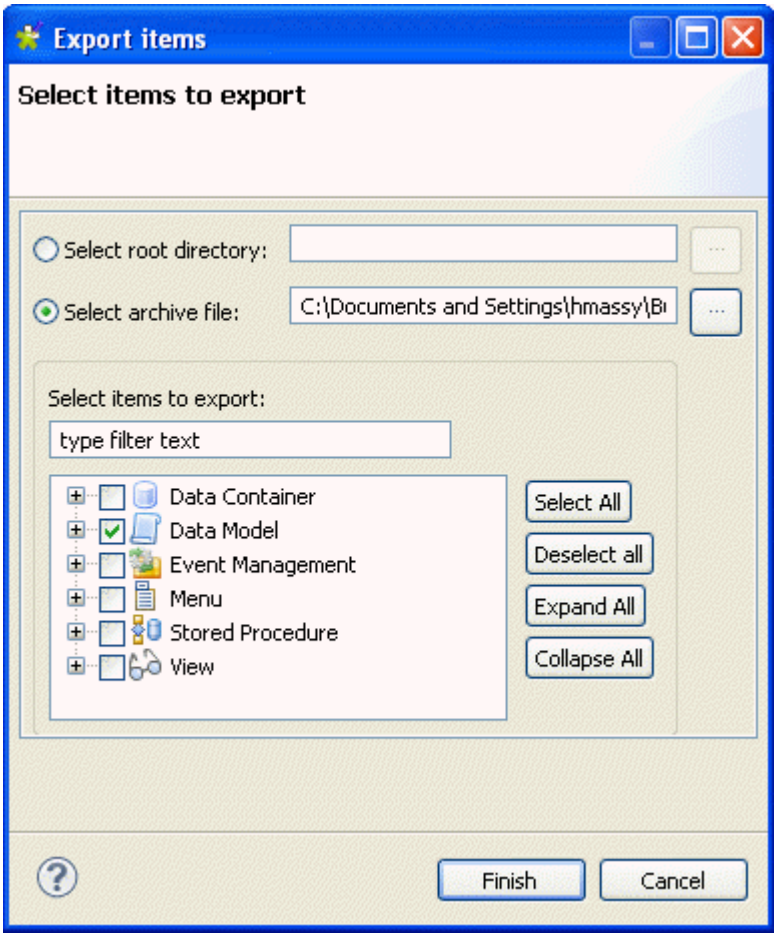
- two different MDM servers or repositories,
- two different Versions from the same/different MDM servers or repositories, for example.

Prerequisite(s): You have already connected to the MDM server from *Talend Open Studio for MDM*. At least one data model exists.

To export one or multiple data models:

1. In the **MDM Repository** tree view, right-click the **Data Model** node and select **Export items** from the contextual menu.

   The **[Export items]** dialog box displays and the **Data Model** check box is selected by default.



   If you want to export all data objects in the repository and not only data models, click the **Select All** button to select the check boxes of all data objects in the list.

2. To export the selected data models in a folder, select the **Select root directory** option and click the three-dot button next to the option.

   A dialog box displays.

3. Browse to where you want to save the data model and click **OK** to close the dialog box.

   The path to the target folder is set in the field next to the selected option.

4. In the **[Export items]** dialog box, click **Finish** to validate your changes and close the dialog box.

   A folder holding all data models in the current MDM Repository is created in the specified path. An XML file that lists all exported elements is also created.

5. To export the selected data models in a zip file, select the **Select archive file** option in the **[Export Items]** dialog box and then click the three-dot button next to the option.

   A dialog box displays.

6. Browse to where you want to save the data model archive file.

7. In the **File Name** field, enter a name for the archive file and then click **Open** to close the dialog box.

The path to the target archive file is set in the field next to the selected option.

8. In the **[Export items]** dialog box, click **Finish** to validate your changes and close the dialog box.

   An archive file holding all data models in the current MDM Repository is created in the specified path. An XML file that lists all exported elements is also created.

If you want to export only one data model of all those in the MDM Repository, expand **Data Model** in the **[Export items]** dialog box and then select the check box next to the name of the data model you want to export and proceed as usual.

3.2.3.2. How to import data models

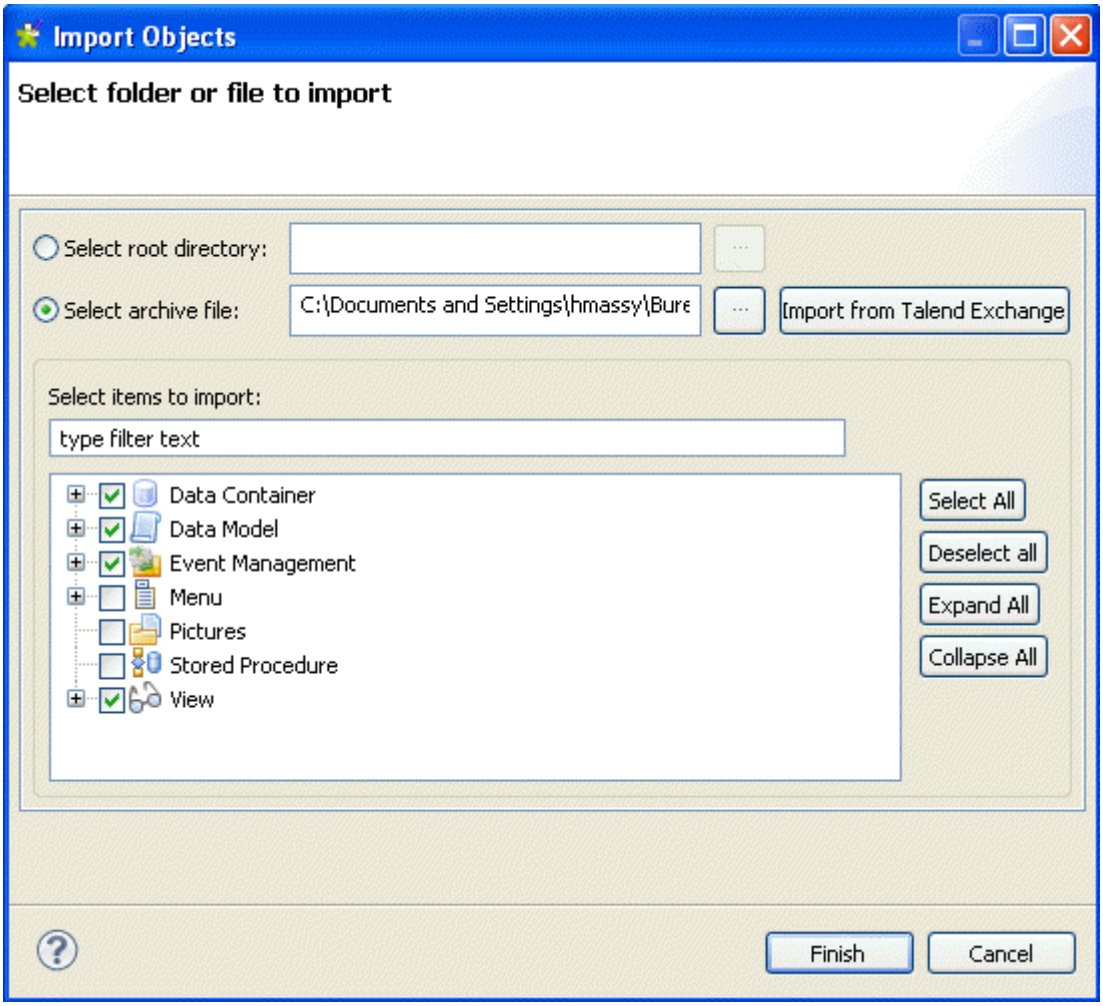
From *Talend Open Studio for MDM* you can import data models into the current MDM Repository that have been created in other MDM Repositories or in different Versions of the current MDM Repository.

Prerequisite(s): You have already connected to the MDM server from *Talend Open Studio for MDM*. You have exported one or multiple data models from other MDM repositories.

To import one or multiple data models into the current MDM Repository:

1. In the **MDM Repository** tree view, right-click the **Data Model** node and select **Import items** from the contextual menu.

   The **[Import Objects]** dialog box displays.



2. Select the root directory or archive file option according to whether the data models you want to import are saved in a folder/archive file and then click the three-dot button next to the option.

   A dialog box displays.

3. Browse to the folder/archive file holding the data models and click **OK**/**Open** to close the dialog box.

4. In the **[Import Objects]** dialog box, click **Finish** to validate your changes and close the dialog box.

   All data models in the folder/archive file are exported and listed under the **Data Model** node in the current *Talend Open Studio for MDM*.

If you want to import only one data model of all those saved in the folder/archive file, expand **Data Model** in the **[Import Objects]** dialog box and then select the check box next to the name of the data model you want to import and proceed as usual.

3.2.3.3. How to edit a data model

You can open a data model you have already created to check its settings and/or edit any of the defined elements (entities, attributes, annotations or reusable types) in order, for example, to adapt the data model to the specific needs of an enterprise.

Prerequisite(s): You have already connected to the MDM server from *Talend Open Studio for MDM*. At least one data model exists.

To edit a data model from the data model editor, do the following:

1. In the **MDM Repository** tree view, expand the **Data Model** node.

2. Right-click the data model you want to edit and select **Edit properties** from the contextual menu.

   The corresponding editor opens on the selected data model.

3. Double-click an element in the data model editor to open the corresponding dialog box.

4. Modify the element as needed and then click **OK** to close the dialog box.

5. Click the save icon on the toolbar or press **Ctrl + S** on your keyboard to save your changes.

   The selected element is modified accordingly.

If you try to update a data model that has been modified by somebody else after you have retrieved it from the database, a warning message displays to warn you that saving your modifications will overwrite the other user's changes.

You can also edit any element in the data model from the **Properties** view in the data model editor. For further information, see section *How to use the Properties view in the data model editor*.

3.2.3.4. How to copy/paste a data model

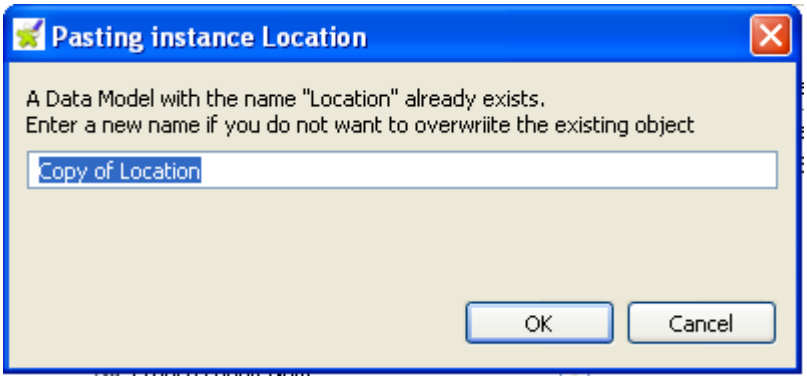
To avoid creating a data model from scratch, you can copy an existing one in the **MDM Repository** tree view and modify its metadata to have a new data model.

Prerequisite(s): You have already connected to the MDM server from *Talend Open Studio for MDM*. At least one data model exists.

To copy/paste a data model, do the following:

1. In the **MDM Repository** tree view, expand the **Data Model** node.
2. Right-click the data model you want to duplicate and select **Copy** from the contextual menu.
3. Right-click the **Data Model** node and select **Paste** from the contextual menu.

   A dialog box displays prompting you to enter a name for the new data model.



4. Enter a name for the new data model and click **OK** to validate the changes and close the dialog box.

   The new data model is listed under the **Data Model** node in the **MDM Repository** tree view.

3.2.3.5. How to duplicate a data model

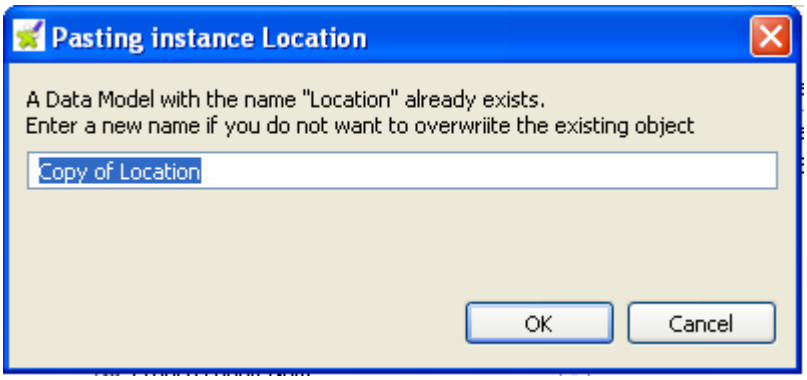
To avoid creating a data model from scratch, you can duplicate an existing one in the **MDM Repository** tree view and modify its metadata to have a new data model.

Prerequisite(s): You have already connected to the MDM server from *Talend Open Studio for MDM*. At least one data model exists.

To duplicate a data model, do the following:

1. In the **MDM Repository** tree view, expand the **Data Model** node.
2. Right-click the data model you want to duplicate and select **Duplicate** from the contextual menu.

   A dialog box displays prompting you to enter a name for the new data model



3. Enter a name for the new data model and click **OK** to validate the changes and close the dialog box.

   The new data model is listed under the **Data Model** node in the **MDM Repository** tree view.

You can also duplicate the data object if you drop it onto its parent node in the **MDM Repository** tree view.

3.2.3.6. How to delete a data model, a business entity or an attribute

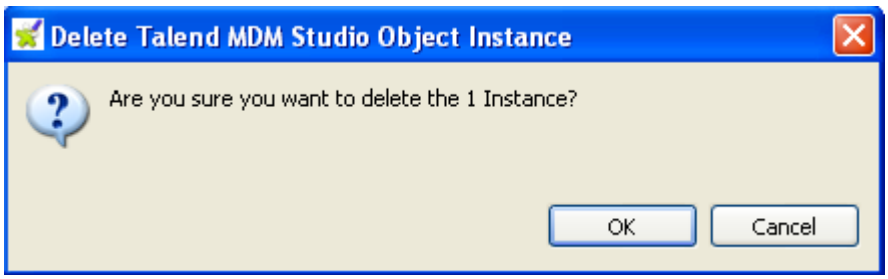
You can delete any of the data models, business entities or attributes you create through a simple right-click on the selected item.

Prerequisite(s): You have already connected to the MDM server from *Talend Open Studio for MDM*. At least one data model, business entity or attribute exists.

To delete a data model:

1. In the **MDM Repository** tree view, expand the **Data Model** node.
2. Right-click the data model you want to delete and select **Delete** from the contextual menu.

   A confirmation dialog box displays prompting you to confirm the deletion operation or to cancel it.



3. Click **OK** to close the dialog box and delete the selected data model from the **MDM Repository** tree view.

To delete a business entity or an attribute in a data model:

1. In the **MDM Repository** tree view, expand the **Data Model** node and double-click the data model from which you want to delete a business entity or an attribute.

   The corresponding editor opens on the selected data model.
2. Right-click the business entity or the attribute in the business entity you want to delete and select the delete option relevant to your selection.

   The selected business entity or attribute is deleted from the data model editor.

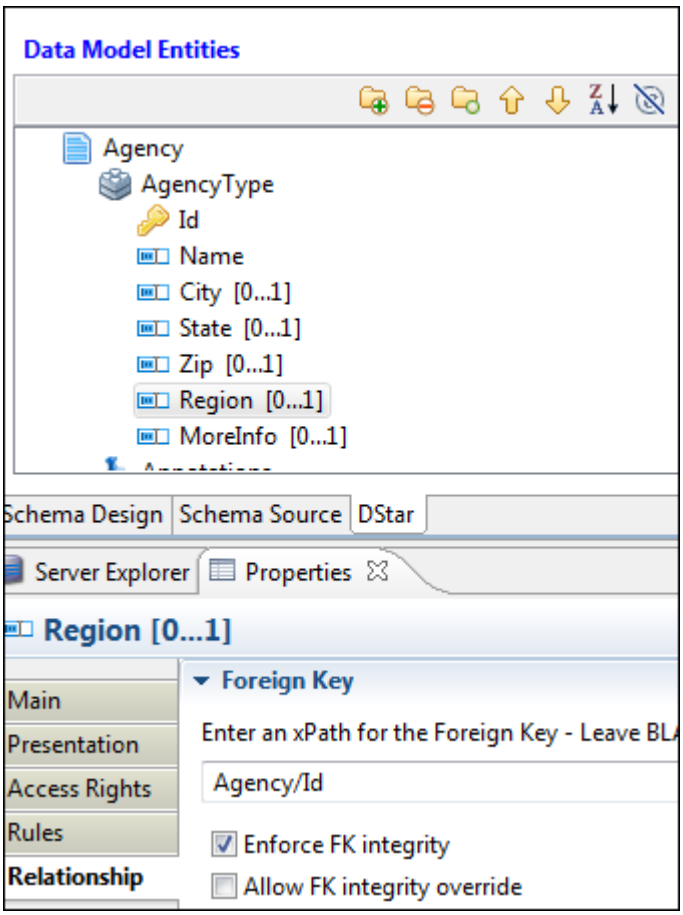
3.2.3.7. How to enable foreign key integrity checking

You can enable foreign key integrity checking to manage whether users are allowed to delete a record that is linked to another record through a foreign key. By default, this is not allowed.

To enable foreign key integrity checking:

1. In the **MDM Repository** tree view, expand the **Data Model** node and double-click the data model for which you want to enable foreign key integrity checking.

   The corresponding editor opens with the selected data model displayed.



2. Select the entity containing the foreign key for which you want to enable integrity checking.
3. In the **Properties** view, click the **Relationship** tab.
4. Under the **Foreign Key** section:
   - Select the **Enforce FK integrity** checkbox to enable foreign key integrity checking, that is, to raise an error whenever a user tries to delete a record if the record is linked to any other records through this foreign key.
   - Select the **Allow FK integrity override** checkbox to allow users to force the delete of a record even if it is linked to another record through this foreign key.

These two options can be combined in the following ways.

	Enforce FK integrity enabled	**Enforce FK integrity disabled**
Allow FK integrity override enabled	Delete allowed, with user prompted for override	Delete allowed
Allow FK integrity override disabled	Delete forbidden	Delete allowed

In the case of conflicts, the least tolerant rule is always applied.

If a user is prevented from deleting a record, details are provided in *<jboss-dir>/server/default/log/server.log*.

Handling circular dependencies

If you are using a SQL database to store your data records, you may encounter an error during initialization because a circular dependency is detected. This means that, for example, *EntityA* references *EntityB*, *EntityB* references *EntityC*, and *EntityC* references *EntityA*.

To fix the data model in such a case, simply turn off Foreign Key integrity checking for the relation, as described in section *How to enable foreign key integrity checking*.

3.3. Data Containers

Talend Open Studio for MDM allows you to persist master data in one or several containers within a single MDM Hub. Data containers are then “partitioned” to help you organize the master data, typically by domains.

A data container in the MDM Hub can hold data from multiple entities. An entity in a data container is not visible from another data container.

You can create, import/export, edit, copy/paste, duplicate and delete data containers.

The following sections detail the procedural steps to carry out each of the above management options.

3.3.1. Creating a data container

From *Talend Open Studio for MDM* you can create one or several data containers in the MDM Hub in which you can load data using the **tMDMInput** and **tMDMOutput** components.

For more information on the MDM components, see *Talend Open Studio Components Reference Guide*.

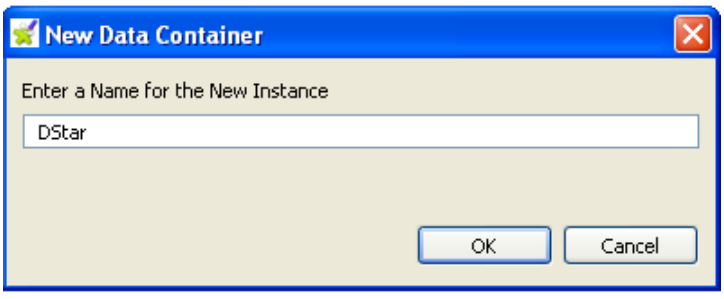
Later, a business user can write/edit data in the created container using *Talend MDM Web User Interface*. For more information, see *Talend MDM Web User Interface User Guide*.

Prerequisite(s): You have already connected to the MDM server from *Talend Open Studio for MDM*.

To create a data container, do the following:

1. In the **MDM Repository** tree view, right-click **Data Container** and select **New** from the contextual menu.

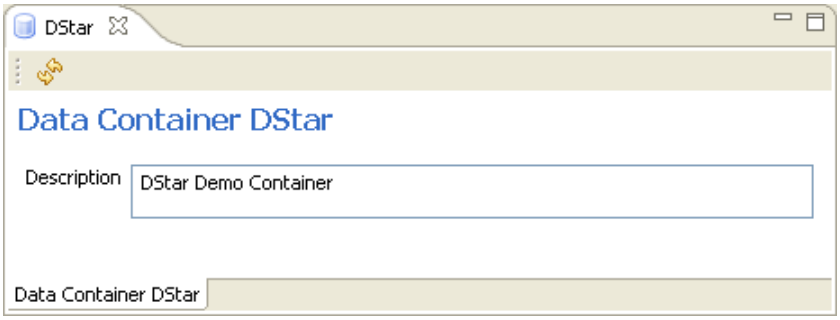
   The **[New Data Container]** dialog box displays.



2. Enter a name for the new data container and then click **OK** to close the dialog box.

   A editor for the newly created data container opens in the workspace.

No spaces are allowed in the data container name. The OK button in the dialog box will be unavailable as long as there is a space in the instance name.



3. If needed, enter a description for the newly created data container and then click the save icon on the toolbar or press **Ctrl + S** on your keyboard to save your changes.

The newly created data container is listed under the **Data Container** node in the **MDM Repository** tree view.

In order to populate the data container with your data, you can use the **Integration** perspective in the Studio to design a Job using **Talend** MDM components. For further information on the MDM components, see *Talend Open Studio Components Reference Guide*. For an MDM Demo example, see *http://talendforge.org/wiki/doku.php?id=mdmce:demo*.

3.3.2. Managing data containers

An authorized user can also import/export, copy/paste and delete created data containers from *Talend Open Studio for MDM*.

It is also possible to import and share MDM complete projects or only a data model or part of the data model from the community web page, **Talend Exchange**. For further information, see section *Projects/objects on Talend Exchange*.

3.3.2.1. How to browse a data container

Using *Talend Open Studio for MDM*, you can easily search and locate data records “attached” to the corresponding business entities in the MDM Hub.

More than one search option is available to specify the search criteria and to narrow down your search for the data records you want to view. You can:

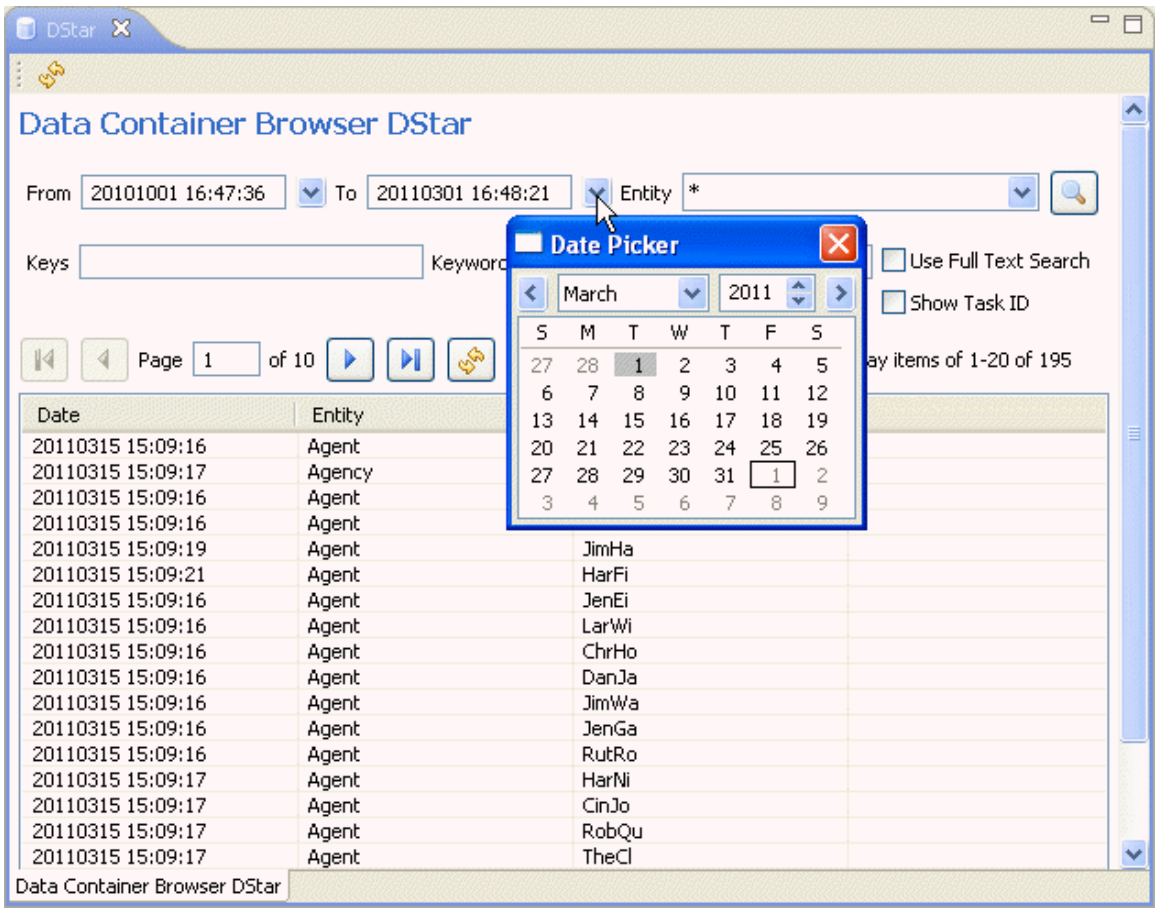
- use a time-based filter (from/to dates),
- select a business entity in which to search data records,
- type in the unique key for the business entity you want to search for,
- type in a search keyword.

Prerequisite(s): You have already connected to the MDM server in *Talend Open Studio for MDM*. A data container exists and is populated with data records.

To browse a data container:

1. In the **MDM Repository** tree view, expand the **Data Container** node.
2. Double-click the data container you want to browse.

   The data container editor displays.



You can set one or several search criteria to narrow your search on the returned data records. Note that, if you are using a SQL database, you cannot combine a full-text query with other search criteria.

3. Click the **From** arrow to display a calendar where you can select a start date for the data records created in the selected data container.

4. Click the **To** arrow to display a calendar where you can select an end date for the data records created in the selected data container.

It is possible to click the browse button directly after setting the time range to base your search only on one criterion. Otherwise, you can use one or more of the other search criteria to narrow down your search on the returned data records.

5. From the **Entity** list, select a business entity from the list to search data records only in the selected business entity.

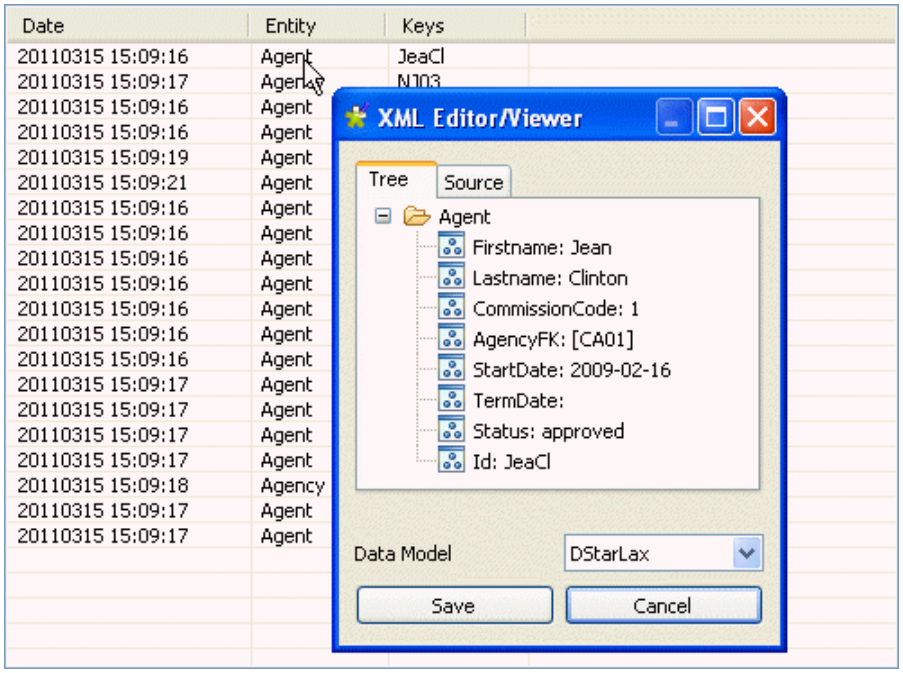
6. In the **Keys** field, enter the unique key for the entity you want to search.

7. In the **Keywords** field, enter a search keyword to include only matching values in the returned data records.

8. Click the  icon.

   The lower part of the editor lists all data records that match the search criteria along with their creation dates, the business entities they are created in and their unique keys.

9. Double-click a given record to open up a dialog box that displays the data record details.



You can click the **Source** tab at the top of the dialog box to view the data record in its XML format.

3.3.2.2. How to export data containers

From *Talend Open Studio for MDM* you can export one or multiple data containers in order to exchange them between:

- two different MDM Servers or Repositories,
- two different Versions from the same/different MDM Servers or Repositories, for example.

The steps to export one or multiple data containers are similar to those for any other data object in the **MDM Repository** tree view. For detailed information on how to export data containers, see section *How to export data models*.

3.3.2.3. How to import data containers

From *Talend Open Studio for MDM* you can import data containers created on other MDM servers, in different Versions on the same MDM server, or in different MDM Repositories into the current MDM Repository.

The steps to import one or multiple data containers are similar to those for any other data object in the **MDM Repository** tree view. For detailed information on how to import data containers, see section *How to import data models*.

3.3.2.4. How to copy/paste a data container

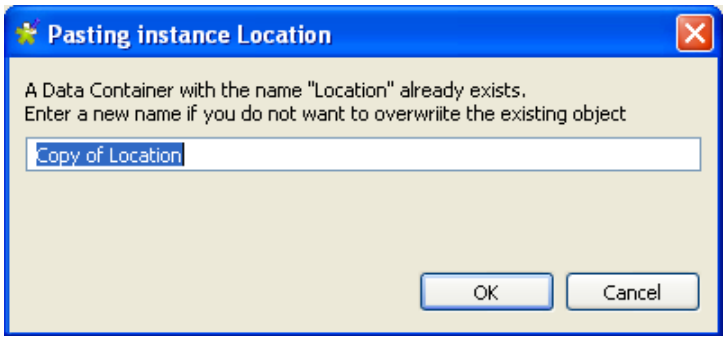
To avoid creating one or multiple data containers from scratch, you can copy an existing one in the **MDM Repository** tree view and decide what business entities to copy from the source data container onto the target data container.

Prerequisite(s): You have already connected to the MDM server in *Talend Open Studio for MDM*. At least one data container exists.

To copy/paste a data container, do the following:

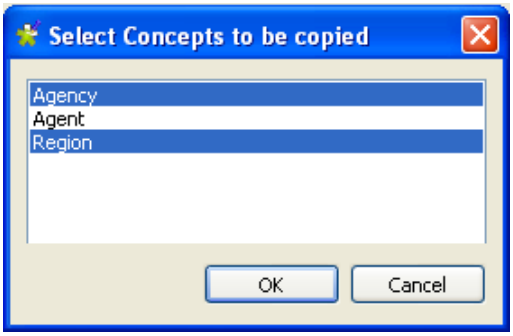
1. In the **MDM Repository** tree view, expand the **Data Container** node.
2. Right-click the container you want to duplicate and select **Copy** from the contextual menu.
3. Right-click the **Data Container** node and select **Paste** from the contextual menu.

   A dialog box displays prompting you to enter a name for the new container.



4. Enter a name for the new data container and click **OK** to validate the changes and close the dialog box.

   A dialog box displays. This dialog box list all the business entities held in the source data container.



5. Select the business entities you want to copy in the new data container and then click **OK**.

   The new data container that holds the selected business entities is listed under the **Data Container** node in the **MDM Repository** tree view.

3.3.2.5. How to duplicate a data container

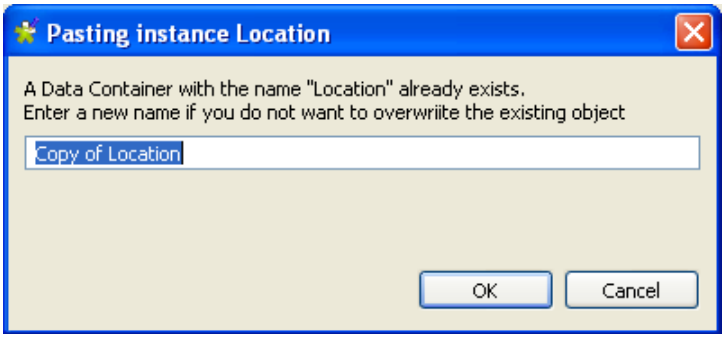
To avoid creating one or multiple data containers from scratch, you can duplicate an existing one in the **MDM Repository** tree view and decide what business entities to copy from the source data container onto the target data container.

Prerequisite(s): You have already connected to the MDM server in *Talend Open Studio for MDM*. At least one data container exists.

To duplicate a data container, do the following:

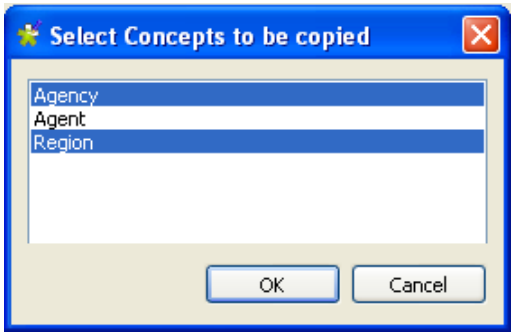
1. In the **MDM Repository** tree view, expand the **Data Container** node.
2. Right-click the container you want to duplicate and select **Duplicate** from the contextual menu.

   A dialog box displays prompting you to enter a name for the new container.



3. Enter a name for the new data container and click **OK** to validate the changes and close the dialog box.

   A dialog box displays. This dialog box list all the business entities held in the source data container.



4. Select the business entities you want to copy in the new data container and then click **OK**.

   The new data container that holds the selected business entities is listed under the **Data Container** node in the **MDM Repository** tree view.

You can also duplicate the data object if you drop it onto its parent node in the **MDM Repository** tree view.

3.3.2.6. How to delete a data container

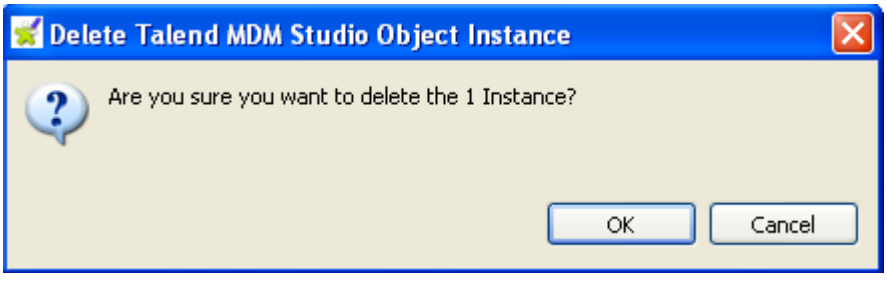
You can delete a data container you create through a simple right-click on the selected data container.

Prerequisite(s): You have already connected to the MDM server in *Talend Open Studio for MDM*. At least one data container exists.

To delete a data container:

1. In the **MDM Repository** tree view, expand the **Data Container** node.
2. Right-click the data container you want to delete and select **Delete** from the contextual menu.

   A confirmation dialog box displays prompting you to confirm the deletion operation or to cancel it.



3. Click **OK** to close the dialog box and delete the selected data container from the **MDM Repository** tree view.

3.4. Views

A View is one of the possible ways to query the MDM Hub. When you create Views in *Talend Open Studio for MDM*, you specify:

- What business entities are viewable/searchable for an authorized business user,
- What elements a business user can view/search in the business entities.
- What elements of the business entities can be returned by the query.
- What content can be returned by the query. The data can be filtered on its content before being delivered by the View (for example, all companies which have their Head Office in France).

3.4.1. Creating a View

From *Talend Open Studio for MDM* you can easily create simple views over the same business entity or composite Views over multiple business entities held in the same data model.

You can also create multiple simple Views over the same business entity to allow users to have different predefined accesses on the records by defining different search criteria in each View. The # character is used to create multiple views on the same entity; for example two views on the *Agency* entity may have the following names: *Browse_items_Agency* and *Browse_items_Agencynt#UK* where the second View will give access only to agencies based in the United Kingdom.

Note also that, for improved readability, in the MDM Repository view, Views are stored in subfolders by type and only the second part of the name is displayed.

Creating a simple View on a business entity allows a business user inside *Talend MDM Web User Interface* to visualize specific elements in this entity according to the defined criteria. For more information on how to view/search elements in a specific business entity, see the *Talend MDM Web User Interface User Guide*.

Creating a composite View on several business entities allows a business user inside *Talend MDM Web User Interface* to visualize specific elements in multiple entities according to a join clause. A composite View can provide greater insight on master data than a simple View. For more information, see *Talend MDM Web User Interface User Guide*.

It is also possible to attach a Process to the View in order to transform/enrich data, on the fly, by looking up fields in the source system. For further information, see section *Running the view result through a Process (registry style lookup)*.

The below sections explains how to create each of these two types of Views.

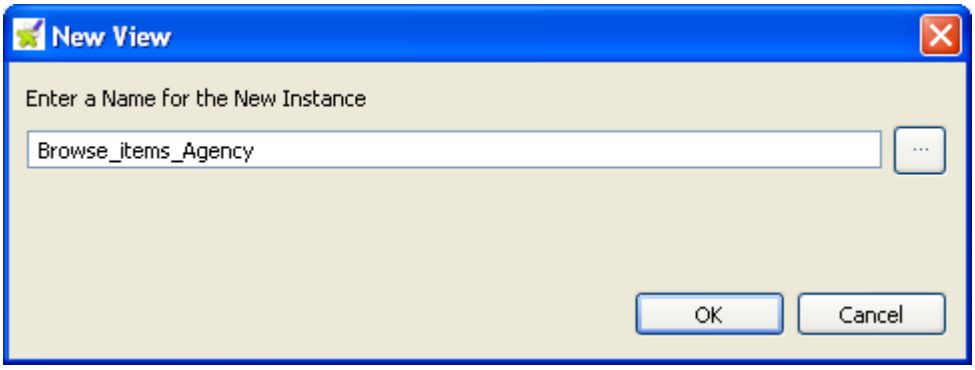
3.4.1.1. How to create a simple View

Prerequisite(s): You have already connected to the MDM server from *Talend Open Studio for MDM*.

To create a simple View, do the following:

1. In the **MDM Repository** tree view, right-click **View** and select **New** from the contextual menu.

   The **[New View]** dialog box displays.



As the simple View is concerned with only one specific entity, its name must follow a specific pattern that includes the name of the entity: *Browse_items_<name of the business entity>*. The dialog box displays with the by-default text *Browse_items*. However, you can use the # character to create multiple views on the same entity; for example two views on the *Agency* entity may have the following names: *Browse_items_Agency* and *Browse_items_Agencynt#UK* where the second View will give access only to agencies based in the United Kingdom.

Note also that, for improved readability, in the MDM Repository view, Views are stored in subfolders by type and only the second part of the name is displayed.

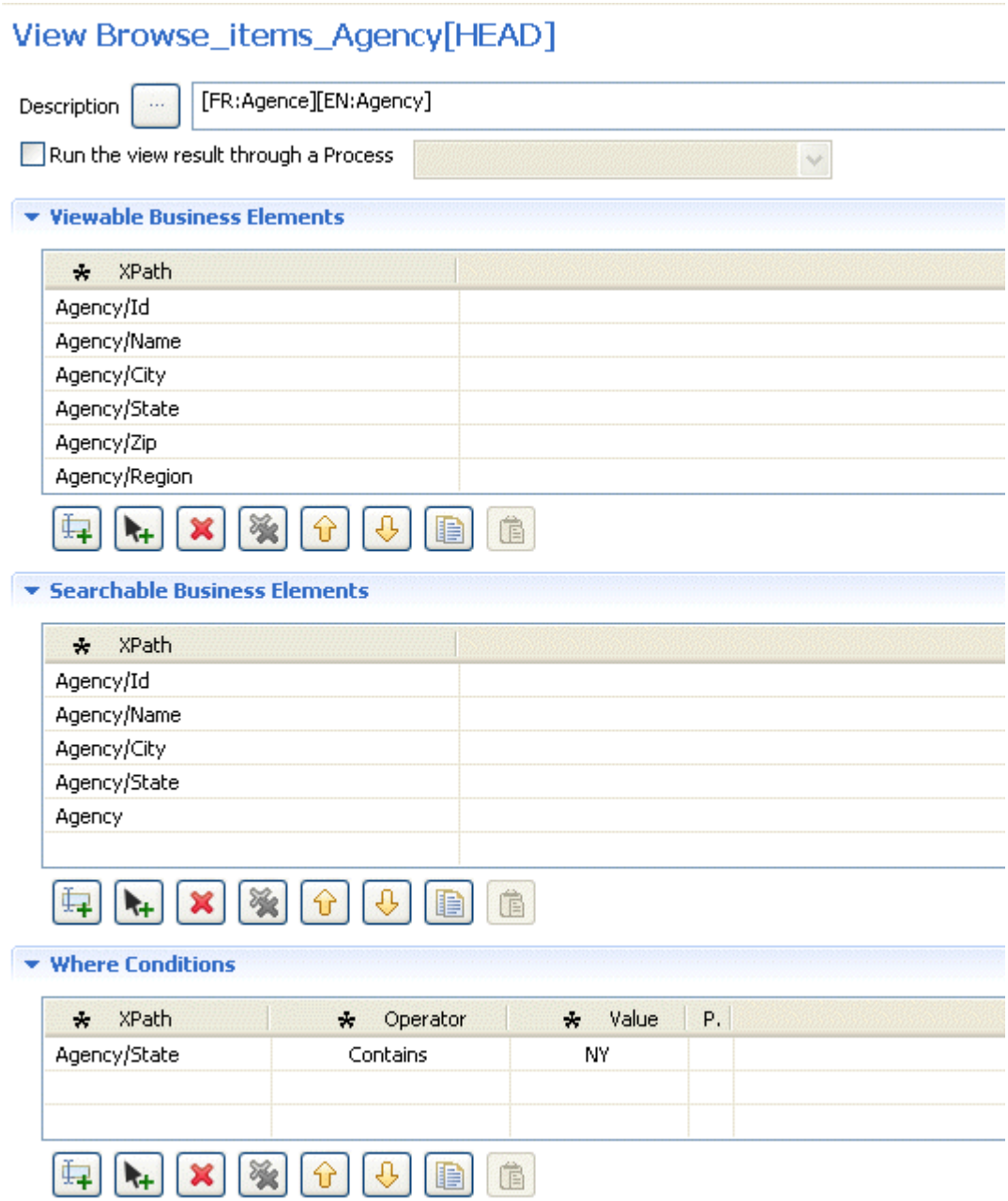
2. Enter a name for the new View and then click **OK** to close the dialog box.

3. Click the three-dot button to open a dialog box where you can select the entity for which you want to create a View and then click **OK** to close the dialog box.

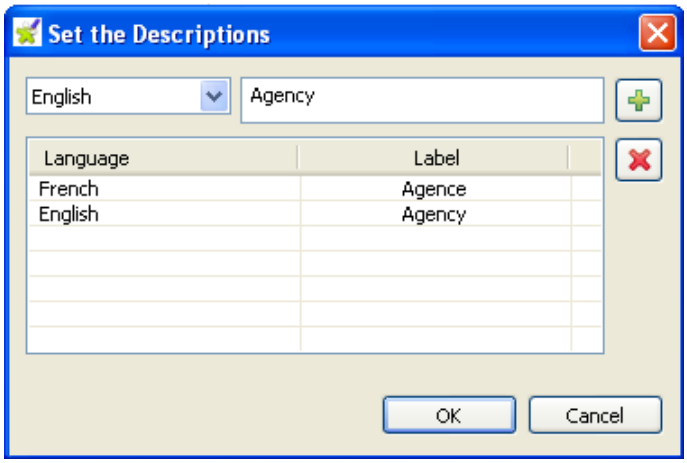
   An editor for the newly created View opens in the workspace.

You can automate the above steps to create a default View. Select one or more entities in the data model, right-click the selection and then click **Generate Default Browse Items Views**. This creates a default View which is listed under the **View** node in the **MDM Repository** tree view. By default, the Viewable Business Elements and Searchable Business Elements areas in the default View are populated with the top five elements.

You can then proceed as below in order to define the View criteria.



4. Click the three-dot button next to **Description** to open a dialog box where you can add multilingual labels to the new View.

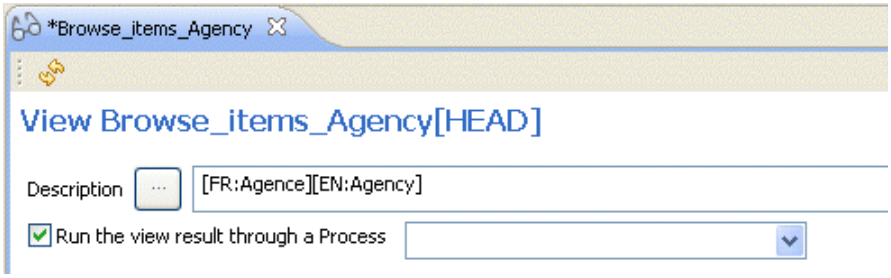


5. From the list to the left, select a language and enter the corresponding label in the field to the right.

   Click the  button to add the description to the **Language/Label** list.

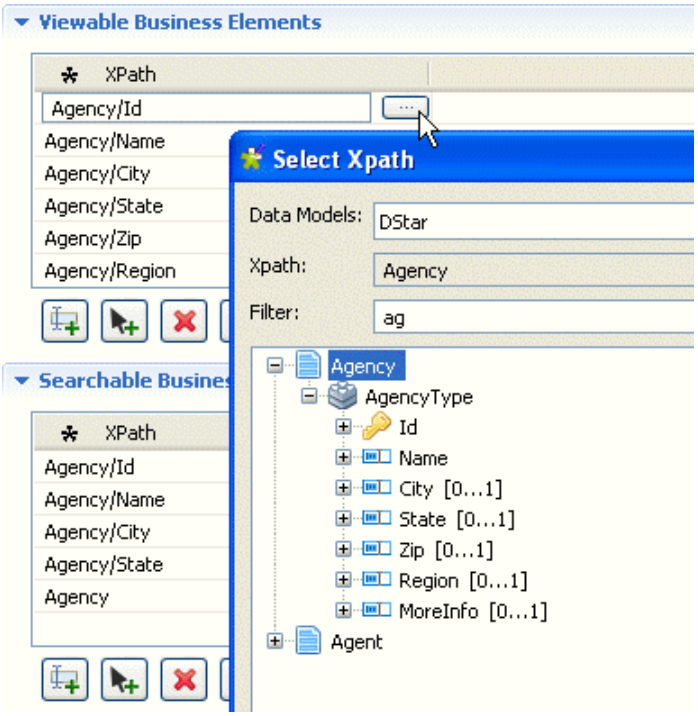
   Repeat the operation to add as many labels as needed and click **OK** to close the dialog box.

   All defined labels display in the **Description** field.



6. If required, select the **Run the view result through a Process** check box and then select from the list the Process you want to attach to the current View. The Process will be used to transform/enrich data, on the fly, by looking up fields in the source system.For further information, see section *Running the view result through a Process (registry style lookup)*.

7. In the **Viewable Business Elements** area, click the  button to add a new **XPath** line to the table.



8. In the new **XPath** line, click the three-dot button to open a dialog box.

   By default, only the selected entity of all the entities that constitute the data model will be listed in the dialog box and its name will display in the **Filter** field.

   You can list multiple entities in the dialog box if you type in the **Filter** field the text according to which you want to filter business entities. This is a search-as-you-type filter where entities are instantly filtered as you type your search string.

9. Select the element to be retrieved by the View and repeat the procedure to add as many elements as needed.

   You can also retrieve the foreign key information and display it to the business user in the **[Browse Records]** page in *Talend MDM Web User Interface*. For further information, see section *How to create a simple View that displays the foreign key information*.

10. In the **Searchable Business Elements** area, click the button to add a new **XPath** line to the table.

11. In the new **XPath** line, click the three-dot button to open a dialog box where you can select the element to be located.

    By default, only the selected entity of all the entities that constitute the data model will be listed in the dialog box and its name will display in the **Filter** field.

    You can list multiple entities in the dialog box if you type in the **Filter** field the text according to which you want to filter business entities. This is a search-as-you-type filter where entities are instantly filtered as you type your search string.

12. In the **Where Conditions** area, click the button to add a new **XPath** line to the table.

13. In the new **XPath** line, click the three-dot button to open a dialog box where you can select the element you want to define conditions on.

    You can list multiple entities in the dialog box if you type in the **Filter** field the text according to which you want to filter business entities. This is a search-as-you-type filter where entities are instantly filtered as you type your search string.

You can define conditions with various operators listed in the **Operator** column. You can also join elements using different operators listed in the **Predicate** column.

14. Click the save icon on the toolbar or press **Ctrl + S** on your keyboard to save your changes.

The newly created View is listed under the **View** node in the **MDM Repository** tree view.

Using *Talend MDM Web User Interface*, an authorized business user can now visualize specific records in this entity according to the defined criteria. For more information on how to view/search elements in a specific business entity, see the *Talend MDM Web User Interface User Guide*.

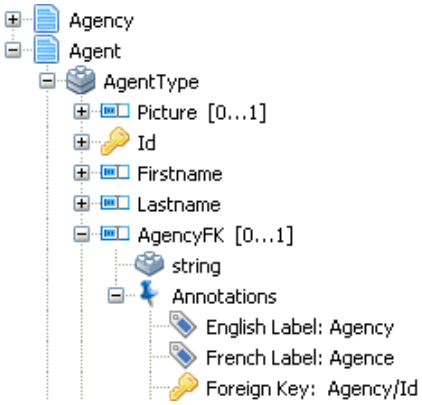
You can use the  and  icons in the view editor to copy/paste items among **Viewable Business Elements**, **Searchable Business Elements** and **Where Conditions** tables.

3.4.1.2. How to create a simple View that displays the foreign key information

The simple views you create on business entities in *Talend Open Studio for MDM* enables business users using *Talend MDM Web User Interface* to visualize specific records/attributes in this entity according to the defined criteria.

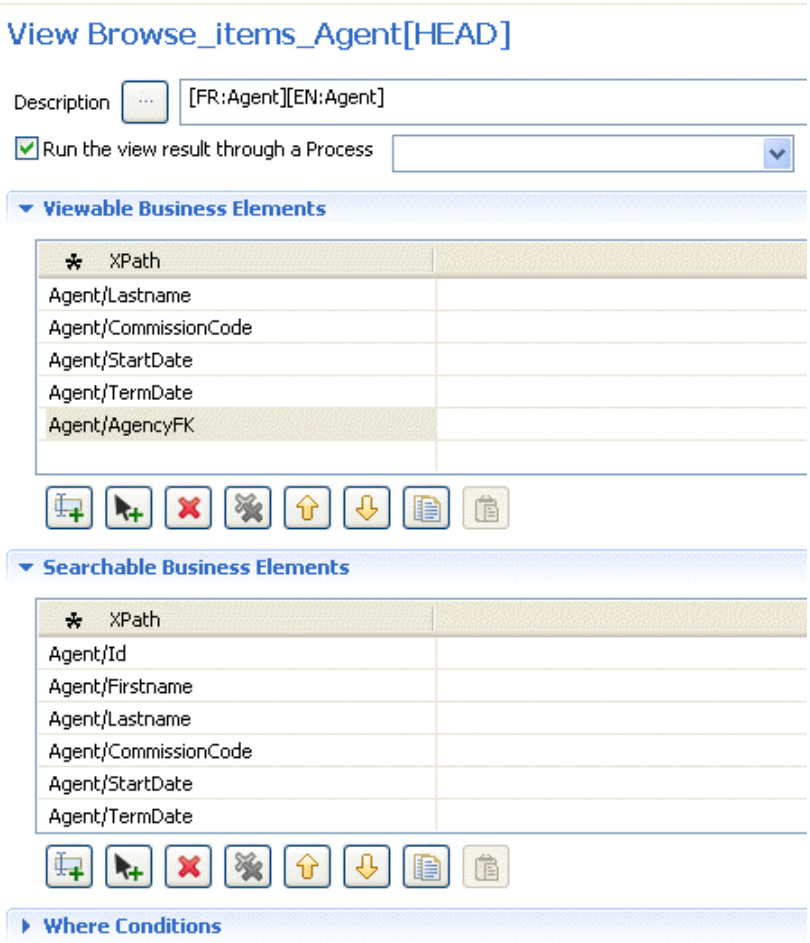
Prerequisite(s): You have already connected to the MDM server from *Talend Open Studio for MDM*.

Consider as an example that your data model holds the *Agency* and *Agent* entities. You have added a foreign key attribute, *AgencyFK*, in the *Agent* entity that points to the *Agency* entity. This attribute will link every agent to an agency. For further information, see section *How to add a foreign key: linking entities together*.

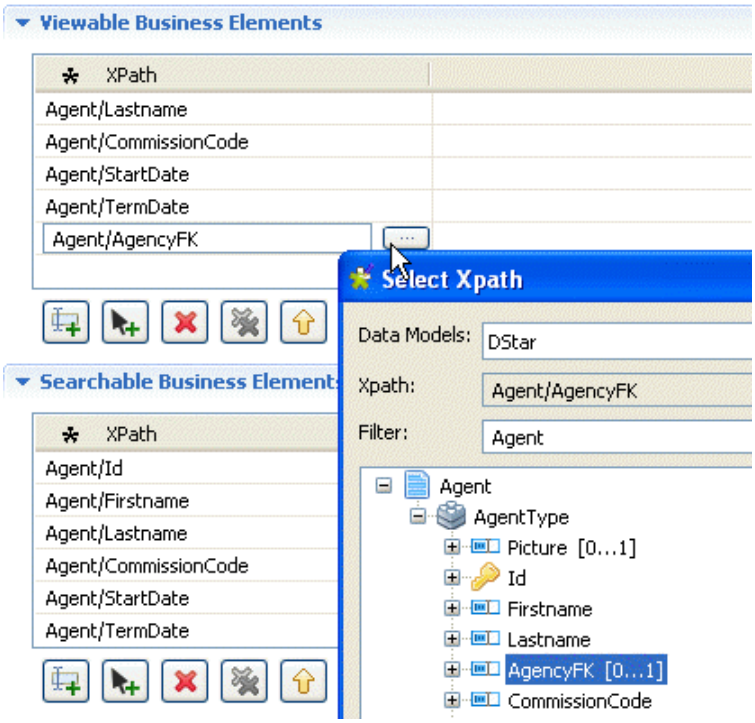


To create a simple View that displays the foreign key information, do the following:

1. Follow the steps outlined in section *How to create a simple View* to create a simple view on the *Agent* entity.



2. In the **Viewable Business Elements** table, click the  button to add a new **XPath** line.



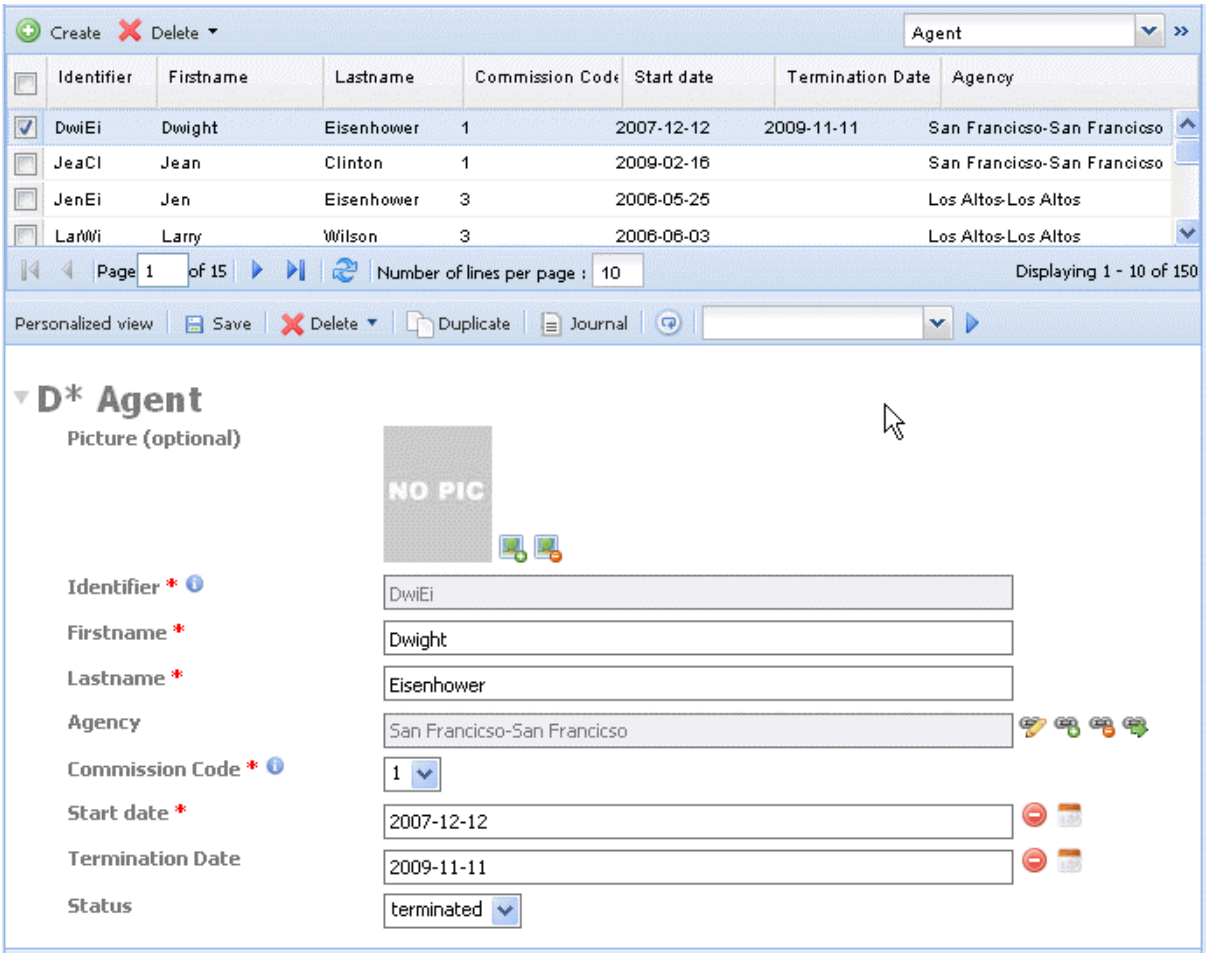
3. In the new **XPath** line, click the three-dot button to open a dialog box.

4. Select the foreign key element to be retrieved by the View, *AgencyFK* in this example.

5. Click the save icon on the toolbar or press **Ctrl + S** on your keyboard to save your View.

The newly created View is listed under the **View** node in the **MDM Repository** tree view.

Using *Talend MDM Web User Interface*, an authorized business user can now visualize the agent records with all the elements you defined in the View including the agency to which each agent is attached.



For more information on how to view/search elements in a specific business entity, see *Talend MDM Web User Interface User Guide*.

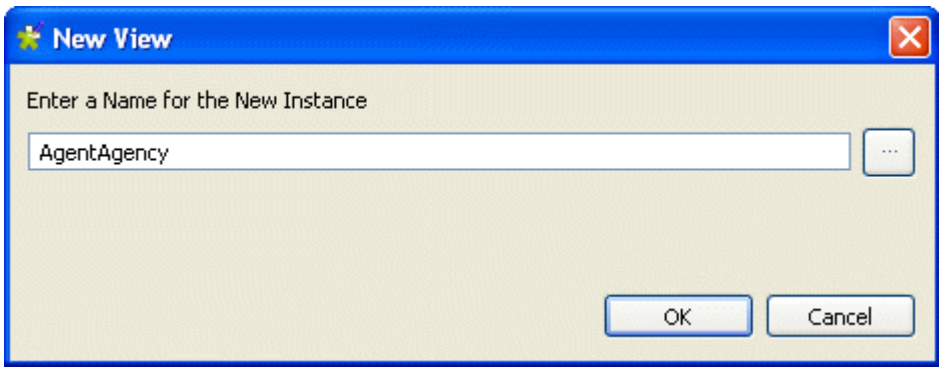
3.4.1.3. How to create a composite View

Prerequisite(s): You have already connected to the MDM server from *Talend Open Studio for MDM*.

To create a composite View, do the following:

1. In the **MDM Repository** tree view, right-click **View** and select **New** from the contextual menu.

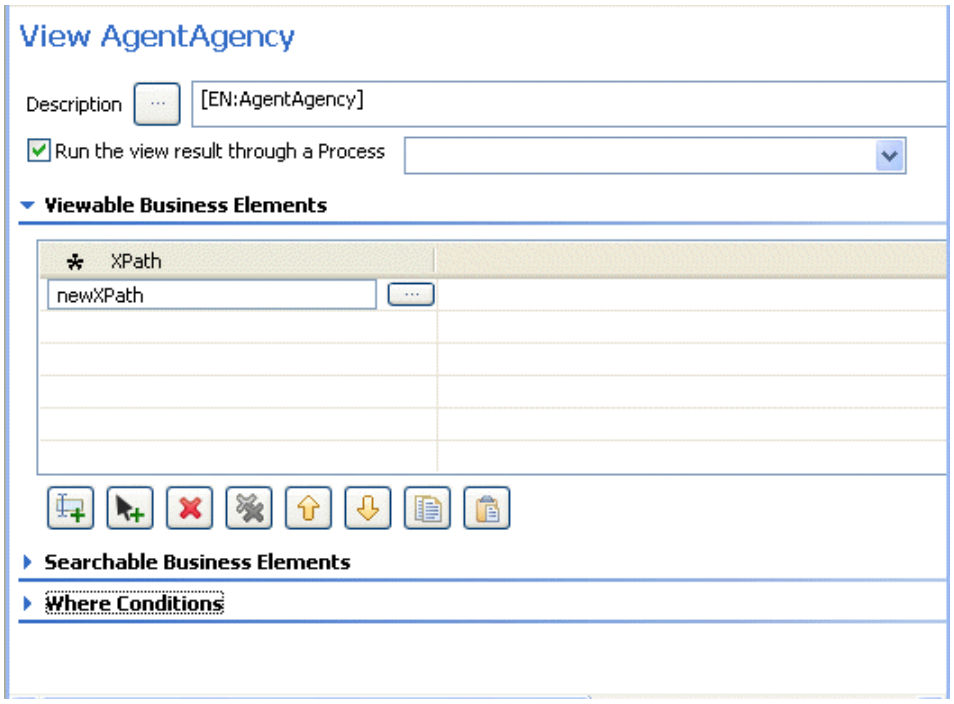
   The **[New View]** dialog box displays.



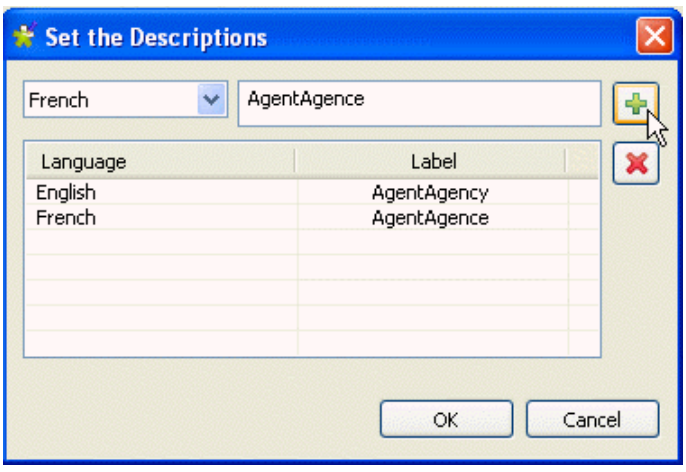
As the composite View is not concerned with only one specific entity, its name does not have to follow a specific pattern as it is the case with the simple View name.

2. Enter a name for the new View and then click **OK** to close the dialog box, *AgentAgency* in this example.

   An editor for the newly created View opens in the workspace.



3. Click the three-dot button next to **Description** to open a dialog box where you can add multilingual labels to the new View.

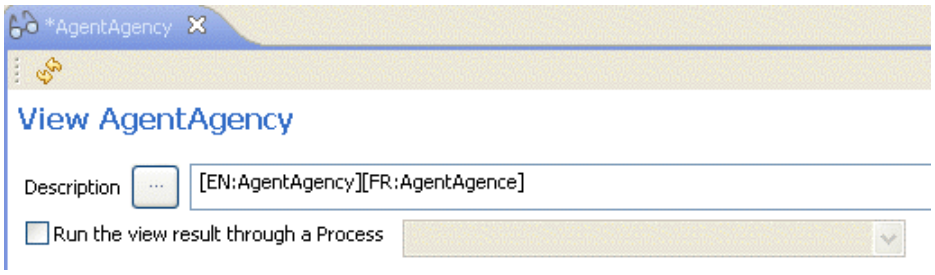


4. From the list to the left, select a language and enter the corresponding label in the field to the right.

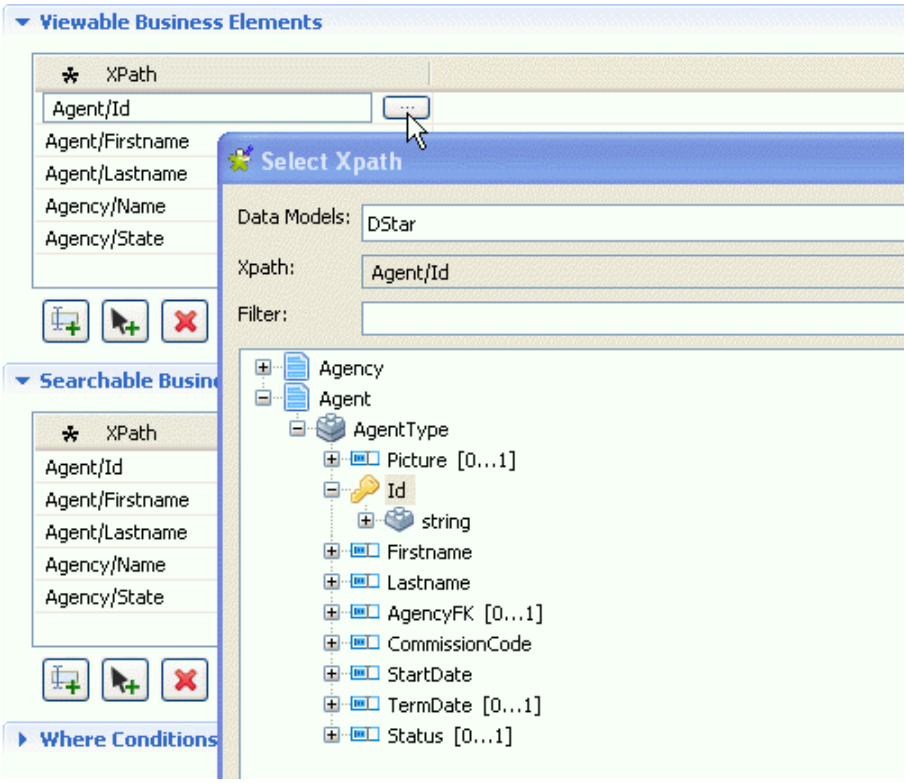
   Click the  button to add the description to the **Language/Label** list.

   Repeat the operation to add as many labels as needed and click **OK** to close the dialog box.

   All defined labels display in the **Description** field.



5. In the **Viewable Business Elements** area, click the  button to add a new **XPath** line to the table.



6. In the new **XPath** line, click the three-dot button to open a dialog box.

   From the **Data Models** list, select the data model holding the entities on which you want to create the composite View.

   All entities constituting the model are listed in the dialog box.

   You can filter the entities you want to display in the dialog box if you type in the **Filter** field the text according to which you want to filter business entities. This is a search-as-you-type filter where entities are instantly filtered as you type your search string.

7. Select the entities/element to be retrieved by the View being defined.

8. Consider as an example that your data model has the two entities: *Agency* and *Agent*. You want to create a composite View that joins elements in both entities to show multiple data records for the same agent if he/she is attached to multiple agencies.

   In the **Searchable Business Elements** area, click the  button to add a new **XPath** line to the table.

9. In the new **XPath** line, click the three-dot button to open a dialog box where you can select the entities or elements to be located.

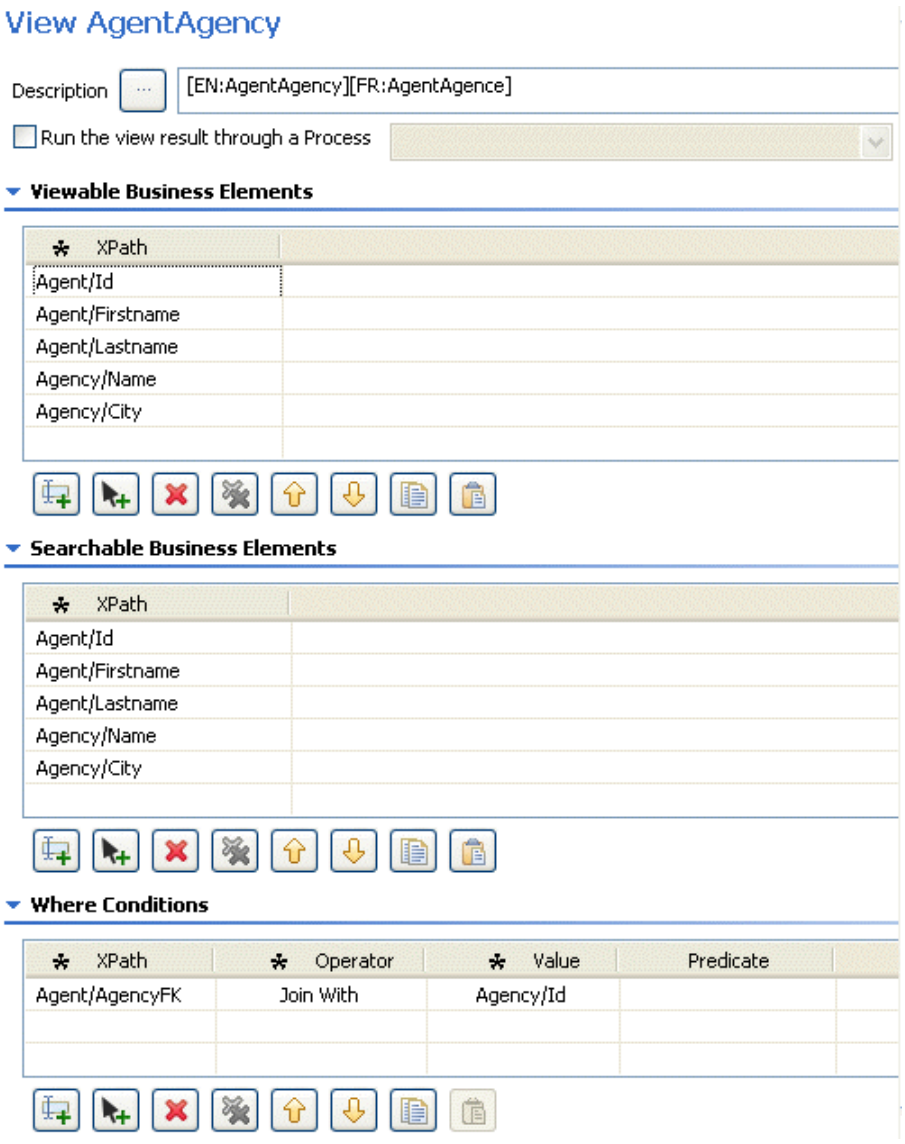
   You can filter the entities you want to display in the dialog box if you type in the **Filter** field the text according to which you want to filter business entities. This is a search-as-you-type filter where entities are instantly filtered as you type your search string.

10. In the **Where Conditions** area, click the  button to add a new **XPath** line to the table.

    In the new **XPath** line, click the three-dot button to open a dialog box where you can select the entities/elements on which you want to define a join condition.

    You can filter the entities you want to display in the dialog box if you type in the **Filter** field the text according to which you want to filter business entities. This is a search-as-you-type filter where entities are instantly filtered as you type your search string.



You must use the "Join With" operator when creating a composite View in order to specify the join in the WHERE clause. Usually, you join on elements keys. If there are more than one keys to join on, you must repeat a "Joins With" clause for each part of the key. For instance, suppose the AgencyFK is a compound key that comprises Id and Date, you must define two conditions in the "Where Condition" table: "Agent/AgencyFK --> Joins With --> Agency/Id" and "Agent/AgencyFK --> Joins With --> Agency/Date".

The WHERE clause used in this example will generate multiple data records for the same agent linked with different agencies.

11. Click the save icon on the toolbar or press **Ctrl + S** on your keyboard to save your changes.

The newly created View is listed under the **View** node in the **MDM Repository** tree view.

Using *Talend MDM Web User Interface*, an authorized business user can now visualize the records in the two specified entities according to the defined join clause. For more information, see the *Talend MDM Web User Interface User Guide*.

You can attach a Process to the View in order to transform/enrich data, on the fly, by looking up fields in the source system. For further information, see section *Running the view result through a Process (registry style lookup)*.

3.4.2. Running the view result through a Process (registry style lookup)

Talend Open Studio for MDM enables you to enrich data, on the fly, by looking up fields in the source system through a Process attached to the View, whether it is a simple or a composite View. Using Processes with Views will help optimizing system performance.

or example, when an interaction is done on a business entity that is not persisted or not totally persisted in the MDM Hub, the MDM Hub will be able to resolve specific elements for this entity by referring to the source system. You can achieve this behavior if you:

- indicate what elements in the business entity or entities are to be transformed or enriched by the Process,
- create the corresponding Process,
- create a View on the viewable/searchable elements in the business entity and attach the Process to this View that will run the view result.

When a steward or a business user connects to *Talend MDM Web User Interface* and browse the records/Views of the business entity/entities, the Process will lookup the defined elements (values) from the source system and display them in the web user interface without really saving them in the MDM Hub.

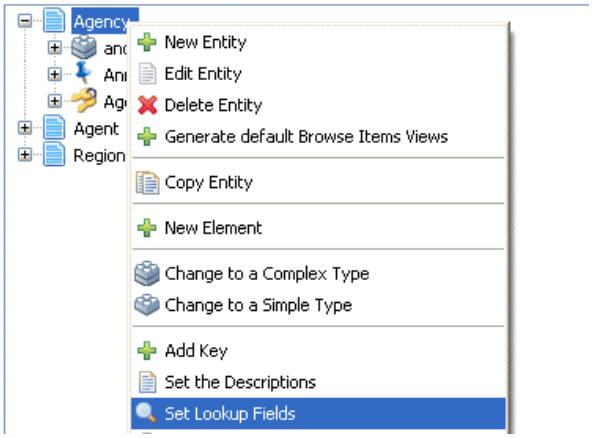
3.4.2.1. How to define the elements to be transformed/enriched by a Process

Consider as an example that you want in your data model to retrieve the agency region, on the fly, from the source system without really saving the region in the MDM Hub.

Prerequisite(s): You have already connected to the MDM server from *Talend Open Studio for MDM*. A data model has been already created.

To define the entity elements to be transformed/enriched by a Process, do the following:

1. In the **MDM Repository** tree view, expand the **Data Model** node and double-click the data model in which you want to define the elements to be handled by the Process.



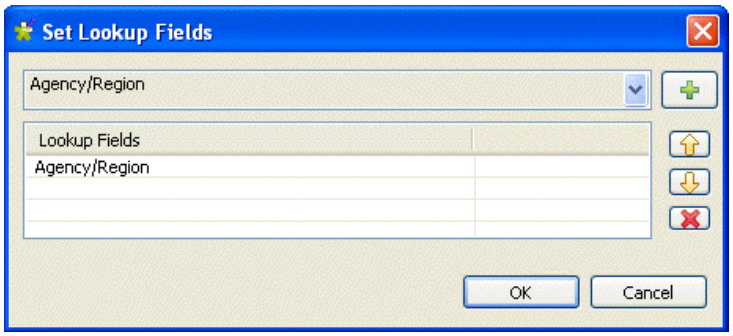
In this example, you want to set a lookup field on the *Agency* entity in order for a corresponding Process to retrieve the agency region from the source system.

2. In the **Data Model Entities** panel, right-click the entity from which you want to select the element(s) to be handled by the Process.

   A contextual menu displays.

3. Select **Set Lookup Field**.

   The **[Set Lookup Fields]** dialog box displays.



4. Click the arrow and select from the list one of the elements pertaining to the selected entity.

5. Click the  icon in the upper right corner to add the selected element to the **Lookup Fields** list.

6. Do the same to add as many elements as necessary to the **Lookup Fields** list.

   Only listed elements will be transformed/enriched on the fly by the Process. If this Process modifies anything else in the XML record, these modifications will be ignored at the entity level because the Process will send only the specified elements and not all the XML record.

7. Click **OK** to close the dialog box.

   The defined look up field(s) display(s) under the **Annotations** node corresponding to the entity selected in the **Data Model Entities** panel.

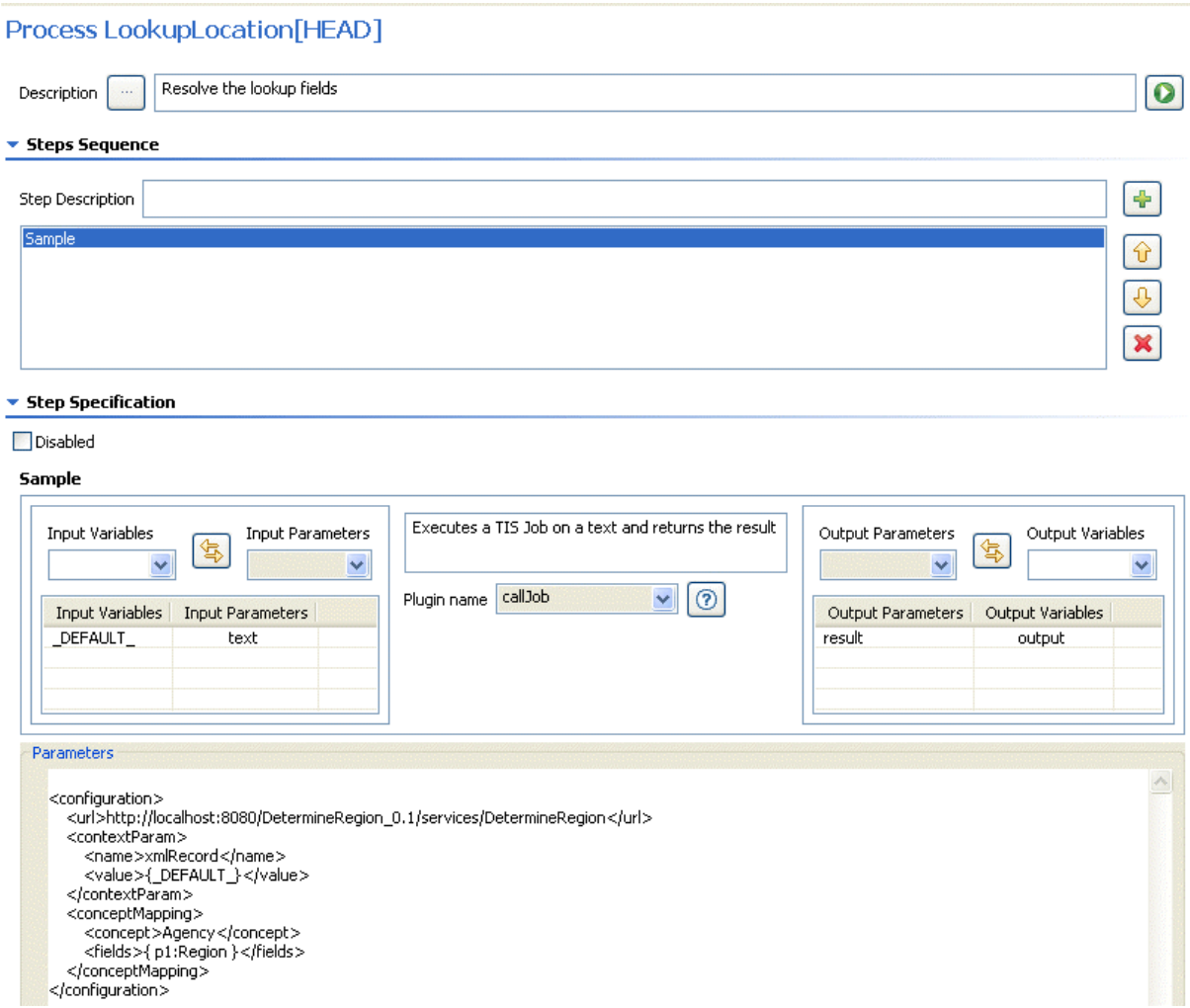
3.4.2.2. How to create a Process to enrich data on the fly

The Process used to run a View results can have a number of steps and plug-ins including CallJob. The Process will map MDM to the source system and then retrieve data from the source system to complete the record. For further information on Process types, plug-ins and the procedure to create a Process, see section *Processes*.

When a steward or a business user connects to *Talend MDM Web User Interface* and browse the records/views of the business entity/entities, the Process will lookup the defined elements (values) from the source system and display them in the web user interface without really saving them in the MDM Hub.

Prerequisite(s): You have already connected to the MDM server from *Talend Open Studio for MDM*.

Follow the steps outlined in section *How to create a Process from scratch* to create a normal Process and call it *LookupLocation*, for example.



1. Add a step name in the **Step Description** field and click the plus button to add it to the **Steps Sequence** list.
2. Click the step name to display the **Step Specifications** area. Here you can define the step parameters.
3. Set the Process input variable name to *_DEFAULT_*.

   The default input value is a text, its format is as the following `<cluster name>.<concept name>.<ids>`, you can use it to do the cross referencing in your customized Job.
4. From the **Plugin name** list, select *callJob*.

   This Process invokes a Job, *DetermineRegion*, to connect to the source system that holds the data you need, the agency region in this example, and then retrieves this data to complete the *Region* field in the data record in *Talend MDM Web User Interface*.
5. Set the Process output variable to *output*.
6. In the **Parameter** area, enter the parameters for the *calljob* plug-in as the following:

```
<configuration>
              <url>http://localhost:8180/DetermineRegion-0.1/services/
DetermineRegion</url>
   <contextParam>
        <name>xmlRecord</name>
        <value>{_DEFAULT_}</value>
   </contextParam>
   <conceptMapping>
        <concept>Agency</concept>
        <fields>{p1:Region}</fields>
   </conceptMapping>

</configuration>
```

7. Click the save icon on the toolbar or press **Ctrl + S** on your keyboard to save the Process.

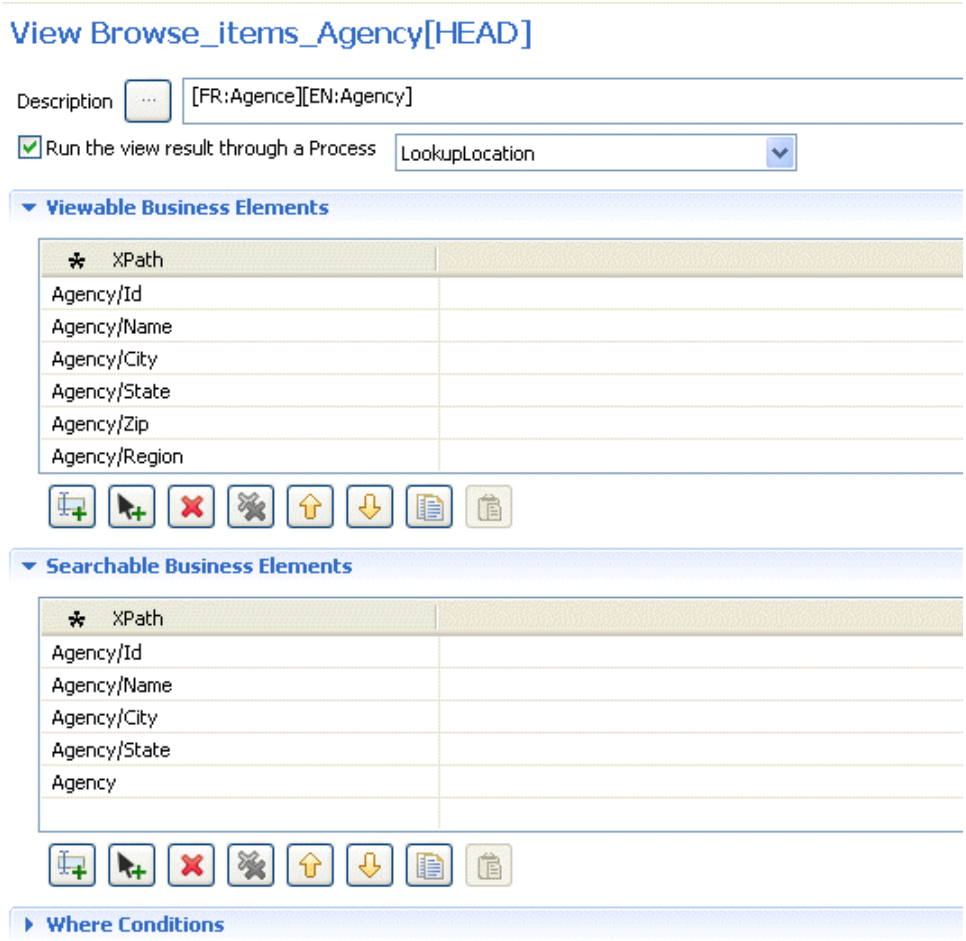
   The newly created Process is listed under the **Process** node in the **MDM Repository** tree view.

3.4.2.3. How to run the view results through a process

Prerequisite(s): You have already connected to the MDM server from *Talend Open Studio for MDM*. A Process corresponding to your needs has already been created.

To run the view results through a Process, do the following:

1. Follow the steps outlined in section *Creating a View* to create a View on the corresponding data model.



2. Select the **Run the view results through a process** to make the Process list available.

3. Click the arrow and select from the list the Process you want to attach to the current View, *LookupLocation* in this example.

4. Click the save icon on the toolbar or press **Ctrl + S** on your keyboard to save the View.

Now every time the business user or data steward browses the data records through this view from *Talend MDM Web User Interface*, the **Region** field will display the value retrieved by the Process from the source system without really saving it in the MDM Hub. For further information on how to browse data records, see *Talend MDM Web User Interface User Guide*.

3.4.3. Managing Views

An authorized user can carry out many management options on simple or composite Views from *Talend Open Studio for MDM*.

It is also possible to import and share MDM complete projects or only a data model or part of the data model from the community web page, **Talend Exchange**. For further information, see section *Projects/objects on Talend Exchange*.

The sections below explain all available management options for simple and composite Views.

3.4.3.1. How to export Views

From *Talend Open Studio for MDM* you can export one or multiple Views in order to exchange them between:

- two different MDM Servers or Repositories,
- two different Versions from the same/different MDM Servers or Repositories, for example.

The steps to export one or multiple Views are similar to those for any other data object in the **MDM Repository** tree view. For detailed information on how to export data containers, see section *How to export data models*.

3.4.3.2. How to import Views

From *Talend Open Studio for MDM* you can import Views into the current MDM Repository that have been created on other MDM Repositories or or on different Versions of the current MDM Repository.

The steps to import one or multiple Views are similar to those for any other data object in the **MDM Repository** tree view. For detailed information on how to import data containers, see section *How to import data models*.

3.4.3.3. How to edit a View

You can open a View you have already created to check its settings and/or edit the defined parameters.

Prerequisite(s): You have already connected to the MDM server in *Talend Open Studio for MDM*. At least one View exists.

To edit a View:

1. In the **MDM Repository** tree view, expand the **View** node.

2. Right-click the View you want to edit and select **Edit** from the contextual menu.

   An editor on the selected View opens in the workspace.

3. Modify the view parameters as needed and then click the save icon on the toolbar or press **Ctrl + S** on your keyboard to save your changes.

The selected View is modified accordingly.

If you try to update a View that has been modified by somebody else after you have retrieved it from the database, a warning message displays to warn you that saving your modifications will overwrite the other user's changes.

3.4.3.4. How to copy/paste a View

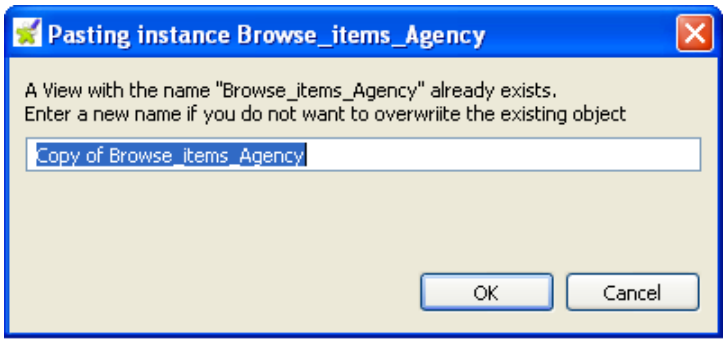
To avoid creating one or multiple Views from scratch, you can copy an existing one in the **MDM Repository** tree view and modify its parameters to have a new View.

Prerequisite(s): You have already connected to the MDM server in *Talend Open Studio for MDM*. At least one View exists.

To copy/paste a View, do the following:

1. In the **MDM Repository** tree view, expand the **Views** node.
2. Right-click the View you want to duplicate and select **Copy** from the contextual menu.
3. Right-click the **View** node and select **Paste** from the contextual menu.

   A dialog box displays prompting you to enter a name for the new View.



4. Enter a name for the new View and click **OK** to validate the changes and close the dialog box.

   The new View is listed under the **View** node in the **MDM Repository** tree view.

3.4.3.5. How to duplicate a View

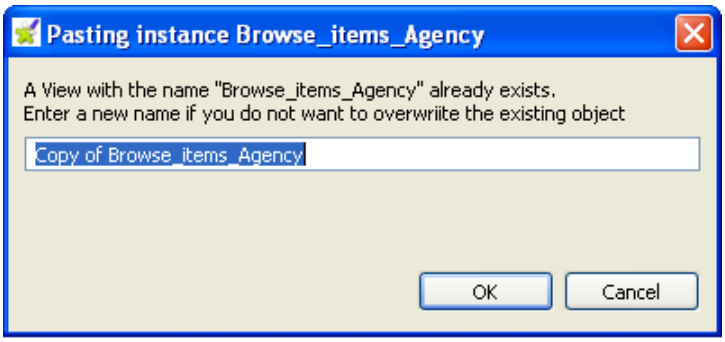
To avoid creating one or multiple Views from scratch, you can duplicate an existing one in the **MDM Repository** tree view and modify its parameters to have a new View.

Prerequisite(s): You have already connected to the MDM server in *Talend Open Studio for MDM*. At least one View exists.

To duplicate a View, do the following:

1. In the **MDM Repository**tree view, expand the **Views** node.
2. Right-click the View you want to duplicate and select **Duplicate** from the contextual menu.

   A dialog box displays prompting you to enter a name for the new View



3. Enter a name for the new View and click **OK** to validate the changes and close the dialog box.

   The new View is listed under the **View** node in the **MDM Repository** tree view.

You can also duplicate the data object if you drop it onto its parent node in the **MDM Repository** tree view.

3.4.3.6. How to delete a View

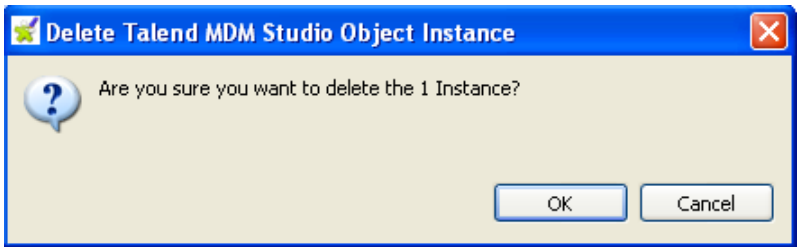
You can delete any of the Views you create through a simple right-click on the selected item.

Prerequisite(s): You have already connected to the MDM server in *Talend Open Studio for MDM*. At least one View exists.

To delete a View:

1. In the **MDM Repository** tree view, expand the **View** node.
2. Right-click the View you want to delete and select **Delete** from the contextual menu.

   A confirmation dialog box displays prompting you to confirm the deletion operation or to cancel it.



3. Click **OK** to close the dialog box.

   The selected View is deleted from the **MDM Repository** tree view.

3.5. Event management

Event management and real-time integration are an important part of Master Data Management. Talend MDM provides both functionalities with dedicated MDM objects as the following:

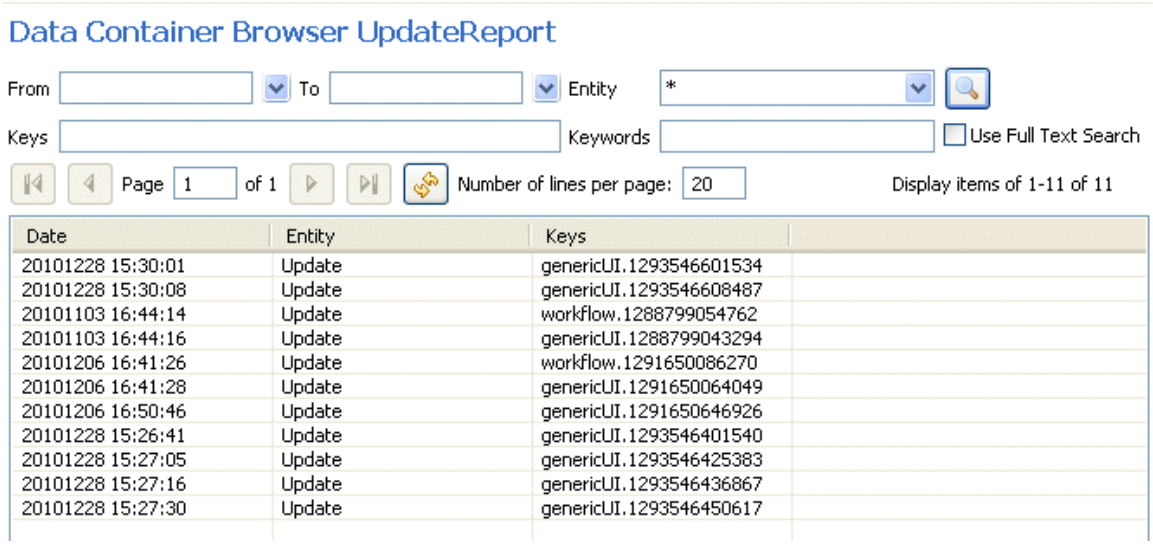
- The **Event Manager** in *Talend Open Studio for MDM* “listens” continuously for "events" that happen on the MDM Hub,
- An event happens when some data is modified, added or deleted in the MDM Hub,
- Talend MDM captures the event with an Update Report that gather the event audit log information,
- A Trigger is used to launch a Process when an event corresponds to this Trigger execution conditions,
- A Process, which is an action that usually executes a **Talend** Job, is launched by the Trigger.

Any event in the MDM Hub generates in *Talend Open Studio for MDM* an Update Report which gathers the event audit log information. This information includes the type of action (create, delete, update), the user who performed the action, source of action (Job, *Talend MDM Web User Interface*, etc.), modified attributes with old and new values and finally timestamp of action. The Update Reports are stored in Talend MDM as an XML document like any other master data but with a pre-defined, static data model.

To access these Update Reports, do the following:

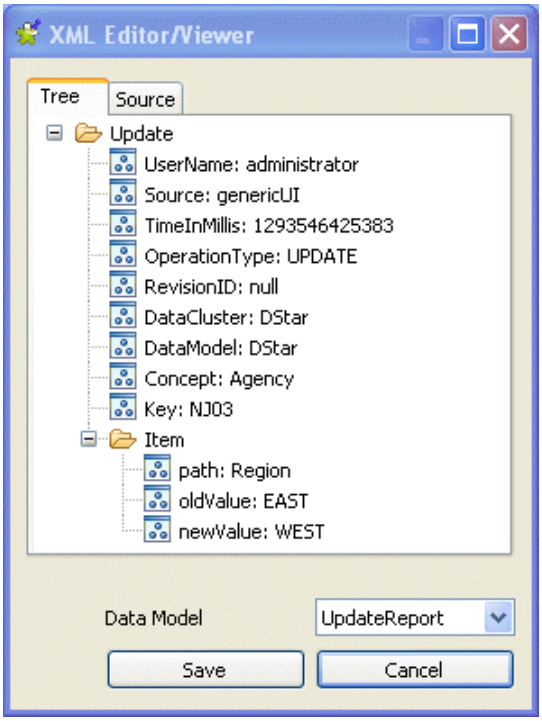
1. In the **MDM Repository** tree view, expand **Data Container - System** and then double-click **UpdateReport**.

   A dialog box displays.



2. Set parameters to filter the Update Reports you want to display, if required, and then click the  icon to display a list of the events done on master data in the MDM Hub.

3. Double-click any of these event to display a dialog box. Here you can see a detail audit log information about the selected event.



4. If required, click **Source** to display the log information in an XML document.

The following sections give detail explanation about the Processes and Triggers involved in the event management mechanism.

3.5.1. Processes

A Process defines multiple steps to achieve a data validation or human validation process, a transformation process, an enrichment process, a data integration process, etc. Each step of a Process can use a plug-in, which is a mechanism to add specific capabilities to the process to perform a single task. The list of available plug-ins is extensible by J2EE developers. The steps defined in each Process and the used plug-ins will differ according to the task you want to accomplish.

One of the plug-ins available in any Talend MDM platform is the *callJob* plug-in, which invokes a **Talend** Job exposed as a Web Service. For further information on Process types and associated plug-ins, see section *Process types* and section *Important plug-ins*.

Parameters to set when you define a Process include:

- Process name,
- Process description,
- sequence of steps: the list of all plug-ins included in the Process to be executed, one after the other,
- step specifications: a Process plug-in consumes data in input parameters and produces a result in the output parameters. You must define variables to hold the result of a step. Then you send the variable to the input of the next step, and so forth. Eventually, you define a “pipeline” where each step result is chained to the next step through a variable.

  A plug-in may have multiple input variables and parameters, as well as multiple output variables and parameters.

When you design a Process, you combine specific Process plug-ins in a sequence of steps, which are executed one after the other to perform specific tasks.

For each step:

1. Choose the appropriate plug-in,
2. Enter or select an input variable and an input parameter,
3. Select the output parameter and select or enter the output variable.

It is possible to disable one or more steps at any time if you select the **Disable** check box in the **Step Specification** area for the selected step.

For a step-by-step procedure on creating a Process, see section *How to create a Process from scratch*.

3.5.1.1. Process types

When creating a new Process from *Talend Open Studio for MDM*, you can select one of the proposed Process types: **SmartView**, **Before-Deleting**, **Before-Saving**, **Runnable Process**, **Normal** and **Standalone**.

The table below describes Process types:

Process type	Description	Associated plug-in
Normal	Any type other than the listed ones. This process is usually executed after an event occurs on master data in the MDM Hub.	Any in the list of available plug-ins.
Before-Saving	A Process that validates master data according to certain conditions before saving it in the MDM Hub. This Process can be linked with a Job designed in the **Integration** perspective to perform the validation operation automatically. This Process can alter the MDM record before it is committed in the database. For instance, a Job may be run that completes the record with calculations and/or enrichments.  The naming of this Process follows a specific pattern: `beforeSaving_<entity>`. Note also that, for improved readability, in the MDM Repository view, Processes are stored in subfolders by type and only the second part of the name is displayed.	Any in the list of available plug-ins. If this Process is linked with a **Talend** Job, it uses the *CallJob* plug-in. This plug-in executes a call to the Job created in the **Integration** perspective to evaluate the data to be saved and returns an error message if validation of this data is necessary. For further information, see section *How to set schema for a before saving/deleting Job*.
Before-Deleting	A Process that evaluates master data according to certain conditions before deleting it from the MDM Hub. This Process can be linked with a Job designed in the **Integration** perspective to perform the operation automatically.  The naming of this Process follows a specific pattern: `beforedeleting_<object name>`. Note also that, for improved readability, in the MDM Repository view, Processes are stored in subfolders by type and only the second part of the name is displayed.	Any in the list of available plug-ins. If this Process is linked with a **Talend** Job, it uses the *CallJob* plug-in. This plug-in executes a call to the Job created in the **Integration** perspective to evaluate data before deleting it and forbid the change by an error message. For further information, see section *How to set schema for a before saving/deleting Job*.
SmartView	An XSLT-based Process that is automatically detected by *Talend MDM Web User Interface*. It sets-up a more customized graphical presentation for a given data object (hiding some fields, displaying icons, etc.). The business user may choose to display or print the object with this read-only personalized view or to switch to the usual generated view where edits are possible.  The naming of this Process follows a specific pattern: `Smart_view_<entity>_[<ISO2>][<#name>]`, where the 2 character language ISO code and the name suffix are optional. <ISO2> allows you to define multilingual smart views, and the `<#name>` suffix allows you to have several alternates smart views of the same entity.	Usually XSLT which transforms an XML document using XSLT. But you can choose any number of steps using any plug-in, as long as the result at the end is HTML.

Process type	Description	Associated plug-in
	This Process name will automatically fallback to `Smart_view_<entity>` if the language is not found, and the language you use to define the smart view HTML parameters in the smart view editor will be picked as the default language. Note that, for improved readability, in the MDM Repository view, Processes are stored in subfolders by type and only the second part of the name is displayed. For further information, see section *How to create a smart view Process*.	
Runnable Process	A Process that is created in the Studio and automatically listed in the **Browse Records** page in *Talend MDM Web User Interface*. A business user can then select any of these Processes listed in the Web User Interface and click the **Launch Process** button to start the selected Process. For further information, see *Talend MDM Web User Interface User Guide*. This Process is always linked to a specific entity. You can design this Process to do any tasks you want, for example sending the entity by email. The naming of this Process follows a specific pattern: `Runnable_<entity>`. However, if you want to customize the Process name, you must add a hash sign before the word you want to add to the Process name; for example `Runnable_Agency#Send`. Note also that, for improved readability, in the MDM Repository view Processes are stored in subfolders by type and only the second part of the name is displayed.	Any in the list of available plug-ins. However, a typical associated plug-in is *CallJob* if the Process is linked with a **Talend** Job.
Standalone Process	Similar to the Runnable Process but is not linked to a specific entity. This Process is created in the Studio and automatically listed in the **[Welcome]** page in *Talend MDM Web User Interface*. You can design this Process to do any tasks you want, for example adding a new record/entity. The naming of this Process follows a specific pattern: `Runnable#<name>`; for example `Runnable#AddNewRecord`. Note also that, for improved readability, in the MDM Repository view, Processes are stored in subfolders by type and only the second part of the name is displayed. For further information, see section *How to create a standalone Process*	Any in the list of available plug-ins. However, a typical associated plug-in is *CallJob* if the Process is linked with a **Talend** Job.

3.5.1.2. Important plug-ins

Plug-ins are extra components that add specific capabilities to the Talend MDM. *Talend Open Studio for MDM* proposes a list of plug-ins to be combined with a given Process. These plug-ins include *callJob*, *workflowtrigger*, *groovy*, *xslt* and *crossreferencing*.

The table below explains some of the plug-ins listed in the Studio and details their parameters.

Plug-in	Action	Description
callJob	Executes a **Talend** Job on master data (to modify or propagate it, for example). For further information on the schemas used, see section *Schemas used in MDM processes to call Jobs*.	This plug-in executes a Web service call to the server where the Web service is deployed, usually the MDM server. Parameters: `url`: the webservice port URL. `Name`: name of the input variable.

Plug-in	Action	Description
		`Value`: value of the input variable. If you want to view the related Job, click the **Open Job** button to open it in the **Integration** perspective.
groovy	Calls the groovy script and uses it to process and transform data.	This plug-in implements all the capabilities of the groovy script to process and transform data when it receives an Update Report. It can read the XML document, transform data and write in the XML document as well.
crossreferencing	Transforms XML content using cross referencing tables.	This plugin takes an input XML and cross-references (e.g. replace the content of) a designated list of elements or attributes using the content extracted from items stored in the database. It offers the possibility to: -perform multiple cross-references on the source XML document in one pass, -either ignore a failed cross-reference or throw an error and stop execution of the Process, -specify a default value when the cross-referencing fails and errors are ignored. This plug-in provides enhanced functionality over the built-in cross-referencing provided by the XSLT plugin. Input variables: -*xml*: the source xml on which the cross-referencing will be performed. Output variables: -*text*: the result of the processing. For further information about this plug-in, see section *Example of the crossreferencing plug-in and its parameters*.

xslt	Transforms an XML document using XSLT.	This plug-in implements xslt transformations on an input XML document. It supports XSLT 2.0 and is enriched with cross-referencing capabilities: specific instructions that help to perform on the fly cross-referencing on any master data stored in the MDM Hub. When the output method of the XSLT is set to *xml* or to *xhtml*. Cross-referencing is carried out after the XSLT is processed on all elements with the following attributes:
		
		`<MyElement`
		`xrefCluster='CLUSTER'`
		`xrefIn='TEST1, ..., TESTN'`
		`xrefOut='XPATH_IN_ITEM'`
		`xrefIgnore='true|false'`
		`xrefDefault='DEFAULT_VALUE'`
		`>OLD_VALUE</MyElement>`
		
		Below is a definition of each of these attributes:
		
		`-xrefCluster`: the container (cluster) where the items used for cross-referencing are stored.
		
		`-xrefIn`: a series of XPaths tests to match this item content with a remote item.
		
		`-xrefOut`: the XPath in the remote item, starting with the entity (concept) name, of the content that will replace the content of this item.
		
		`-xrefIgnore`: optional, defaults to false. If set to true, the cross referencing will not fail if no item is found and the `xrefDefault` value will be inserted.
		
		`-xrefDefault`: if `xrefIgnore` is set to true and the cross-referencing fails, this value will be used instead.

Plug-in	Action	Description

		Input variables:
		
		-*xml*: the xml on which to apply the XSLT.
		
		-*parameters*: optional input parameters to the XSLT in the form of:
		
		`<Parameters>`
		`  <Parameter>`
		`  <Name>PARAMETER_NAME</Name>`
		`  <Value>PARAMETER_VALUE</Value>`
		`  </Parameter>`
		`</Parameters>`
		
		Output variables:
		
		- *text*: the result of the XSLT.
		
		For an example on this plug-in, see section *Example of the xslt plug-in*.

partialupdate	Performs partial updates on an item.	The *partialupdate* plugin updates elements of an existing item from the content of a supplied XML This plugin provides the ability to: -add sub elements or update existing elements, -add sub elements to an existing list of sub-elements starting from a specified position. Input variables: -*xml-instance*: the XML used to find and update an existing item. The updated item is searched based on the XML content, and the XML must follow certain specifications: First, the root element must have the same name as the name of the Entity of the item. Second, the XML must contain the value of all the item keys at the same XPath as those of the item unless `item_primary_key` is specified. Third, other than the keys, the XML can contain more elements than the one updated on the item but does not have to validate the item data model. `-item_primary_key`: optional if the key values are set on the `xml_instance`. The primary key must be supplied as an object of type `application/xtentis.itempk` as returned by the *project* item plugin. `-data_model`: (optional) the Data Model used to validate the item after update. Overwrites the corresponding value supplied in the parameters. `-clear_cache`: optional, defaults to false. If set to true, the Data Model is re-read and parsed from the database for each invocation of the plugin during the Process execution Output variables: `-item_primary_key`: the primary key of the updated item as an object of type `application/xtentis.itempk`.
project	Projects content to the Data Manager.	This plug-in saves an XML document in a data container after validation against a data model. If no item primary key is passed as a variable in the pipeline, the default data model and default data container to be used must be specified in the parameters. Input variables: -*xml-instance*: the XML of the item to project. `-item_primary_key`: (optional) the primary key of the projected XML. When not supplied, the primary key will be inferred from the xml by parsing the data model (slower).

Plug-in	Action	Description
		-data_model: (optional) the Data Model used to validate the item after update. Overwrites the corresponding value supplied in the parameters. -clear_cache: optional, defaults to false. If set to true, the data model is re-read and parsed from the database for each invocation of the plugin during the Process execution Output variables: -item_primary_key: the primary key of the projected item as an object of type application/xtentis.itempk. For further information about this plug-in, see section *Example of the project plug-in and its parameters*.

Example of the xslt plug-in

The following example parameters will loop over all the lines of the input XML and send them to the transformer as XML fragments:

```
<Country
    xrefCluster='MYCLUSTER'
    xrefIn='.=Country/Codes/ISO2, ../Customer/Name=[ACME]'
    xrefOut='Country/Name/FR'
><xsl:value-of select='State/CountryCode'/></Country>
```

The example above does the following:

- The XSLT generates a <*Country*> element in the target document,
- The content of *State/CountryCode* of the source document is inserted as the value of the element,
- The rest of the xslt transformations complete,
- The system queries the *Country* data in cluster *MYCLUSTER* where:*Codes/ISO2Code* is equal to *State/CountryCode* (the current value of the *Country* element), and */Customer/Name* in the target document is equal to hard coded value *ACME*,
- The matching *Country* document is returned and the value in Name/FR is extracted,
- The value in *Country* of the target document is replaced with the extracted value.

Example of the crossreferencing plug-in and its parameters

crossreferencing parameters:

The parameters are specified as an XML in the form of a list of *CrossRef* elements.

```
<parameters>
    <CrossRef>
           <xrefName>A_UNIQUE_NAME</xrefName>
           <xrefCluster>ITEMS_CLUSTER</xrefCluster>
           <xrefRootElement>ROOT_XPATH</xrefRootElement>
       <xrefIn>
              <mapping>
                     <xrefElement>XML_SOURCE_XPATH</xrefElement>
                     <xrefPath>ITEM_KEY_XPATH</xrefPath>
              </mapping>
       </xrefIn>
```

```
      <xrefOut>
            <mapping>
                  <xrefElement>XML_TARGET_XPATH</xrefElement>
                  <xrefPath>ITEM_VALUE_XPATH</xrefPath>
            </mapping>
      </xrefOut>
      <xrefIgnore>FALSE</xrefIgnore>
      <xrefDefault>EMPTY</xrefDefault>
   </CrossRef>
   <CrossRef>
   ...
   </CrossRef>
   ...
</parameters>
```

Below is a definition of each of these attributes:

- `xrefName`: an unique name for this CrossRef.
- `xrefCluster`: the name of the data-container where matching items will be searched.
- `xrefRootElement`: the XPath to an element in the source XML document to which all other XPaths will be relative. This is useful to process a list of sub-elements in the source documents: assuming a *PurchaseOrder* with multiple `LineItems`, specifying `//LineItem` will process the cross-referencing on all `LineItems`.
- `xrefIn/mapping/xrefElement`: an XPath relative to `xrefRootElement` in the source XML document. The content of this element or attribute will be used to search a matching `Item`.
- `xrefIn/mapping/xrefPath`: an XPath to an `Item` element or attribute, starting with the `Item` entity name. The first item where this XPath content matches the content of `xrefIn/mapping/xrefElement` will be selected.
- `xrefOut/mapping/xrefElement`: an XPath relative to `xrefRootElementin` the source XML document. The content of this element or attribute will be replaced with the content extracted from `xrefOut/mapping/xrefPath`.
- `xrefOut/mapping/xrefPath`: an XPath to an `Item` element or attribute, starting with the same `Item` entity name as the xrefIn/mapping/xrefPath. The content of this XPath will be used to replace the content of the `xrefOut/mapping/xrefElement` XPath.
- `xrefIgnore`: (optional; defaults to false). If set to true a failed cross-referencing will not stop the Process and the `xrefDefault` value will be mapped to the `xrefOut/mapping/xrefElement` element or attribute.
- `xrefDefault`: (optional; default to empty). The target value used when a cross-referencing fails and `xrefIgnore` is set to true. See `xrefIgnore`.

Example:

The following example parameters will cross-reference (e.g. replace the content of) the `UnitOfMeasure` attribute of the sub-elements `Quantity/Value` in all elements `PurchaseOrderLineItem` of the source XML by:

1. first, getting the value of the original UOM attribute of the `Quantity/Value` element,
2. second, using this value to find a `UnitOfMeasure` item in the data container **Cross Referencing** where the text content of the sub-element `cXMLUnit` equals the value,
3. and third, extracting the text content of the `XBITSunit` sub-element of the `UnitOfMeasure` item.

The Process will stop if the cross-referencing cannot be performed.

```
<parameters>
    <CrossRef>
            <xrefName>myCrossRefName</xrefName>
            <xrefCluster>CROSSREFERENCING</xrefCluster>
            <xrefRootElement>//PurchaseOrderLineItem</xrefRootElement>
            <xrefIn>
```

```
                    <mapping>
                            <xrefElement>Quantity/Value/@OriginalUOM</xrefElement>
                        <xrefPath>UnitOfMeasure/cXMLUnit</xrefPath>
                    </mapping>
            </xrefIn>
            <xrefOut>
                    <mapping>
                         <xrefElement>Quantity/Value/@UOM</xrefElement>
                           <xrefPath>UnitOfMeasure/XBITSUnit</xrefPath>
                    </mapping>
            </xrefOut>
    </CrossRef>
</parameters>
```

Example of the project plug-in and its parameters

project parameters:

The parameters are specified as an XML.

```
<parameters>
    <defaultDataCluster>DATA_CLUSTER_NAME</defaultDataCluster>
    <defaultDataModel>DATA_MODEL_NAME</defaultDataModel>
    <overwrite>true|false</overwrite>
</parameters>
```

Below is a definition of each of these attributes:

- `defaultDataCluster`: optional if input `item_primary_key` is supplied. It is the data container in which the item will be stored.
- `defaultDataModel`: optional if input data model is supplied. It is the data model used to validate the item.
- `overwrite`: optional defaults to true. If set to false, an existing item will not be overwritten by this xml.

Example:

The following example parameters will project a supplied XML to the *myDataCluster* data container validating against the *myDataModel* data model. An existing item will be overwritten:

```
<parameters>
   <defaultDataCluster>myDataCluster</defaultDataCluster>
   <defaultDataModel>myDataModel</defaultDataModel>
   <overwrite>true</overwrite>
</parameters>
```

3.5.1.3. Schemas used in MDM processes to call Jobs

When a Job is called from an MDM process, it receives an XML document based on a specific schema. In return, the Job sends back a document which must also conform to a particular schema.

How to set the schema for a Job called through a Trigger

This is the typical case when a Process is called by a Trigger. The Process uses a *callJob* plug-in to invoke a **Talend** Job created in the **Integration** perspective of *Talend Open Studio for MDM*.

Input Schema A document is passed on to the Job. The schema is:

```
<item>
         ... record ...
</item>
```

Assuming a *Customer* record, the complete result is:

```
<item>
        <Customer>
                <Firstname>Janet</Firstname>
                <Lastname>Richards</Lastname>
        </Customer>
</item>
```

Output schema If the Job returns nothing, MDM will generate a document with the Job return status in *callJob* output variable:

```
<results>
        <item>
                <attr>0=ok or 1=failed</attr>
        </item>
</results>
```

If the Job returns a table though a **tBufferOutput** component, MDM will define the following document in the *callJob* output variable:

<results>

```
     <item>
             <attr>col1</attr>
             <attr>col2</attr>
             etc.
     </item>
```

</results>

This result may be mapped back into an Entity by adding the following fragment in *callJob* configuration:

```
<configuration>
(...)
   <conceptMapping>
         <concept>Customer</concept>
         <fields>
           {
           p0:Firstname,
           p1:Lastname,
           }
        </fields>
   </conceptMapping>
</configuration>
```

Then *callJobs* output variable will receive:

```
<results>
        <Customer>
                <Firstname>col1</Firstname>
                <Lastname>col2</Lastname>
        </Customer>
</results>
```

How to set schema for a before saving/deleting Job

The **Before-Saving**/**Before-Deleting** processes are called directly by naming convention. They do not go through the usual Trigger > Process mechanism. Job called through a **Before-Saving** or **Before-Deleting** Process receive

a different document than when they are called through a Trigger. In addition, they are expected to return a status report or an error message which the Web Interface can use to proceed with / cancel the action.

The process must always return a variable called `output_report`.

Input

The input document comprises the update report as well as the record which is being saved or deleted:

```
<exchange>
      <report>
      ... update report ...
      </report>
      <item>
      ... record ...
      </item>
</exchange>
```

You can always find the exact schema description of an update report in the **MDM Repository** tree view in **Data Model** > **System** > **UpdateReport**.

Within the Job, you may put conditions similar to triggers. For instance, you may use *exchange/report/Update/OperationType* to implement different conditions on `CREATE` and `UPDATE`.

Output

The Job is required to return a document that conforms to:

```
<report><message type="info">message</message></report>
```

Or to:

```
<report><message type="error">message</message></report>
```

When you want to create a **Before-Saving** Process that both checks validation rules and completes the record on the fly, you must define a two-step Process, with one step returning `output_item` and the other step returning `output_report`.

The working principles for the **Before-Saving** and **Before-Deleting** Processes can be summarized as described in the below three cases.

Upon completion of the **Before-Saving** or **Before-Deleting** processes, the MDM server looks for a variable called `output_report` in the Process pipeline.

First case:

- If `<report><message type="info">message</message></report>`: the validation process of the data record has been carried out successfully and a message will display. The data record will be successfully saved with the **Before-Saving** Process, or successfully deleted with the **Before-Deleting** Process.
- If `<report><message type="error">message</message></report>`: the validation process of the data record fails and a message displays. The data record will not be saved with the **Before-Saving** Process, and it will not be deleted with the **Before-Deleting** Process.

Second case:

The MDM server has not found the `output_report` variable. The validation process of the data record has failed and an error message will display to confirm this. The data record will not be saved with the **Before-Saving** Process, and it will not be deleted with the **Before-Deleting** Process.

Third case:

The Process throws an exception (typically one of the steps in the Process leads to a technical error: wrong configuration, XSLT syntax error, Job not found or could not be called, etc.). A technical error message will

display and the data record will not be saved with the **Before-Saving** Process, and it will not be deleted with the **Before-Deleting** Process.

3.5.1.4. How to set up a callJob Process chain using the Create Process wizard

The **[Create Process]** wizard takes you through the generation of the complete *callJob* Process chain for each of the following types of Process: *Before*, *Runnable* and *Other*. For a description of each type of Process, see section *Process types*.

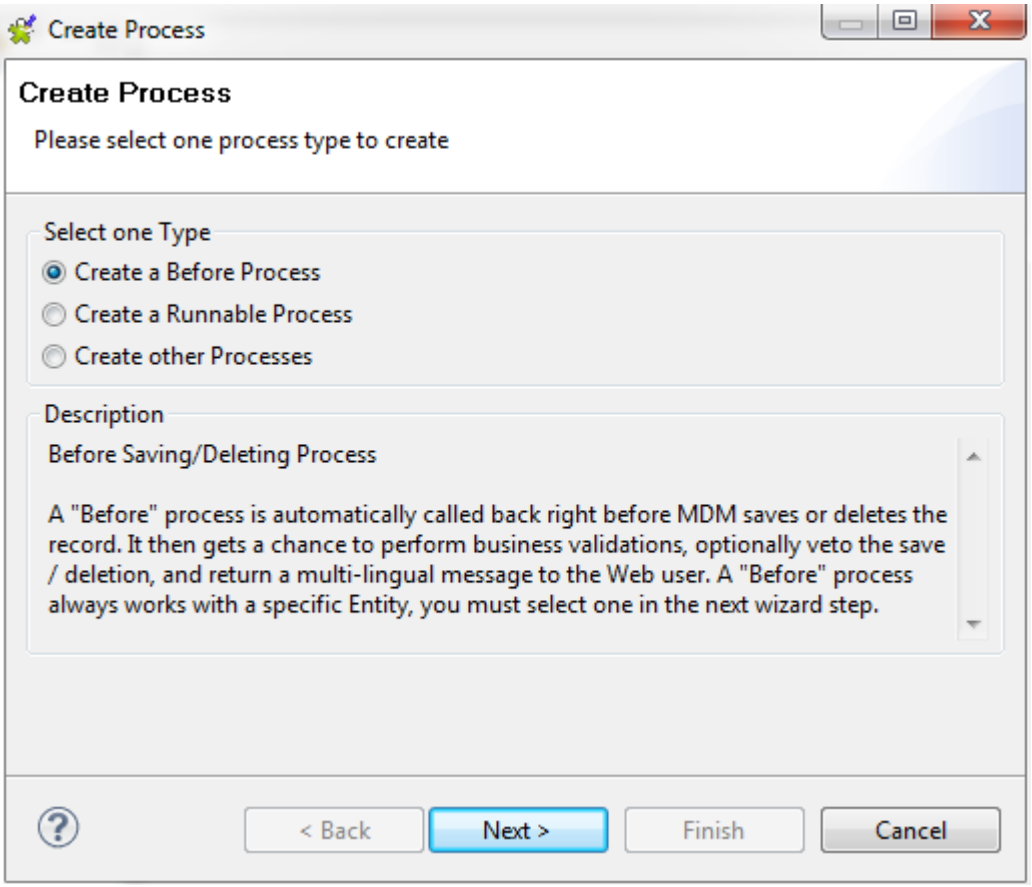
The steps to follow vary depending on the type of Process being generated.

Setting up a callJob Process chain for a Before Process

To set up the *callJob* Process chain for a *Before-Saving Process* or a *Before-Deleting Process* using the **[Create Process]** wizard, do the following:

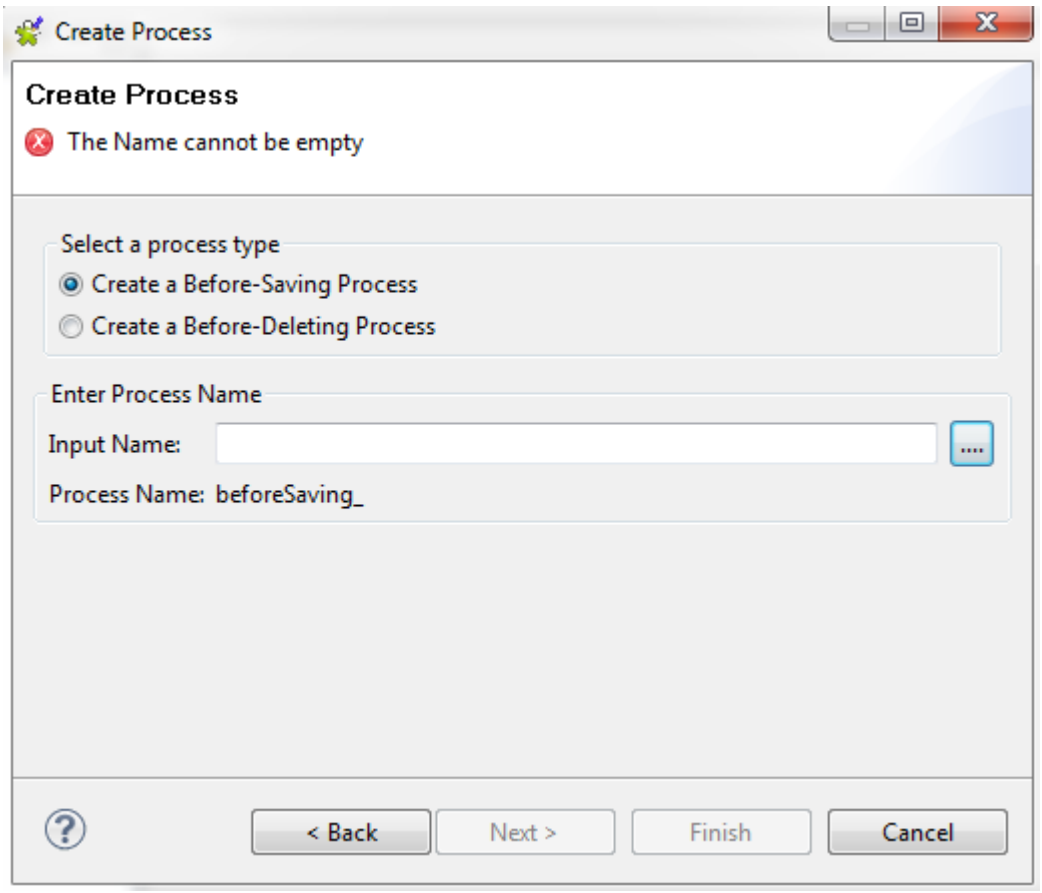
1. In the **MDM Repository** tree view, expand the **Event Management** node, right-click the **Process** node, and then click **New**.

   The **[Create Process]** wizard opens.



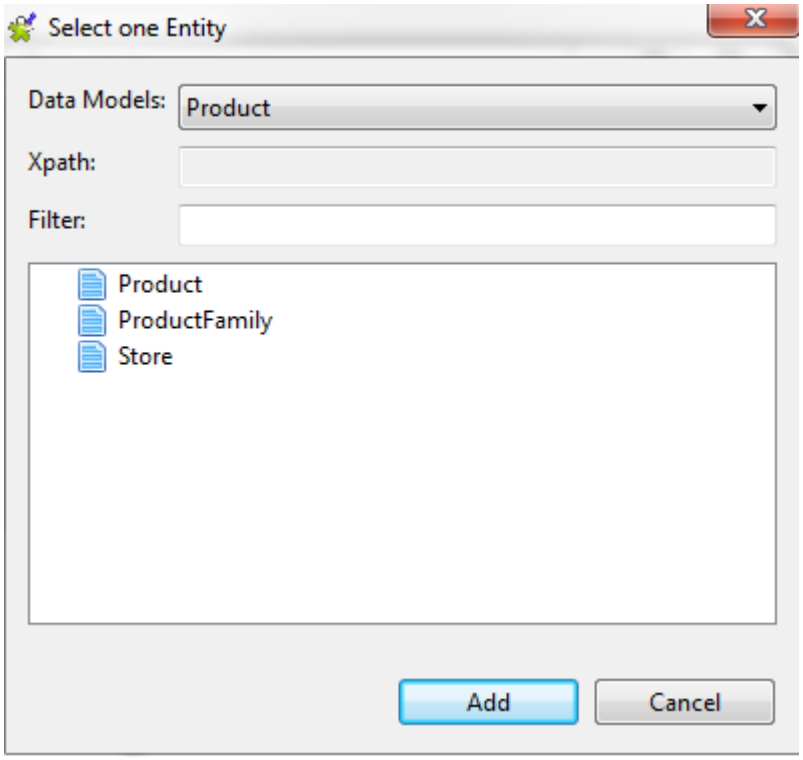
2. Select which type of Process you want to create, and then click **Next**.

   The next screen of the **[Create Process]** wizard opens.



3. Select whether you want to create a **Before-Saving Process** or a **Before-Deleting Process,** and then click the **[...]** button next to the **Input Name** field.

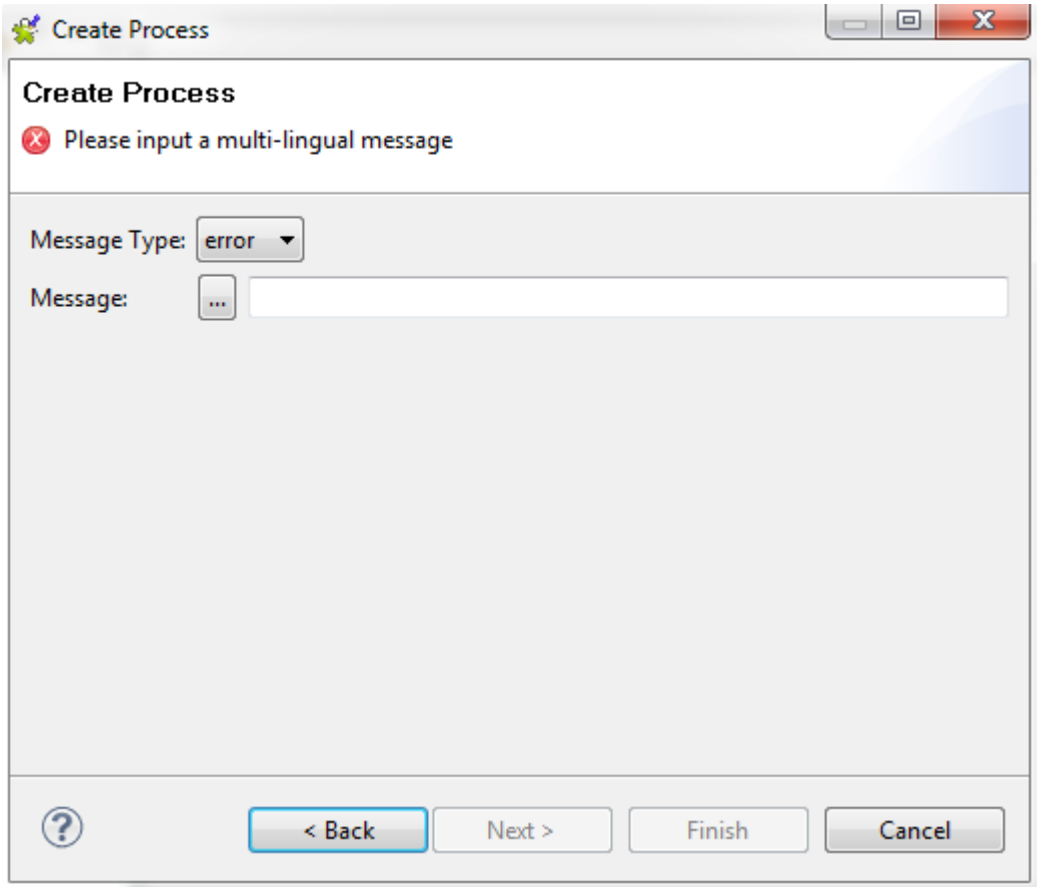
   The **[Select one Entity]** dialog box opens.



4. Select the specific Entity for which you want to generate the Process, and then click **Add** to return to the **[Create Process]** wizard.

5. Click **Next**.

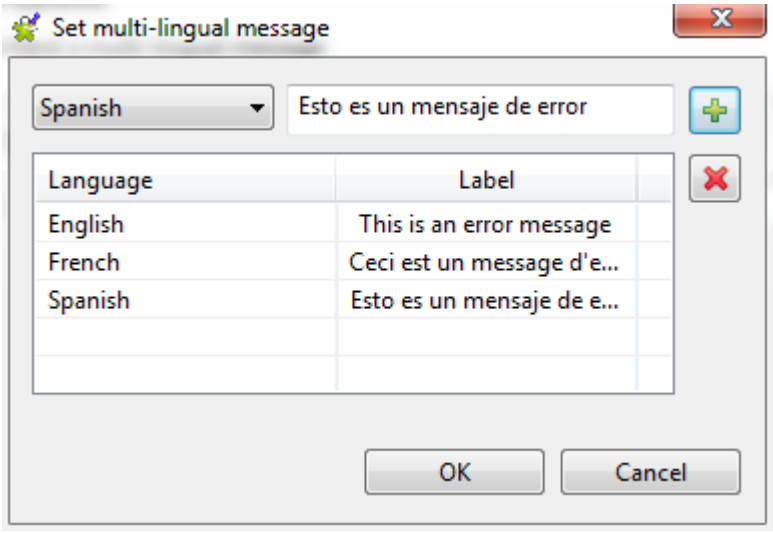
   A dialog box opens in which you can input a multi-lingual message to accompany your Process.



6. Define the error or informational message you want to display, as follows:

   1. Select the Message Type, **error** or **info**.

   2. Click the **[...]** button next to the **Message** field to open a dialog box in which you write the message and, if appropriate, a localized version in one or more additional languages.



3. Click **OK** to close the **Set multi-lingual message** dialog box and return to the **[Create Process]** wizard.

7. Click **Next**.

8. Select or deselect the **Generate the template job** checkbox to specify whether you want to generate a template job for the process, and then click **Finish**.

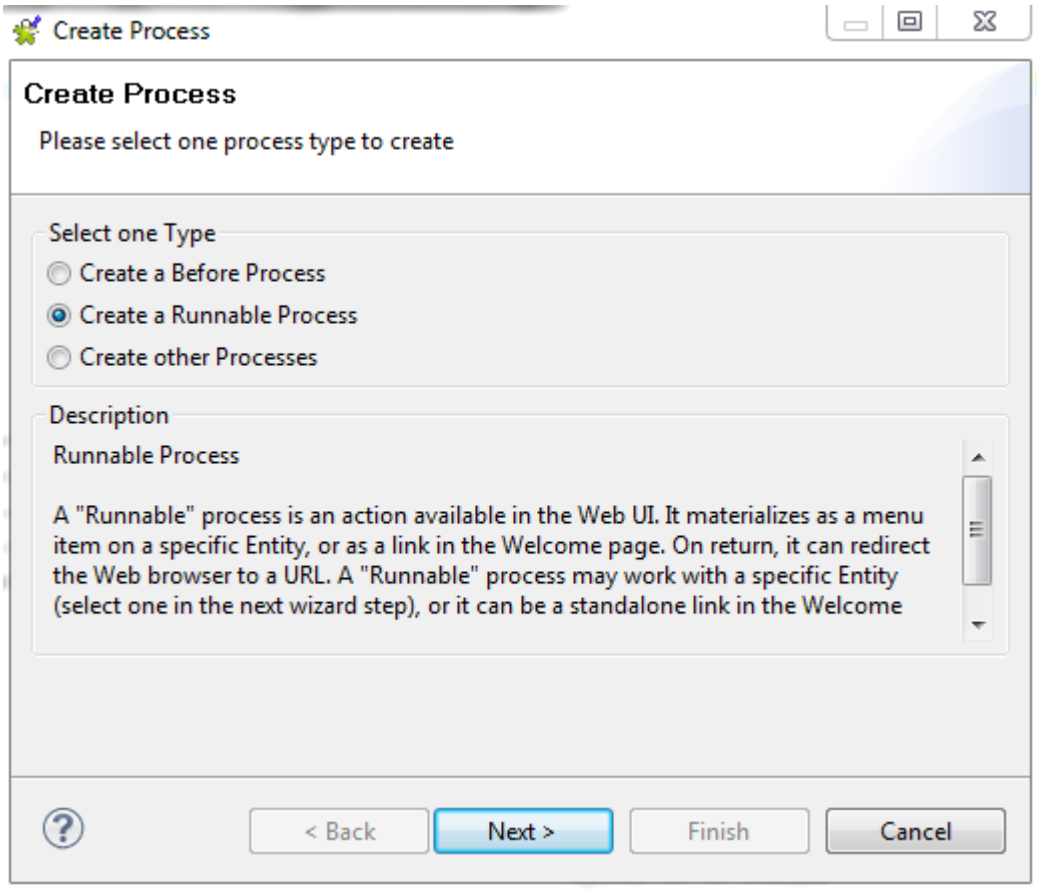
   The Process and, if appropriate, the template Job open.

Setting up a callJob Process chain for a Runnable Process

To set up the *callJob* Process chain for a *Runnable Process* using the **[Create Process]** wizard, do the following:

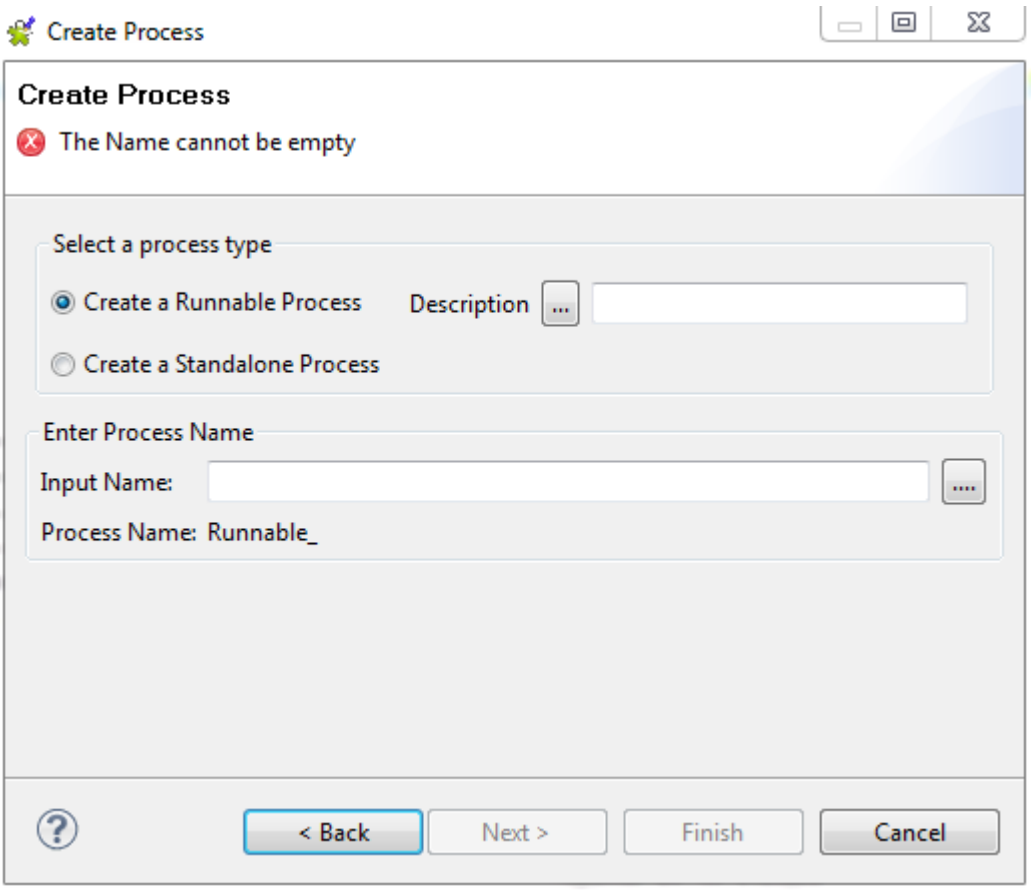
1. In the **MDM Repository** tree view, expand the **Event Management** node, right-click the **Process** node, and then click **New**.

   The **[Create Process]** wizard opens.



2. Select which type of Process you want to create, and then click **Next**.

   The next screen of the **[Create Process]** wizard opens.

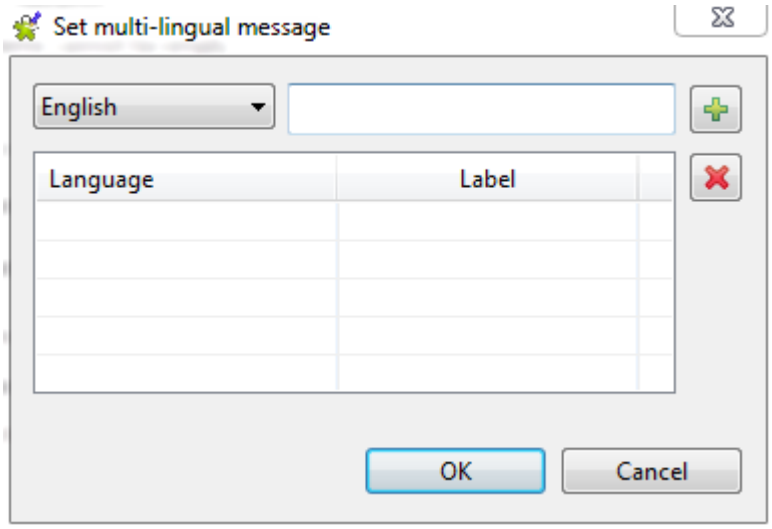


3. Select whether you want to create a *Runnable Process* or a *Standalone Process*.

   A *Runnable Process* is linked to a specific Entity (which you define in the next step of the wizard). A *Standalone Process* appears as a standalone link in the Welcome page of the *Talend MDM Web User Interface*.

4. Click the **[...]** button next to the **Description** field.

   The **[Set multi-lingual message]** dialog box opens.

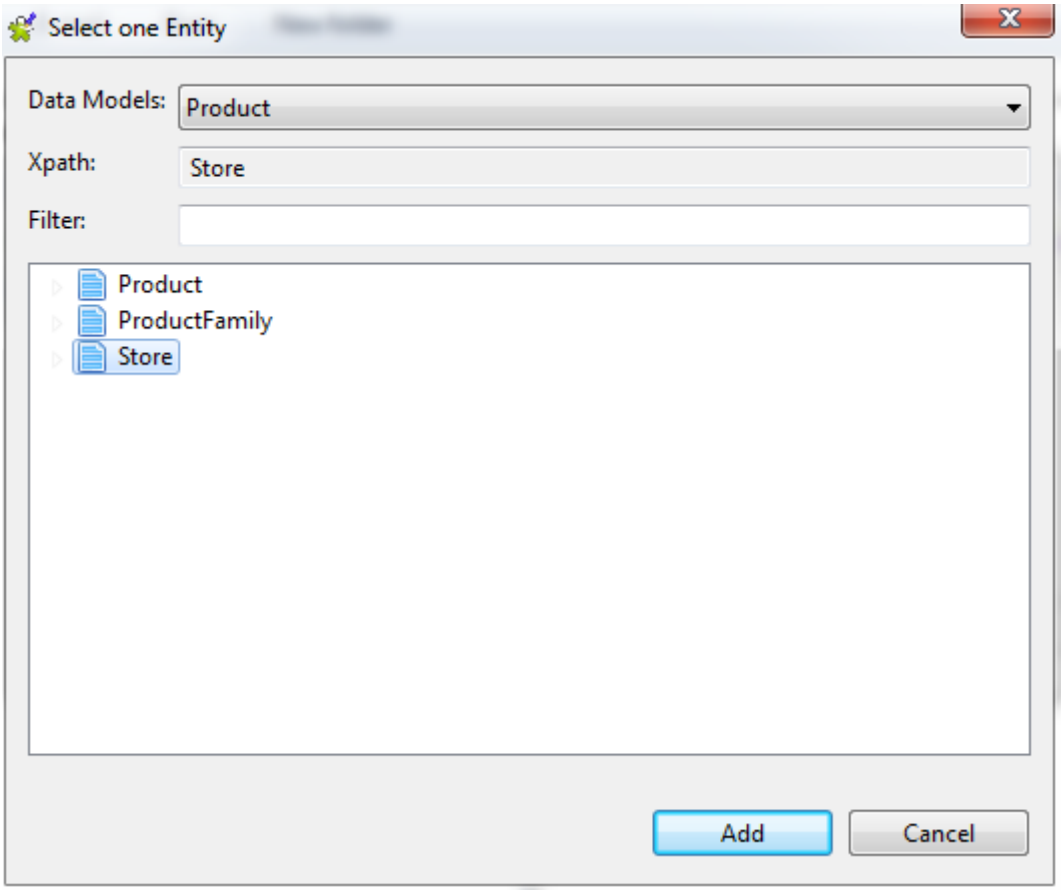


5. Enter the text for the English-language label to accompany your Process, and then click the **[+]** button.

You can also define localized versions of the label for other languages if required.

6. Click **OK** to return to the **[Create Process]** wizard.
7. Click the **[...]** button next to the **Input Name** field.

   The **[Select one Entity]** dialog box opens.

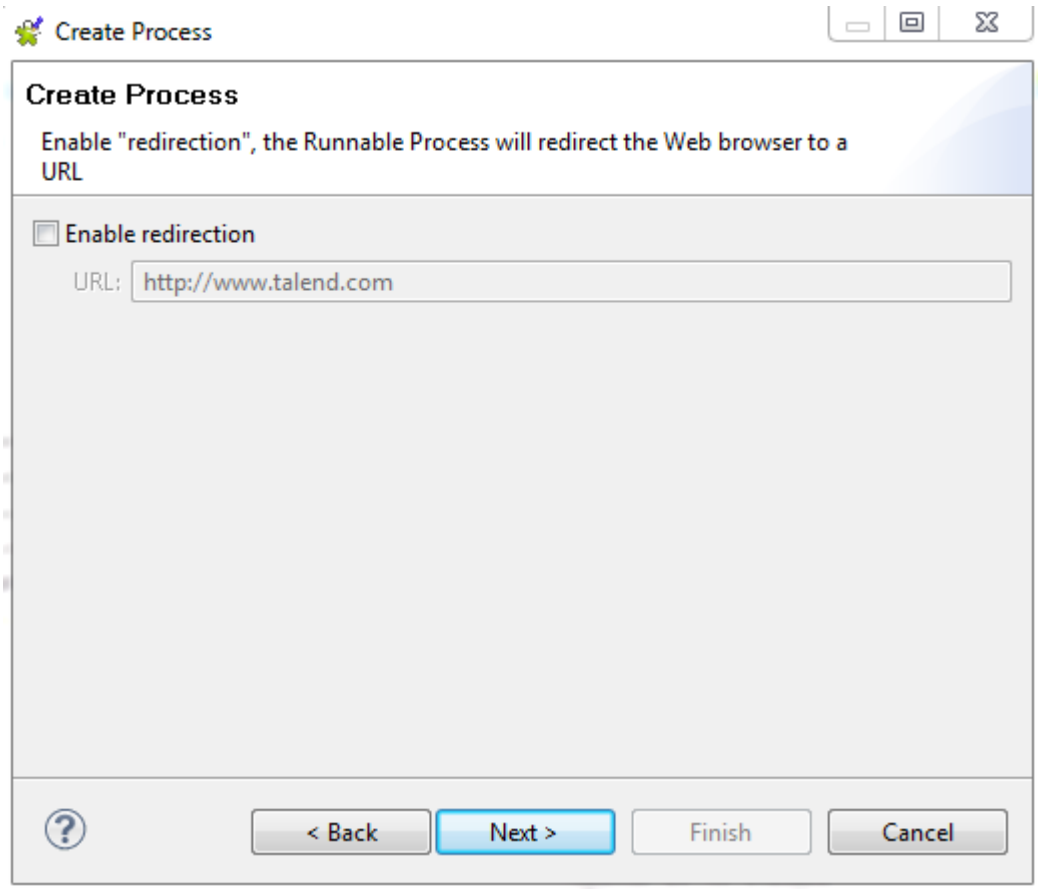


8. Select the Entity for which you want to generate the Process, and then click **Add** to return to the **[Create Process]** wizard.

   If you are creating a *Standalone Process*, skip the Entity selection step and manually enter a name for your Process in the **Input Name** field instead.

9. Click **Next**.

   The next screen of the **[Create Process]** wizard opens.



10. Select the **Enable redirection** checkbox if you want the Process to redirect the Web browser to a URL, and specify the URL in the **URL** field.

11. Click **Next**.

12. Select or deselect the **Generate the template job** checkbox to specify whether you want to generate a template job for the process, and then click **Finish**.

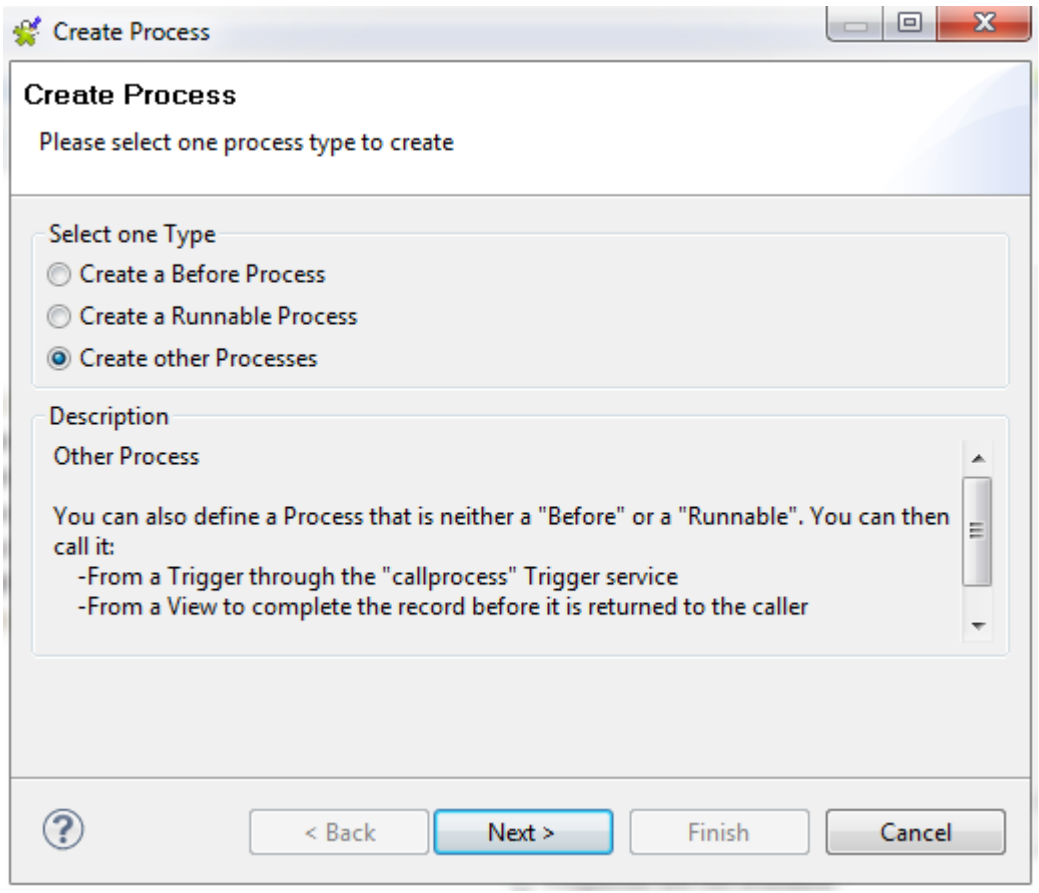
    The Process and, if appropriate, the template Job open.

Setting up a callJob Process chain for an Other Process

To set up the *callJob* Process chain for an *Other Process* using the **[Create Process]** wizard, do the following:

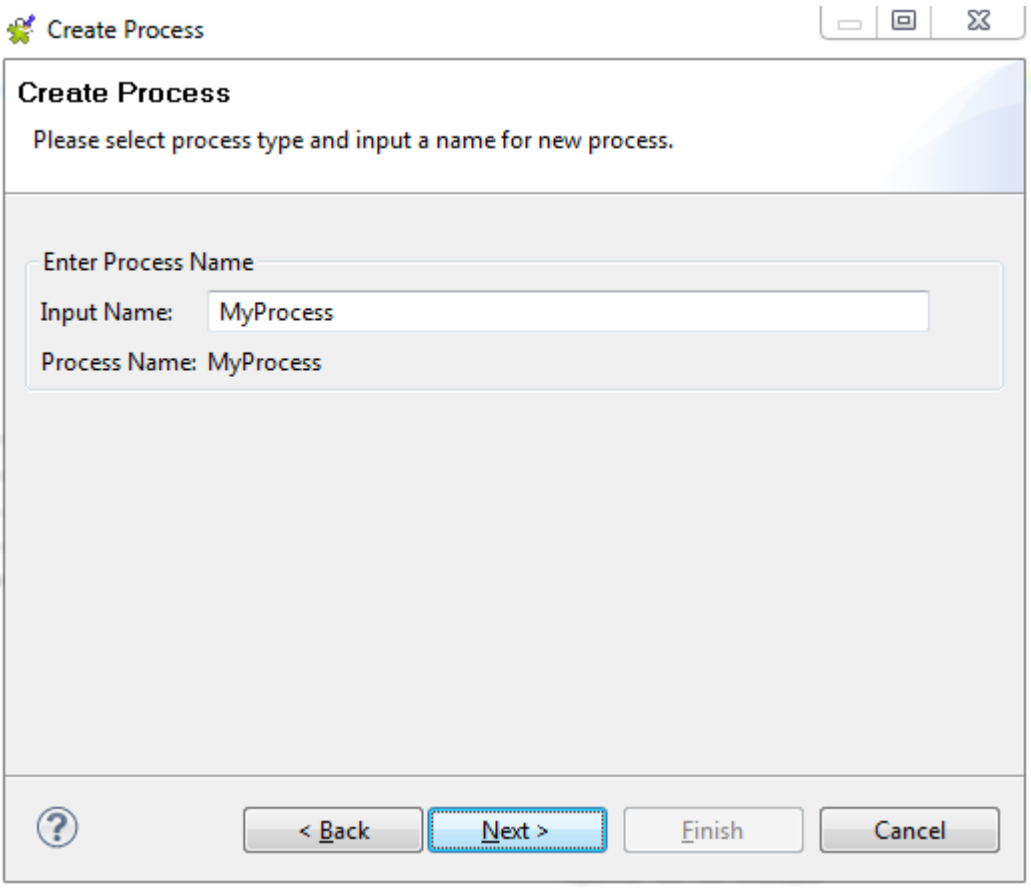
1. In the **MDM Repository** tree view, expand the **Event Management** node, right-click the **Process** node, and then click **New**.

   The **[Create Process]** wizard opens.



2. Select which type of Process you want to create, and then click **Next**.

   The next screen of the **[Create Process]** wizard opens.



3. Enter a name for your Process in the **Input Name** field, and then click **Next**.

4. Select or deselect the **Generate the template job** checkbox to specify whether you want to generate a template job for the process, and then click **Finish**.

   The Process and, if appropriate, the template Job open.

3.5.1.5. How to create a Process from scratch

When you design a Process, you combine specific Process plug-ins into a sequence of steps. These steps are then executed one after the other to perform specific tasks.

Whenever a data record is created/updated/deleted, the MDM Server generates a document and lists it under the **UpdateReport** node in the **System** data container in the **MDM Repository** tree view. This document describes the event in answering a who, what and when questions and in giving the record primary key and the values before and after in case of an update. This **UpdateReport** document does contain everything about the event that just happened, however it does not contain the complete XML record that was created/updated/deleted. This document is then sent to the Event Manager. Whenever the Event Manager receives a document, it tries to evaluate “route” every Trigger conditions against this document. For further information about Triggers, see section *Triggers*.

So the sequence of events that occur whenever a Create/Update/Delete (CRUD) is performed in Talend MDM is as the following:

- the Event Manager evaluates every defined Trigger to see if one or more Triggers have valid conditions,
- the services defined in the Trigger are performed,

- in case a *callJob* service has been defined in the Trigger, the Trigger uses the *callJob* plug-in to run a **Talend** Job

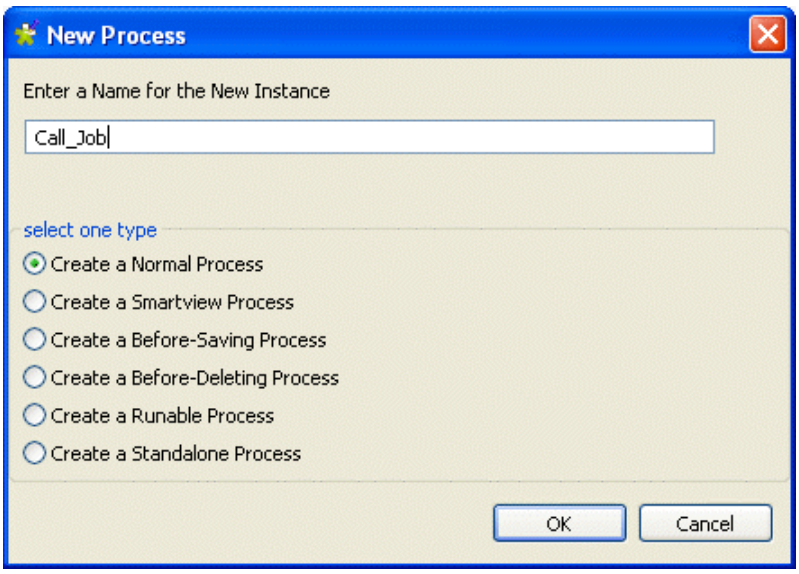
In the example below, a data model called *Product* has been created in *Talend Open Studio for MDM*. This data model has two business entities: *Product* and *ProductFamily*. Several attributes have been created in the *Product* entity including *Price* and *Family*. You want to create a Process to automatically trigger a validation Job whenever a price of an item that belongs to a specific family has been changed through *Talend MDM Web User Interface*.

Prerequisite(s): You have already connected to the MDM server from *Talend Open Studio for MDM*. You have the appropriate user authorization to create Processes.

To create a Process to automatically trigger a **Talend** Job:

1. In the **MDM Repository** tree view, expand **Event Management** and then right-click **Process** and select **New** from the contextual menu.

   The **[New Process]** dialog box displays.



2. Select the option corresponding to the Process type you want to create and then enter a name for the new Process.

   In this example, you want to create a normal Process. For more information on Process types, see section *Process types*.

   If there are naming conventions for the Process you want to create, they will display in the name field when you select the Process type. You just need to enter the name of the entity on which to run the Process.

   No spaces are allowed in the process name. The OK button in the dialog box will be unavailable as long as there is a space in the instance name.

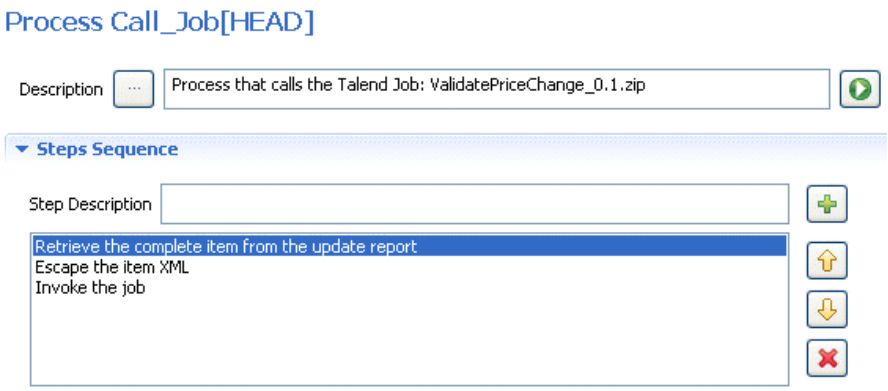
3. Click **OK** to close the dialog box.

   An empty editor for the newly created Process opens in the workspace.

4. If required, click the three-dot button next to the **Description** field to open a dialog box where you can set multilingual descriptions of your Process.

5. In the **Step Description** field, enter a name for the first step you want to define in the created Process and then click the icon to add the step name in the rectangle below the field.

6. Repeat to add the two other steps included in this Process.



The Process for this example must include one basic step, *Invoke the job* through the *callJob* plug-in, and two obligatory additional steps as the following:

First step: to retrieve the complete XML record through the *XSLT* plug-in.

This step is obligatory in this example since you need the whole record in order to check to what family belongs the item which price has been changed and you cannot find such information in the Update Report.

Second step: to decode XML in order to remove the escape function you used in the XSLT document and thus send a real XML document to the Job.

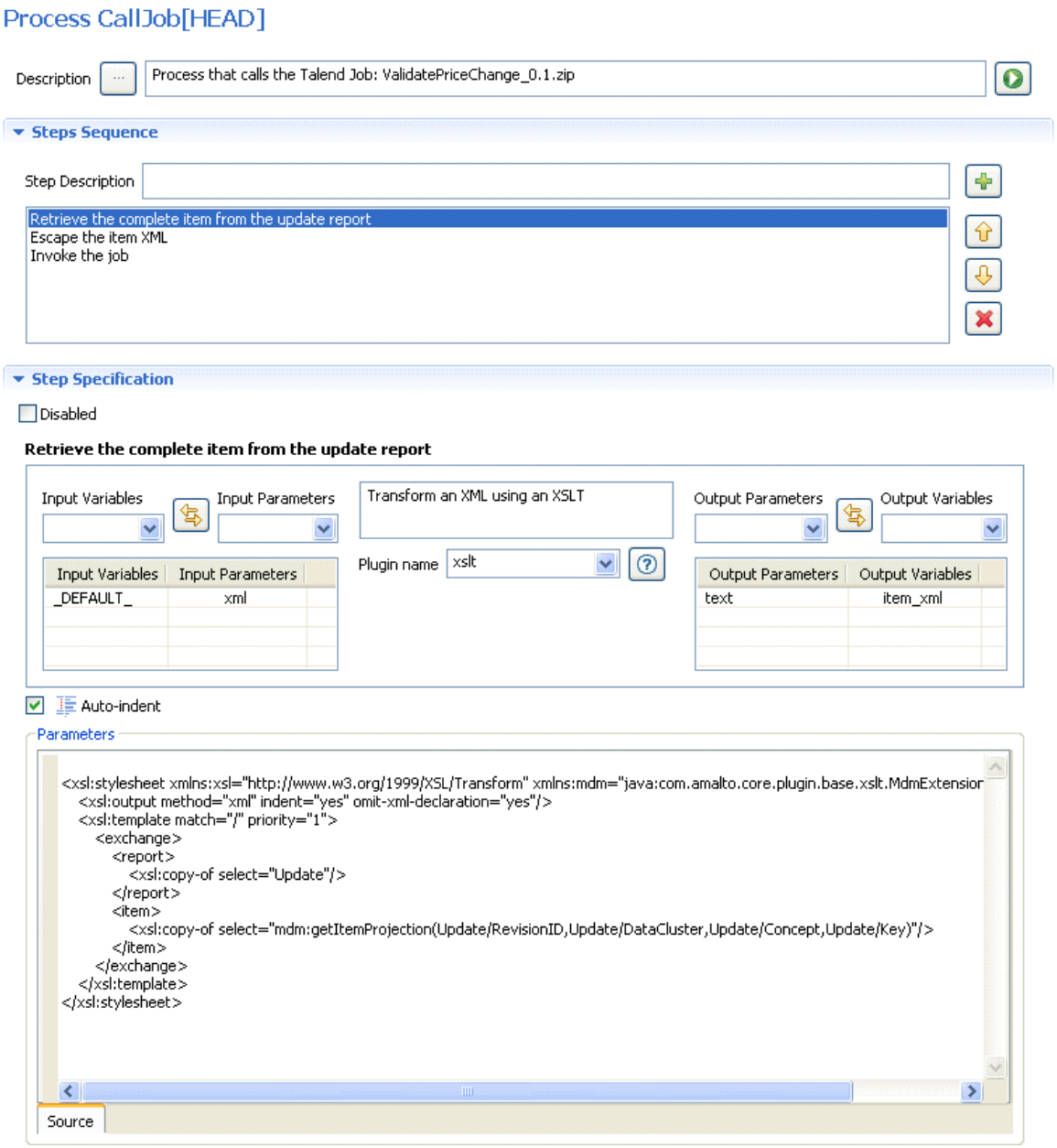
Final step: to send the XML document to a **Talend** Job.

Usually, you can create a Process with only the *callJob* plug-in if the complete XML record is not needed, depending on the type of the task you want to accomplish. For example, if you want to validate the price change of an item in general without mapping it to a specific family, you can create this Process with only one step: *Invoke the job*.

The below procedures describe in detail how to define each of the listed steps.

How to retrieve the complete XML record

1. Click the first step to display the **Step Specifications** area where you can define the step parameters.



You can disable the selected step in the open Process editor at any time if you select the **Disable** check box in the **Step Specification** area.

2. From the **Plugin name** list, select *xslt*.

   A description for the selected plug-in displays in the box above the **Plugin name** field.

   The button displays a dialog box that describes the plug-in and details its parameters.

3. As the *xslt* plug-in is our first step, leave the input variable empty, select the *xml* input, and click the button to add them to the table.

   The MDM server starts the pipeline with a default variable called *_DEFAULT_* .

   Here you want to map variables to the input parameter of the plug-in, and conversely map output parameter of a plug-in to another variable.

4. Select the output parameter *text* and define a new output variable, *record_xml* in this example and then click the  button to add them to the table.

   For each step, you can select one of the by-default variables or one of the variables defined for the preceding plug-in the Process you define.

5. In the **Parameters** area, complete the XSLT definition as the following:

```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns:mdm="java:com.amalto.core.plugin.base.xslt.MdmExtension" version="1.0">
<xsl:output method="xml" indent="yes" omit-xml-declaration="yes" />
<xsl:template match="/" priority="1">
<exchange>
<report>
<xsl:copy-of select="Update"/>
</report>
<item>
<xsl:copy-of select='mdm:getItemProjection(Update/RevisionID,Update/
DataCluster,Update/Concept,Update/Key)'>
</item>
<exchange>
</xsl:template>
</xsl:stylesheet>
```

   The text you enter in the text editor is indented by default upon saving. This may cause problems in certain cases such as the HTML formatting when creating a smart view process. If required, clear the ***Auto-indent*** *check box to disable the text indentation upon saving the process.*

In this first step, the Process pipeline starts with a *_DEFAULT_* variable, which contains the Update Report.The content of *_DEFAULT_* is sent to this first XSLT step through its *text* input parameter.The XSLT step retrieves the complete XML record through the *getItemProjection* function in order to escape every XML sign to avoid any character encoding conflicts in XSLT. The result of the step is sent from the output parameter to the *item_xml* variable. So now *item_xml* is ready to be sent to the next step of the Process pipeline.

How to decode XML

In this step, you want to use a decoding mechanism to remove the escape function from the XML document before sending it to the Job.

1. Click the second step to display the **Step Specifications** area where you can define the step parameters.

   You can disable the selected step in the open Process editor at any time if you select the **Disable** check box in the **Step Specification** area.

2. From the **Plugin name** list, select *codec*.

   A description for the selected plug-in displays in the box above the **Plugin name** field.

   The  button displays a dialog box that describes the plug-in and details its parameters.

3. Map the *item_xml* variable defined in the previous step to this step *law_text* input parameter and click the  button to add them to the table.

4. Map this step output parameter to a new variable called *decode_xml*, for example then click the  button to add them to the table.

   For each step, you can select one of the by-default variables or one of the variables defined for the preceding plug-in the Process you define.

5. In the **Parameters** area, complete the definition as the following:

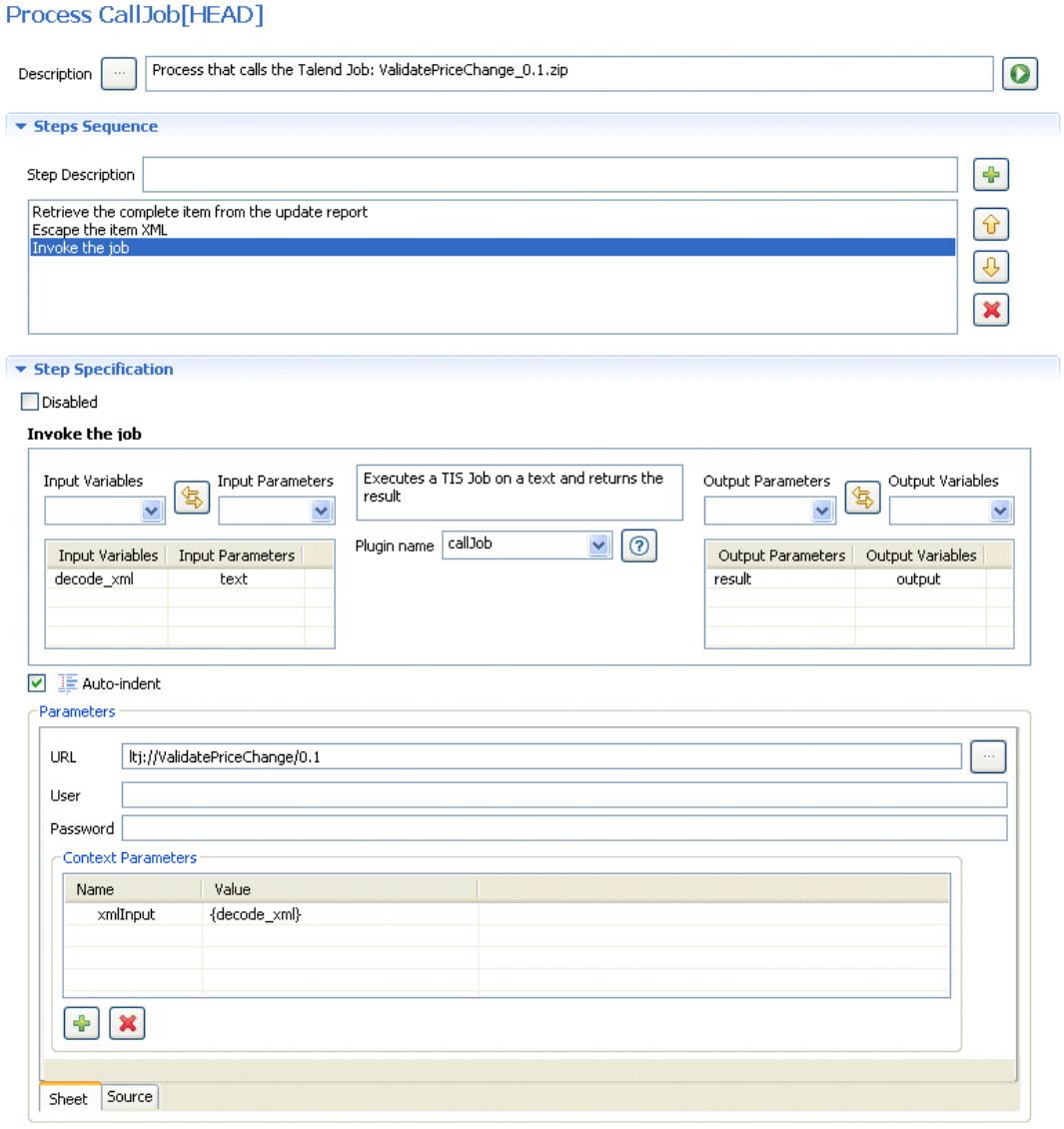
```
<parameters>
<method>DECODE</method>
<algorithm>XMLESCAPE</algorithm>
</parameters>
```

The text you enter in the text editor is indented by default upon saving. This may cause problems in certain cases such as the HTML formatting when creating a smart view process. If required, clear the ***Auto-indent*** *check box to disable the text indentation upon saving the process.*

So far, the first step in the Process produces the *record_xml* variable with the complete XML record. This step sends the *record_xml* to the codec plug-in *law_text* input parameter. The codec plug-in unescapes the XML and posts the result in *decode_xml* through its output variable. So now there is a *decode_xml* document ready to be sent to the last step: call the Job.

How to send the XML document to the Job

1. Click the third step to display the **Step Specifications** area where you can define the step parameters.



You can disable the selected step in the open Process editor at any time if you select the **Disable** check box in the **Step Specification** area.

2. From the **Plugin name** list, select *callJob*.

   A description for the selected plug-in displays in the box above the **Plugin name** field.

   The button displays a dialog box that describes the plug-in and details its parameters.

3. Map the *decode_xml* variable defined in the previous step to this step *text* input parameter and click the button to add them to the table.

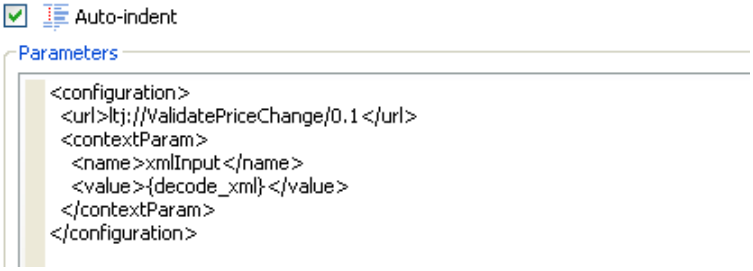
4. Map this step output parameter to a new variable called *output*, for example then click the button to add them to the table.

   For each step, you can select one of the by-default variables or one of the variables defined for the preceding plug-in the Process you define.

5. In the **Parameters** area, click the three-dot button next to the **URL** field to open a dialog box. Here you can set the actual URL to the Job web service by selecting the Job you want to invoke.

6. In the **Parameters** area, set the actual name of the context variable that will receive the XML record.

7. If required, click the **Source** tab in the lower left corner of the Process editor to switch to the text editor. Here you can enter the parameters directly in XML.

   The text you enter in the text editor is indented by default upon saving. This may cause problems in certain cases such as the HTML formatting when creating a smart view process. If required, clear the Auto-indent check box to disable the text indentation upon saving the process.

   The Job will thus need to parse the context variable using a **tXMLExtractXMLField** component. You can send as many context variables as necessary, including literal values.

8. Click the save icon on the toolbar or press **Ctrl + S** on your keyboard to save your changes.

The newly created Process is listed under the **Process** node in the **MDM Repository** tree view.

Now that the Process has been created, you can call it using a Trigger. Once called, it will transform the master data records according to the tasks defined in the Process.

For more information on how to call a specific Process in a Trigger, see section *How to select the service to Trigger and set the service parameters*. For more information on Triggers, see section *Triggers*.

If you have the following error message when trying to call a Process, please verify the possible causes listed below: Error message:

```
ERROR [TISCallTransformerPluginSession]
(EJB-Timer-1271251638944[target=jboss.j2ee:jndiName=amalto/remote/core/
transformerv2ctrl,service=EJB]smile
Could not execute the tisCall transformer plugin javax.xml.ws.WebServiceException:
Could not send Message
```

Possible causes:

-the URL of the Job to call is not correct or the Job is not deployed,

-the context variable does not have the correct name (it has to be the same in the Process and in the Job),

-there is an error in the Job at runtime. Try to deactivate all components in your Job and use only a tFixedFlowInput and a tLogRow components to display the context variable contents. If no error happens, then there is probably an error in your Job.

3.5.1.6. How to create a standalone Process

Like a Runnable Process, a Standalone Process is a process designed in the Studio and can be manually launched by a business user from *Talend MDM Web User Interface*. Unlike the Runnable Process, the Standalone Process is not linked to a specific business entity.

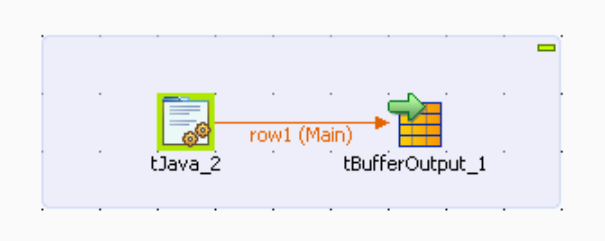
You can design this Process to do any task you want on master data, for example adding a new record/entity in the MDM Hub, launching a Job to do some check or transformation on master data, etc. Both Standalone and Runnable Processes can launch a Job to return a specific result in a new tab in the web interface.

The naming of the Standalone Process follows a specific pattern: *Runnable#[name]*; for example *Runnable#AddNewRecord or Runnable#LaunchJob*.

Prerequisite(s): You have already connected to the MDM server from *Talend Open Studio for MDM*. You have the appropriate user authorization to create Processes. You have already created a Job that will provide the URL for the “output_variable” and deployed it on the MDM server.

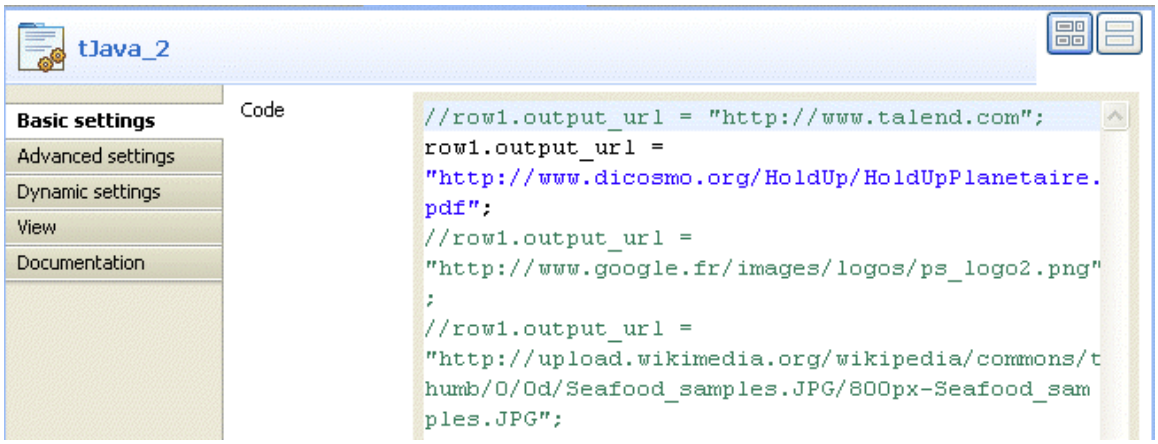
The below example guides you through the steps of creating a Standalone Process that returns a result in *Talend MDM Web User Interface*. It namely calls a Job that provides the URL of a PDF file you want to display in the web interface. You can use the Job to display any type of content, for example Word documents or PDFs, images, web sites, etc.

Start with creating the Job in the **Integration** perspective of your Studio. A simple example of such a Job could be:



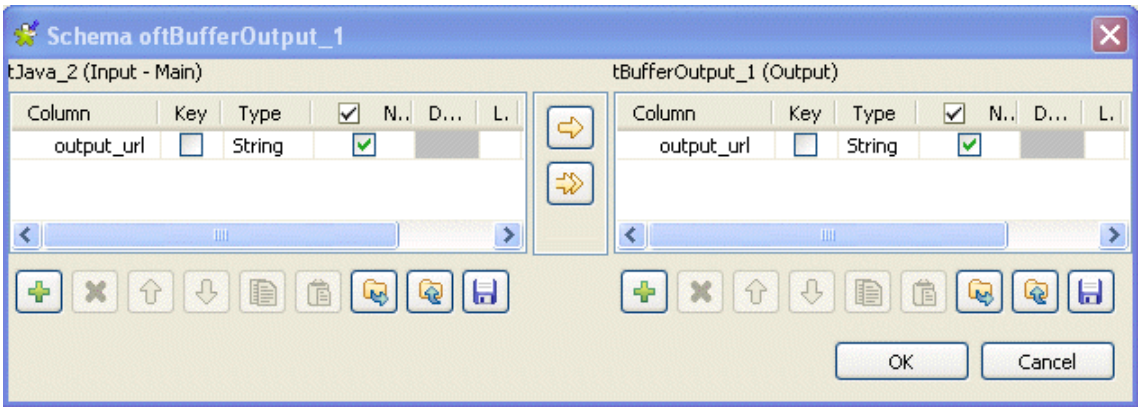
1. Configure the **tJava** component with the Java commands you want to execute.

   This example will generate the URL for a PDF.



The *output_url* is the output variable that is used to return the result in the web interface. You must always define this same variable as the output variable in the **Standalone** process editor, as explained further below.

2. Configure the **tBufferOutput** component to buffer the URL defined in **tJava** and provide it to the output variable you will use when you define the Standalone Process. Drop the column from the **tJava** input schema to the **tBufferOutput** schema.



3. Click **OK** to validate your changes and close the dialog box.

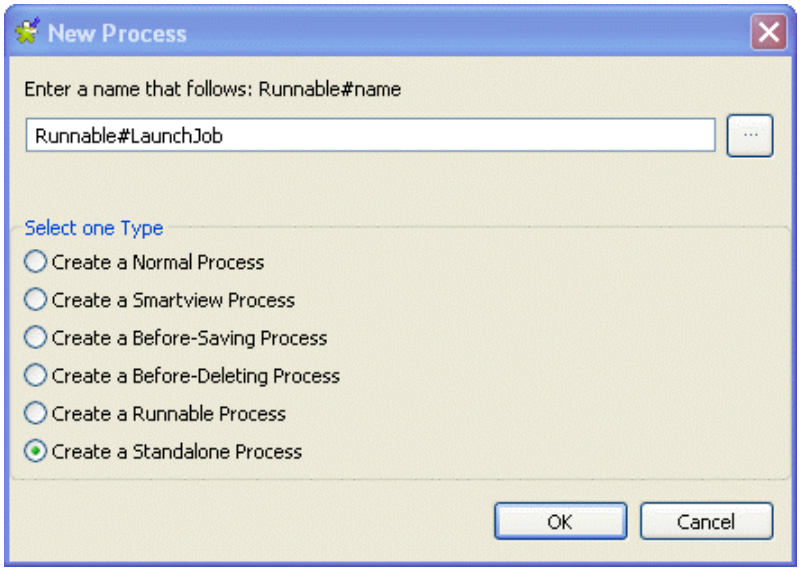
   For further information on **Talend** Jobs and components, see the *Talend Open Studio Components Reference Guide*.

4. Save your Job and then deploy it to the MDM server. For further information on Job deployment, see section *Deploying Jobs automatically on the MDM server*.

To create the Standalone Process that will launch the above Job to display a PDF in a new tab in the MDM web browser, do the following:

1. In the **MDM Repository** tree view, expand **Event Management** and then right-click **Process** and select **New** from the contextual menu.

   The **[New Process]** dialog box displays.



2. Select the **Create a Standalone Process** option.

   Runnable# displays in the name field.

   As the naming for this Process follows certain pattern, *Runnable #* displays automatically in the name field.

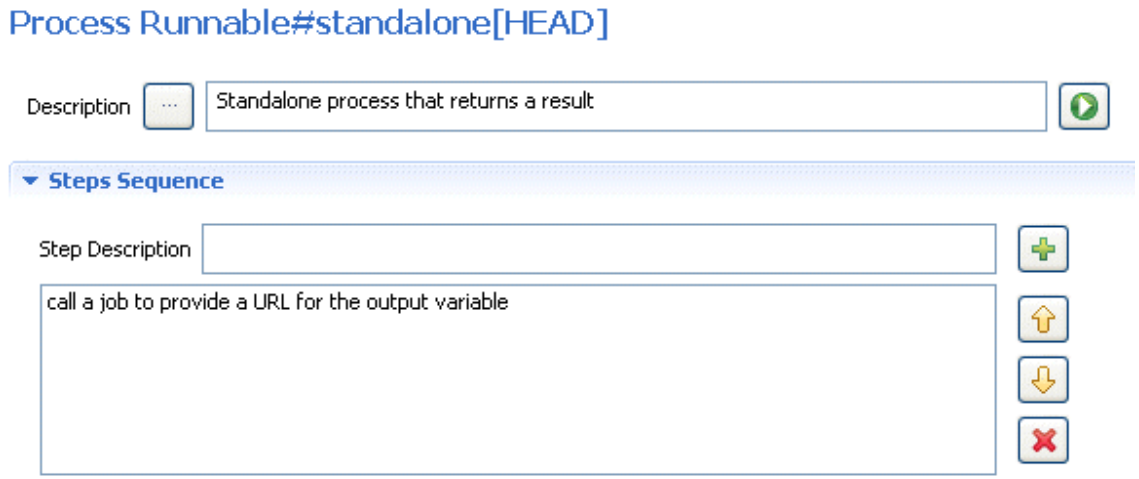
3. Complete the Process name by entering any word(s) after *Runnable#* .

   In this example, the name of the Process is *Runnable#LaunchJob*.

No spaces are allowed in the process name. The OK button in the dialog box will be unavailable as long as there is a space in the instance name.

4. Click **OK** to close the dialog box.

   An empty editor for the newly created Process opens in the workspace.



5. If required, click the three-dot button next to the **Description** field to open a dialog box where you can set multilingual descriptions of your Process.

   This description will be listed as the standalone process name in the **[Welcome]** page in *Talend MDM Web User Interface*. For further information, see *Talend MDM Web User Interface User Guide*.

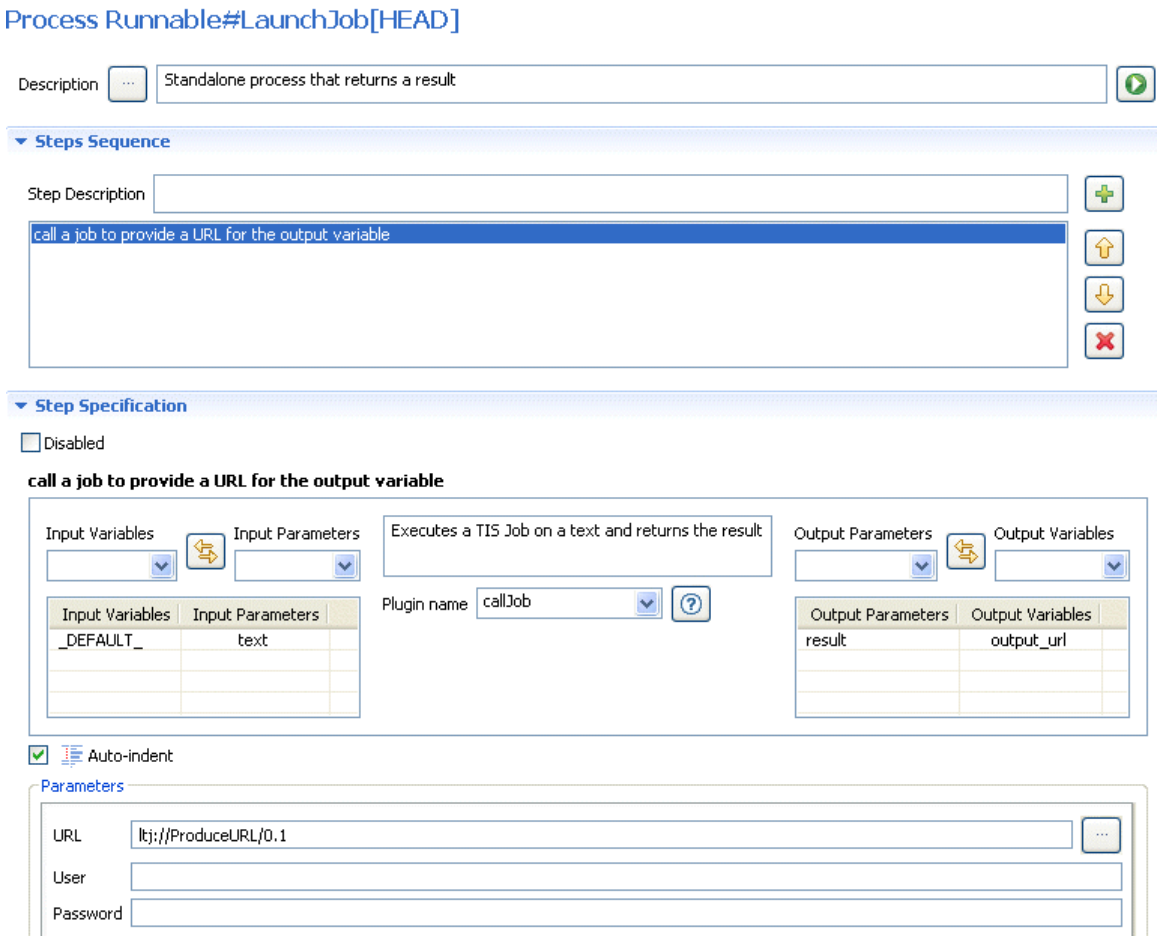
6. In the **Step Description** field, enter a name for the step you want to define in the created Process and then click the  icon to add the step name to the list that follows.

7. If required, do the same to add a second step to do some other task.

The Standalone Process you are going to create here has one step that triggers a **Talend** Job using the *callJob* plug-in. This plug-in consumes data in input parameters and produces a result in the output parameters. The result in this example is to display the content of a PDF in a new tab in the web interface. So the Process calls a Job that by turn provides the URL of the file in an *output_url* variable.

You need now to define the input and output parameters and variables for the listed step(s), only one in this example.

1. Select the step to display the **Step Specification** view in the editor. Here you can define the step input and output parameters and variables.



2. From the **Plugin name** list, select *callJob*.

   The icon displays a description of the plug-in and details its parameters.

3. As the *callJob* plug-in is the first step, leave the input variable empty, select the *text* input, and click the button to add them to the table.

   The MDM server starts the pipeline with a default variable called *_DEFAULT_* .

   Here you want to map variables to the input parameter of the plug-in, and conversely map output parameter of a plug-in to another variable.

4. Select the output parameter *result* and define the output variable, *output_url*, and then click the button to add them to the table.

5. In the **Parameters** area, click the three-dot button to display a list of all **Talend** Jobs that have been deployed on the MDM server.

6. Select the Job you want to attach to the Process, *ProduceURL* in this example.

   If you have more than one step in your Process, you must define variables to hold the result of the first step. Then you send the variable to the input of the second step. Eventually, you define a “pipeline” where each step result is chained to the next step through a variable. For each step, you can select one of the by-default variables or one of the variables defined for the preceding plug-in the Process you define.

   You can disable the selected step in the editor at any time if you select the **Disable** check box.

7. Save your modifications.

   The Standalone Process is listed under the **Process** node in the **MDM Repository** tree view.

This same process is accessible from *Talend MDM Web User Interface*. An authorized business user can launch this Process and display the result from the **[Wecome]** page in the Web User Interface. For further information, see *Talend MDM Web User Interface User Guide*.

3.5.1.7. How to create a smart view Process

Principles

A smart view is a customized, business-oriented view of a data record in the MDM Hub.

A smart view basically renders an HTML presentation of the details of a data record held in a specific entity. Whenever a business user tries to browse a data record through *Talend MDM Web User Interface*, Talend MDM checks for a smart view for such entity. If it finds the smart view, it uses this view to render the HTML presentation of the record detail instead of displaying the “conventional” generated form. The business user can then switch back and forth between the smart view presentation and the generated form.

A smart view is not a view but an XSLT-based process that must have an XSLT step which transforms the XML document using XSLT. The pipeline for this XSLT step must have an output variable called *html*. However, a smart view can have steps other than the XSLT one that can perform different tasks on master data, for further information, see section *Processes*.

A smart view has a naming convention: `Smart_view_<Entity>[_<ISO2>][<#name>]`. The two characters country ISO code is optional and it allows you to define multilingual smart views. The `<#name>` suffix is also optional and allows you to define several smart views for the same entity. For further information, see section *How to create an alternate smart view of a data record*.

At runtime, when a user tries to open a record in an entity through *Talend MDM Web User Interface*, say the *Product* entity for example, here is what happens:

1. Talend MDM first looks for all Processes that begins with *Smart_view_Product*,
2. If it finds one with the `_<ISO2>` suffix (e.g. *Smart_view_Product_ENU* in the U.S., *Smart_view_Product_FR* in France etc.) it uses it,
3. It then sends the XML record into the *_DEFAULT_* variable and executes the Process,
4. When the Process completes, it looks for an output variable called *html*,
5. It finally sends the content of the *html* variable back to the browser.

How to create a “default” smart view of a data record

A smart view is an alternative to the “conventional” generated view used to display the data record detail in *Talend MDM Web User Interface*. A smart view is a customized view that uses an XSLT step to render the HTML presentation from the incoming XML record.

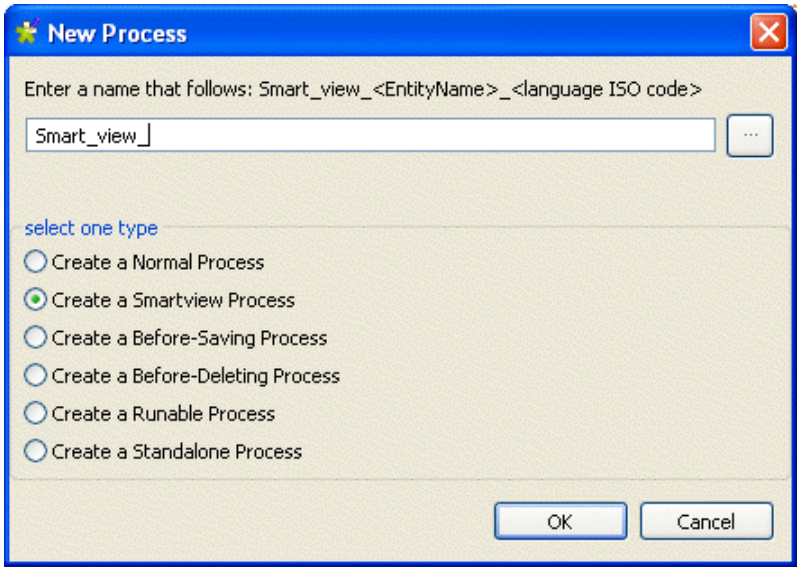
You can either create the HTML elements directly in the parameters of the XSLT step in the Process editor, or create a web template outside *Talend Open Studio for MDM* and then paste the HTML into the parameters of the XSLT step in the Process editor.

Prerequisite(s): You have already connected to the MDM server from *Talend Open Studio for MDM*. You have the appropriate user authorization to create Processes.

To create a smart view of a data record, do the following:

1. In the **MDM Repository** tree view, expand **Event Management** and then right-click **Process** and select **New** from the contextual menu.

   The **[New Process]** dialog box displays.



2. Select the **Create a SmartView Process** option.

   *Smart_View_* displays in the name field.

   As the naming for this Process follows certain pattern, *Smart_View_* displays automatically in the name field.

3. Complete the Process name by entering the name of the entity on which you want to run the Process and a two-character country ISO code for the language. The process name will look as the following in this example: *Smart_view_Product_EN*.

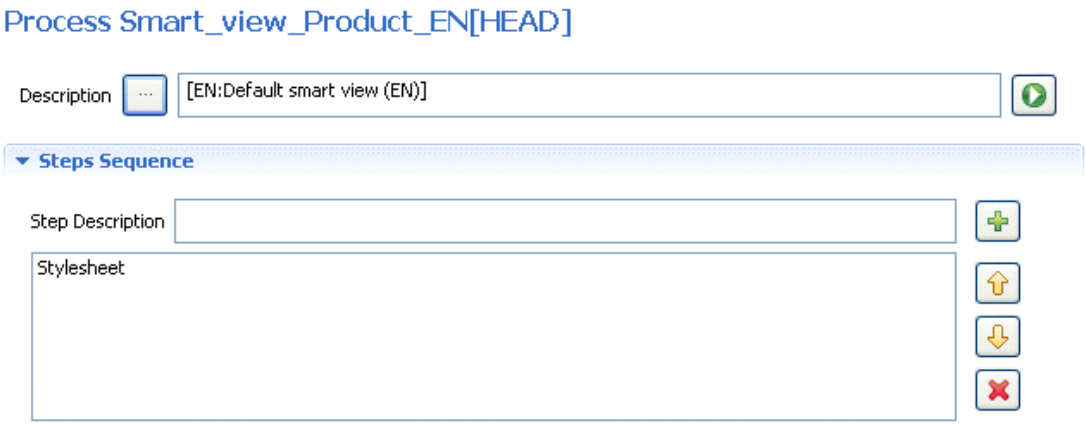
   No spaces are allowed in the process name. The OK button in the dialog box will be unavailable as long as there is a space in the instance name.

   You can use a `<#name>` suffix at the end of the smart view name to define several smart views for the same entity. for further information, see section *How to create an alternate smart view of a data record*.

   In this example, we assume that you have a *Product* data model that has a *Product* business entity. Many attributes have been defined for this entity including: *Name*, *Description*, *Features (Sizes and Colors)*, *Availability* and *Price*.

4. Click **OK** to close the dialog box.

   An editor for the newly created Process opens in the workspace with a *Stylesheet* step already listed in the **Step Sequence** area.



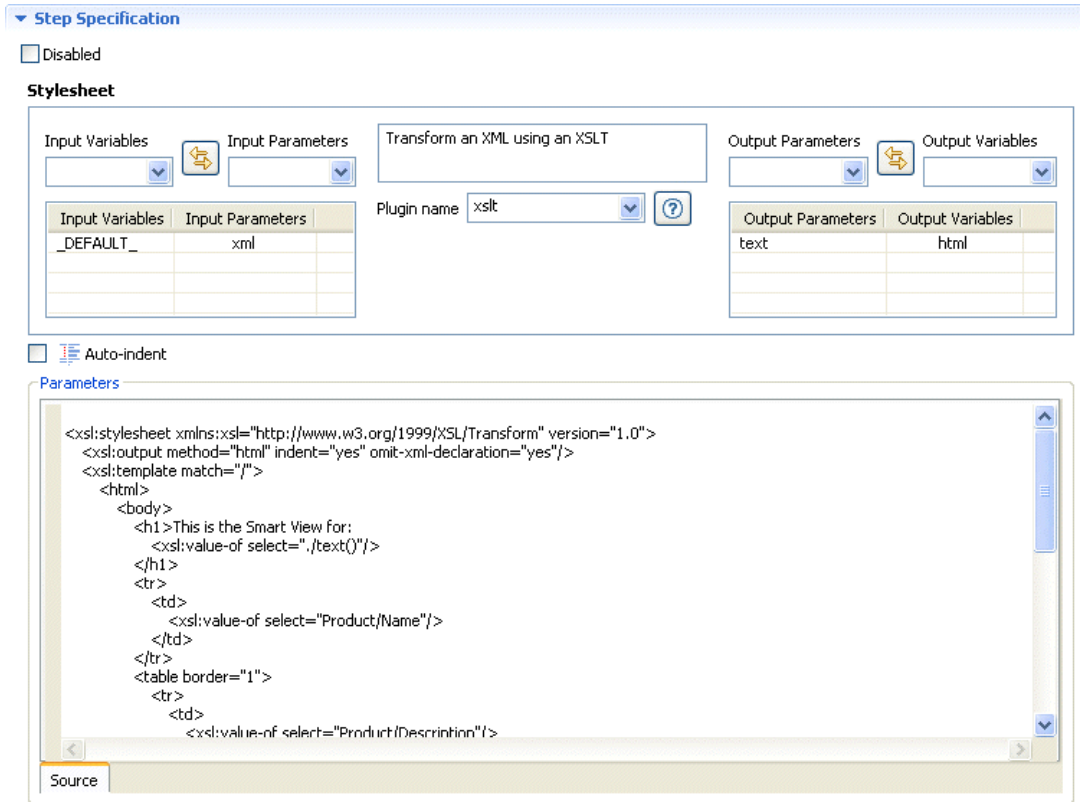
5. If required, click the three-dot button next to the **Description** field to open a dialog box where you can set multilingual descriptions of your Process.

   This description will be listed as the smart view name in the smart view list in *Talend MDM Web User Interface*. For further information, see *Talend MDM Web User Interface User Guide*.

   When you create a new smart view Process, an XSLT step, with the input and output variables, is automatically added. Basic HTML elements are also automatically added in the **Parameters** area.

   You can always add new steps in the Process if you enter the step name in the **Step Description** field and then click the icon. For further information, see section *How to create a Process from scratch*.

6. Select the **Stylesheet** step in order to display the **Step Specifications** and the **Parameters** areas of the selected XSLT step.



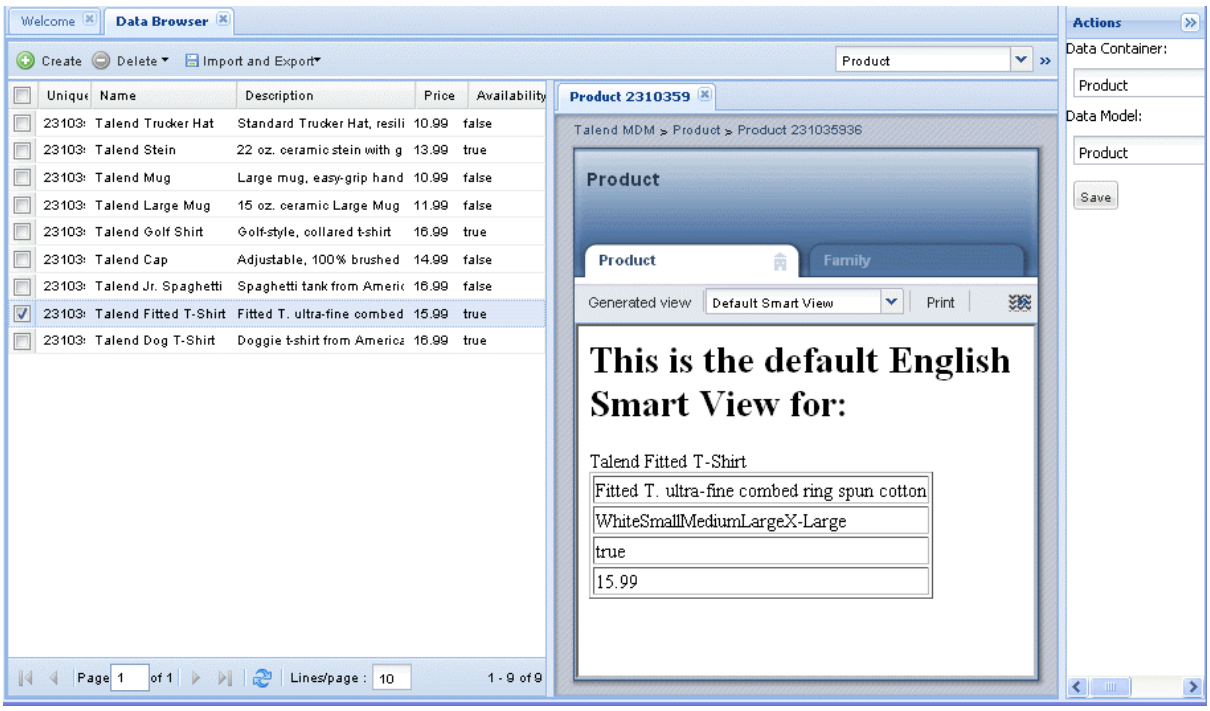
7. In the **Parameters** area, customize the HTML default elements according to your needs and save the Process.

The text you enter in the text editor will be indented by default upon saving. This may cause problems in HTML formatting when creating the smart view Process. Clear the Auto-indent check box to disable the text indentation upon saving the process.

If you define the HTML parameters as the following for the *Product* entity:

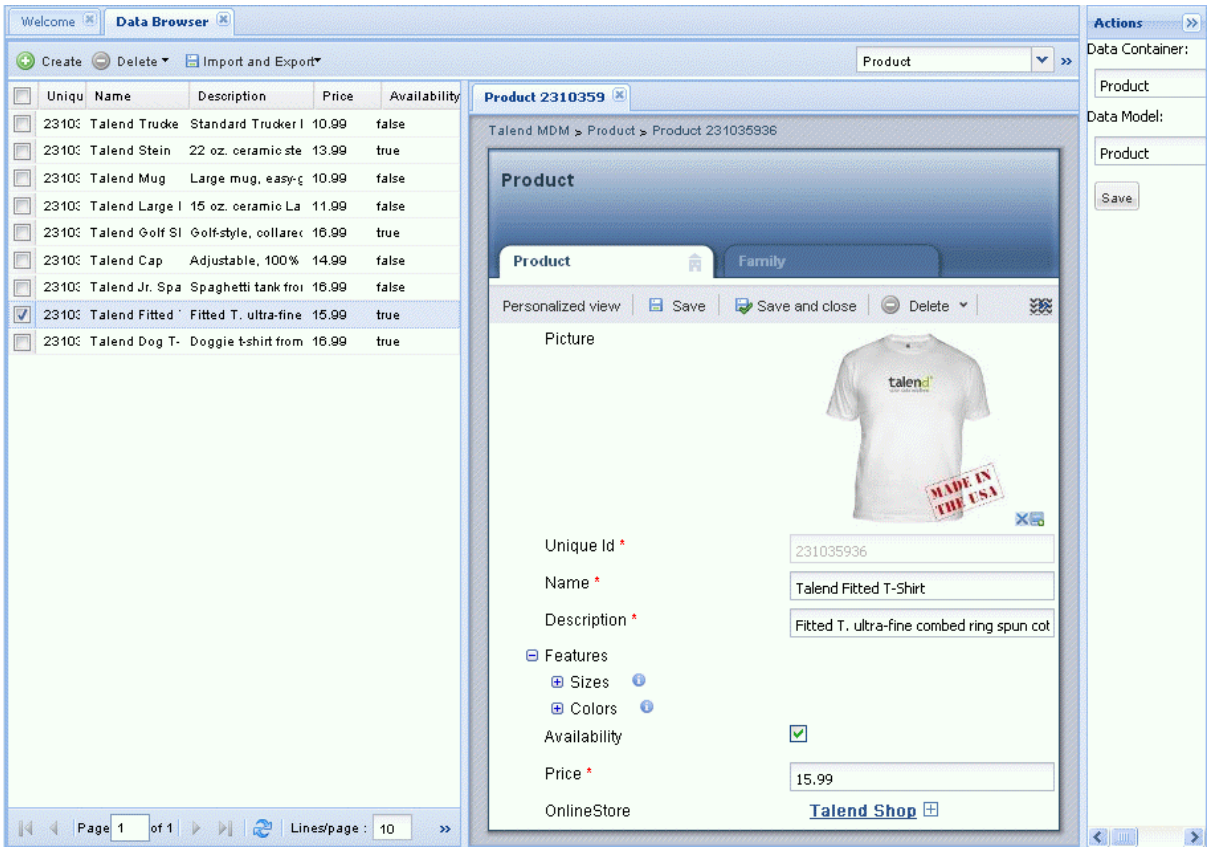
```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="1.0">
  <xsl:output method="html" indent="yes" omit-xml-declaration="yes"/>
  <xsl:template match="/">
   <html>
     <head>
       <title>Product</title>
     </head>
     <body>
       <h1>This is the default ENGLISH Smart View for:
         <xsl:value-of select="./text()"/>
       </h1>
       <tr>
         <td>
            <xsl:value-of select="Product/Name"/>
         </td>
       </tr>
       <table border="1">
       <tr>
         <td>
          <xsl:value-of select="Product/Description"/>
         </td>
       </tr>
       <tr>
         <td>
            <xsl:value-of select="Product/Features"/>
         </td>
       </tr>
       <tr>
         <td>
            <xsl:value-of select="Product/Availability"/>
         </td>
       </tr>
       <tr>
         <td>
            <xsl:value-of select="Product/Price"/>
       </td>
       </tr>
     </table>
   </body>
  </html>
 </xsl:template>
</xsl:stylesheet>
```

Then, every time a business user accesses *Talend MDM Web User Interface* and tries to browse records in the *Product* entity, he/she will get this smart view by default:



From this view in *Talend MDM Web User Interface*, the business user can have access to any Runnable processes created on the selected entity. He/she can run any of the listed process to initiate the tasks listed in the Process.

The business user can switch to the “conventional” generated form by clicking **Generated View** on the menu bar:



The business user can always switch back and forth between the smart view and the generated view through clicking the **Personalized View** and **generated View** tabs respectively in the **Browse Record** view in *Talend MDM Web User Interface*.

You can always disable the smart view process if you select the **Disable** check box in the Process editor.

For further information on how to customize the smart view parameters if you have Cascading Style Sheets (CSS) and JavaScript resources, see section *HTML resources* and section *Foreign Keys and cross referencing*.

How to create an alternate smart view of a data record

A smart view is an alternative to the "conventional" generated view used to display the data record detail in *Talend MDM Web User Interface*. For further information on smart views, see section *How to create a "default" smart view of a data record*.

An alternate smart view is a customized view defined with the suffix `<#name>` at the end of the view name: `Smart_view_<Entity>[_<ISO2>][<#name>]`. This suffix enables you to create several alternate smart views for the same entity. For example `Smart_view_Product_EN` is the "default" smart view for the *Product* entity; and `Smart_view_Product_EN#version1` and `Smart_view_Product_EN#version2` are two possible alternates smart views for the *Product* entity.

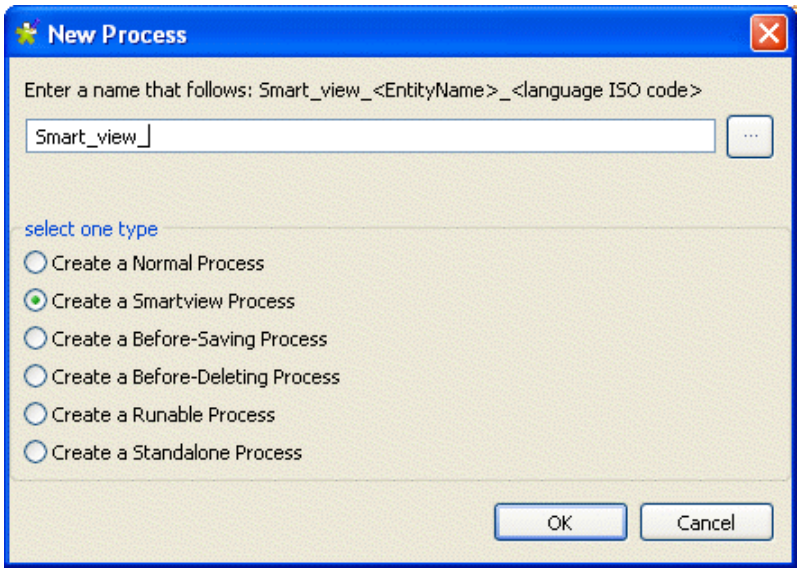
All alternates smart views you create on a specific entity are listed in a drop-down list in the **Browse Records** view in *Talend MDM Web User Interface*.

Prerequisite(s): You have already connected to the MDM server from *Talend Open Studio for MDM*. You have the appropriate user authorization to create Processes.

To create an alternate smart view of a data record, do the following:

1. In the **MDM Repository** tree view, expand **Event Management** and then right-click **Process** and select **New** from the contextual menu.

   The **[New Process]** dialog box displays.



2. Select the **Create a SmartView Process** option.

   As the naming for this Process follows certain pattern, *Smart_View_* displays automatically in the name field.

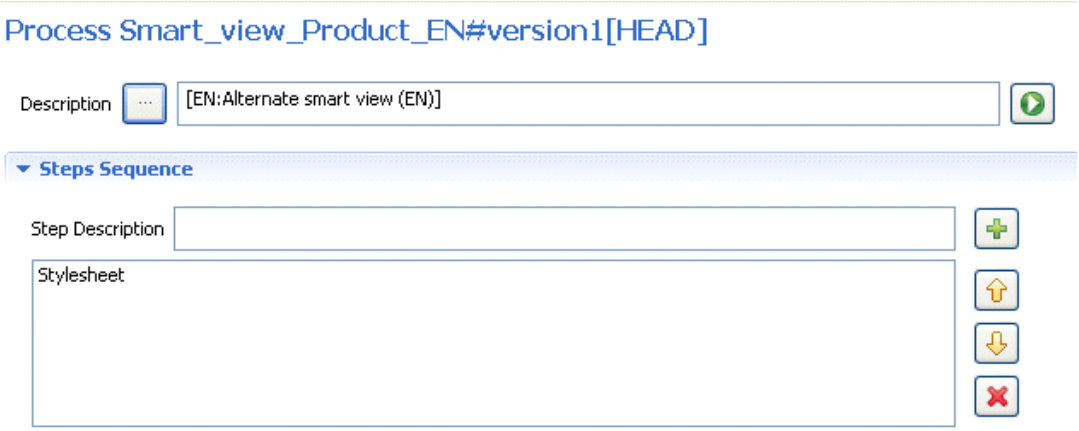
3. Complete the Process name by entering the name of the entity on which you want to run the Process, a two-character country ISO code for the language and a name for the alternate smart view. The process name will look as the following in this example: *Smart_view_Product_EN#alternate_smart_view*.

No spaces are allowed in the process name. The OK button in the dialog box will be unavailable as long as there is a space in the instance name.

In this example, we assume that you have a *Product* data model that has a *Product* business entity. Many attributes have been defined for this entity including: *Name*, *Description*, *Features (Sizes and Colors)*, *Availability* and *Price*.

4. Click **OK** to close the dialog box.

   An editor for the newly created Process opens in the workspace with a *Stylesheet* step already listed in the **Step Sequence** area.



5. If required, click the three-dot button next to the **Description** field to open a dialog box where you can set multilingual descriptions of your Process.

   This description will be listed as the smart view name in the smart view list in *Talend MDM Web User Interface*. For further information, see *Talend MDM Web User Interface User Guide*. However, if you do not enter a description in the **Description** field, the name suffix used in the smart view name, *version1* in this example, will be used to list this view in *Talend MDM Web User Interface*.

   When you create a new smart view Process, an XSLT step, with the input and output variables, is automatically added. Basic HTML elements are also automatically added in the **Parameters** area.

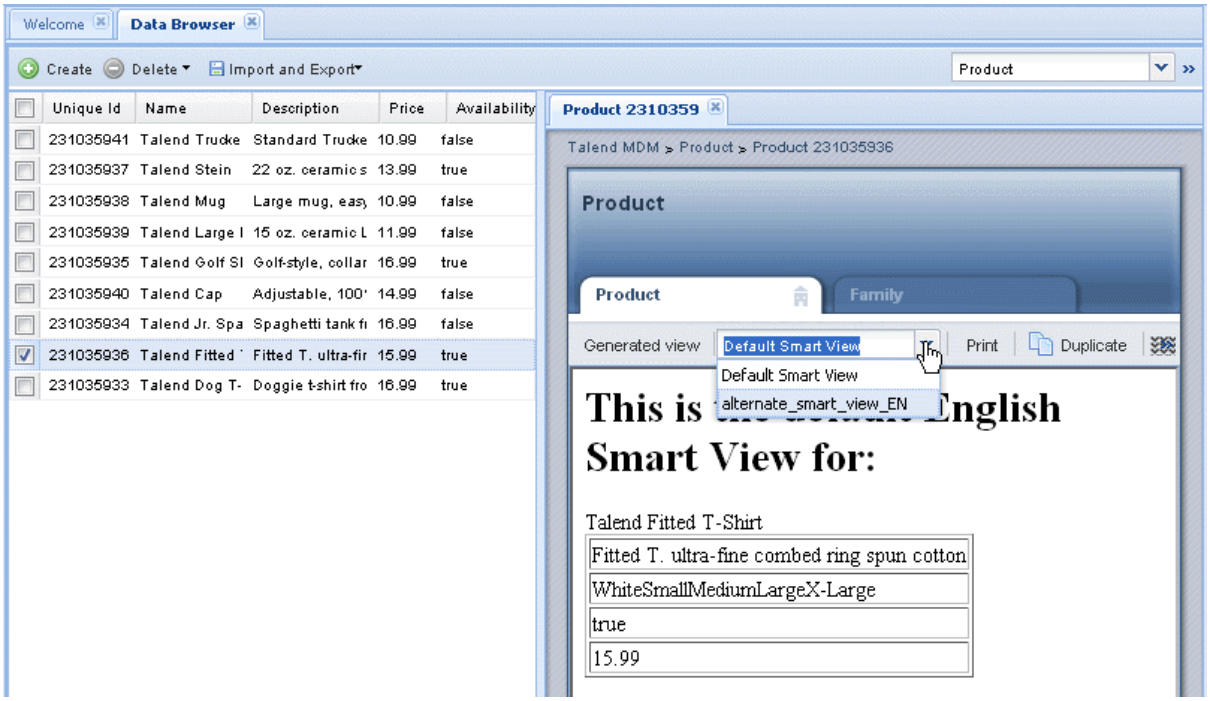
   You can always add new steps in the Process if you enter the step name in the **Step Description** field and then click the icon. For further information, see section *How to create a Process from scratch*.

6. Select the **Stylesheet** step and define your HTML according to your needs as outlined in section *How to create a "default" smart view of a data record*.

Then, every time a business user accesses *Talend MDM Web User Interface* and tries to browse records in the *Product* entity, he/she will:

- get the "default" smart view if one has already been defined on the business entity, or
- get this alternate smart view if a "default" smart view has not been defined.

The business user can always switch back and forth between the "conventional" generated view of the entity and any of the smart views created on the same entity.



If you create only alternate smart views for a specific entity without creating a “default” smart view, the **Browse Records** page in *Talend MDM Web User Interface* will always open on the “conventional” generated view of the data record, and the business user can then select to open any of the listed alternates smart views.

You can always disable the alternate smart view process if you select the **Disable** check box in the Process editor.

For further information on how to customize the smart view parameters if you have Cascading Style Sheets (CSS) and JavaScript resources, see section *HTML resources* and section *Foreign Keys and cross referencing*.

How to create a smart view Process through a template

A smart view uses an XSLT step to render the HTML presentation from the incoming XML record. The easiest thing to do is to create an HTML template with hard-coded values outside *Talend Open Studio for MDM*:

```
<tr>
<td>Product Name</td><td>PROD NAME</td>
<!-- etc -->
</tr>
```

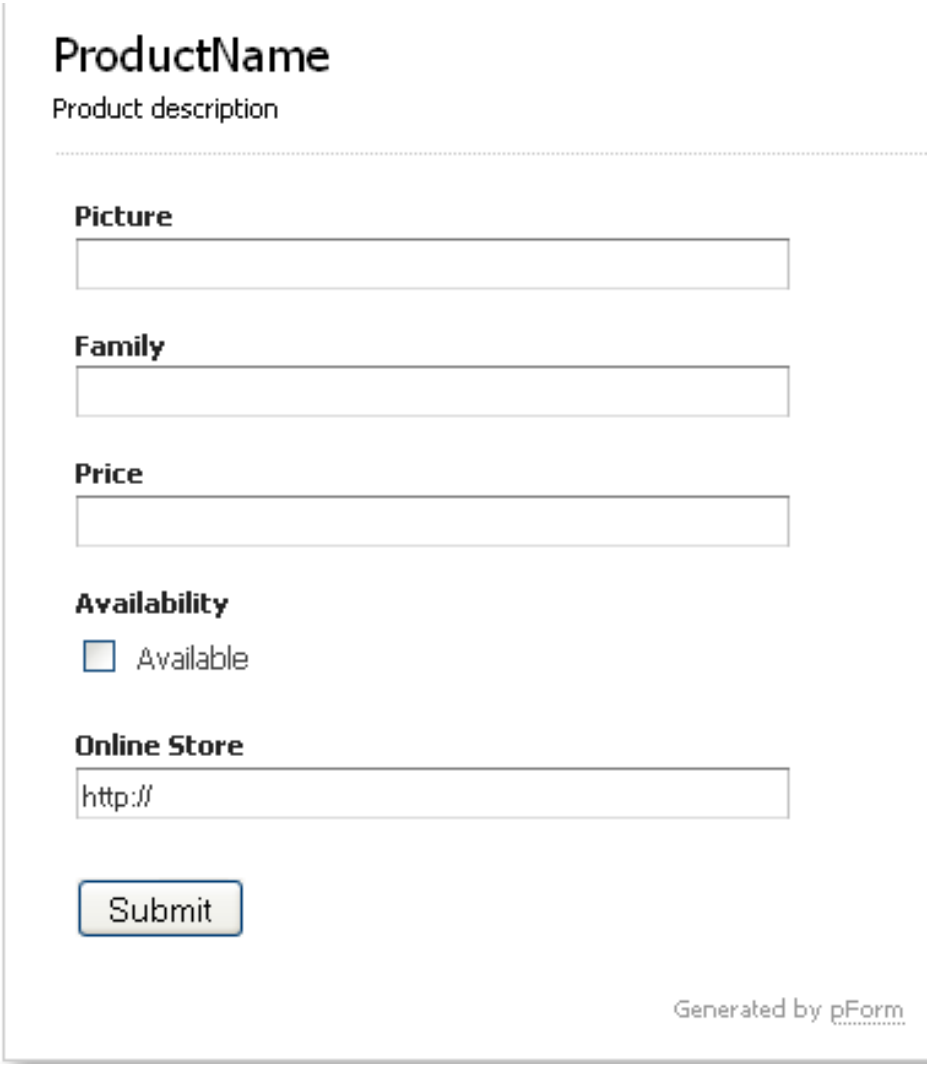
Then copy/paste this template into the body of the XSLT stylesheet in the Process editor, and replace the hard-coded value by `<xsl:value-of>` statements:

```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="1.0">
    <xsl:output method="html" indent="yes"/>
    <xsl:template match="/" priority="1">
        <html>
            <head>
                <title>Product</title>
            </head>
            <body>
<tr>
<td>Product Name</td><td><xsl:value-of select="Product/Name"/></td>
<!-- etc -->
</tr>
            </body>
```

```
        </html>
   </xsl:template>
</xsl:stylesheet>
```

Use, for instance, http://www.phpform.org/ to create a web template of your form. A web template consists of one or several html files in addition to the resources (JavaScript, CSS, images, etc.). The resources must be accessible from within MDM, so unzip the web template in a new “smartview” folder within *jboss-4.2.2.GA/server/default/deploy/jboss-web.deployer/ROOT.war*.

1. Unzip the web template in a new *smartview* folder within *jboss-4.2.2.GA/server/default/deploy/jboss-web.deployer/ROOT.war*.



   Your form should now be accessible through: http://localhost:8180/smartview/Product/form.html.

2. Open the form in a text editor and check the html elements.

   XSLT is XML, and XML is not as lenient as html. You will find start tags with no end tags (`meta`, `img`, `link`, etc.). Make sure all html elements are written with start and end tags.

3. Fix the URL to the JS, CSS and images. Change all the `src` attributes to point to: */smartview/Product/*<*resource*> instead of just <*resource*>. For instance:

   Change:

   ```
   <script type="text/javascript" src="view.js"></script>
   ```

   to: `<script type="text/javascript" src="/smartview/Product/view.js"></script>`.

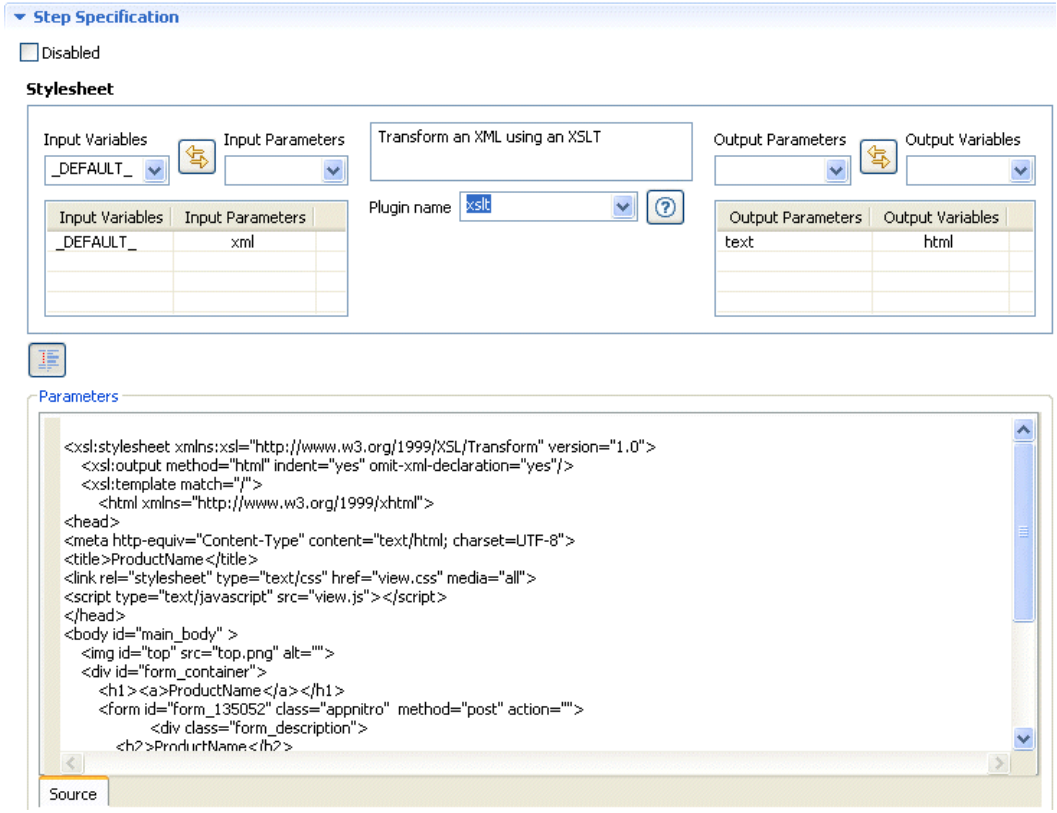
4. Create a smart view Process in *Talend Open Studio for MDM* as outlined in section *How to create a "default" smart view of a data record* and call it *Smart_view_Product*.

5. Change <xsl:template match="/"> to xsl:template match="/Product">, assuming the entity is *Product*.

6. Copy the html code, without `<!DOCTYPE>`, from the html tag and paste it within the

```
<xsl:template
match="/Product">
```

element. Overwrite anything that was already within the `<xsl:template>` element:



7. Set the field values with the actual values extracted from the *Product* record using the `<xsl:value-of>` statement. To ask XSLT to output an attribute, use the `<xsl:attribute>` statement.

As an example, here is the html code before the change:

```
<label class="description" for="element_2">Price
</label>
<div>
    <input id="element_2" name="element_2" class="element text   medium"
type="text" maxlength="255" value=""/>
</div>
```

And after the change:

```
<label class="description" for="element_2">Price
</label>
<div>
    <input id="element_2" name="element_2" class="element text medium"
type="text" maxlength="255">
<xsl:attribute name="value"><xsl:value-of select="Price"/></xsl:attribute>
    </input>
</div>
```

This tells XSLT to issue a "value" attribute within the `<input>` tag and to fetch the value of this attribute from the *Price* element of the *Product* record.

The *Product* attribute used in this example has a foreign key to *ProductFamily*. You may want to add a step before the xslt step in the smart view Process and use a *crossreferencing* plugin to resolve the FK before it reaches the html rendering. Below is the *crossreferencing* configuration:

```
<parameters>
   <CrossRef>
      <xrefName>resolvefk</xrefName>
      <xrefCluster>Product</xrefCluster>
      <xrefRootElement>/Product</xrefRootElement>
      <xrefIn>
         <mapping>
             <xrefElement>Family</xrefElement>
             <xrefPath>ProductFamily/Id</xrefPath>
         </mapping>
      </xrefIn>
      <xrefOut>
         <mapping>
             <xrefElement>Family</xrefElement>
             <xrefPath>ProductFamily/Name</xrefPath>
         </mapping>
      </xrefOut>
   </CrossRef>
</parameters>
```

For further information on the *crossreferencing* plugin, see section *Important plug-ins* and section *Example of the crossreferencing plug-in and its parameters*.

HTML resources

If you have Cascading Style Sheets (CSS) and JavaScript resources, they need to be URL-addressable. You can do that through copying them in JBoss Web.

1. Create a directory in *<MDM Server home>\jboss-4.2.2.GA\server\default\deploy\jboss-web.deployer \ROOT.war*; *Product* for instance.
2. Drop your CSS and JS files in the directory.
3. You can now define your resources in the `<head>` section of the XSLT parameters in the Process editor as the following:

```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="1.0">
    <xsl:output method="html" indent="yes"/>
    <xsl:template match="/" priority="1">
        <html>
<head>
<title>Product</title>
<link rel="stylesheet" type="text/css" href="/product/greatlookandfeel.css"
media="all">
<script type="text/javascript" src="/product/javascriptsuff.js"></script>
</head>
<!-- rest omitted -->
```

Foreign Keys and cross referencing

Talend MDM sends the XML record, without resolving the FKs. It is up to the smart view to lookup/resolve them.

As the Process is a multi-step pipeline, you can have one or several steps before the final XSLT step. So you will need to insert a step to resolve your FKs and/or cross referencing. You can use a **Talend** Job deployed as a ZIP or you can use a <parameters>

<CrossRef>

<xrefName>Country</xrefName>

<xrefCluster>crossreferencing</xrefCluster>

<xrefRootElement>/Product</xrefRootElement>

<xrefIn>

<mapping>

<xrefElement>CountryCode</xrefElement>

<xrefPath>Countries/ISO2</xrefPath>

</mapping>

</xrefIn>

<xrefOut>

<mapping>

<xrefElement>CountryName</xrefElement>

<xrefPath>Countries/Name</xrefPath>

</mapping>

</xrefOut>

</CrossRef>

</parameters> step.

First example: if the data model is *Product* with a FK to *ProductFamily*, use the following parameters to resolve the product family, which is the raw family code (e.g. ”[1234]”), to the actual family name:

```
<parameters>
   <CrossRef>
       <xrefName>FamilyFK</xrefName>
       <xrefCluster>Product</xrefCluster>
       <xrefRootElement>/Product</xrefRootElement>
       <xrefIn>
            <mapping>
                <xrefElement>Family</xrefElement>
                <xrefPath>ProductFamily/Id</xrefPath>
            </mapping>
       </xrefIn>
       <xrefOut>
            <mapping>
                <xrefElement>Family</xrefElement>
             <xrefPath>ProductFamily/Name</xrefPath>
        </mapping>
      </xrefOut>
   </CrossRef>
</parameters>
```

Second example: if there is a *Countries* cross reference table with `key = ISO2` and *value* = *Name* and you want to resolve *Product/CountryCode* into *Product/CountryName*, use the following parameters:

```
<parameters>
   <CrossRef>
       <xrefName>Country</xrefName>
       <xrefCluster>crossreferencing</xrefCluster>
       <xrefRootElement>/Product</xrefRootElement>
       <xrefIn>
          <mapping>
```

```
            <xrefElement>CountryCode</xrefElement>
            <xrefPath>Countries/ISO2</xrefPath>
         </mapping>
      </xrefIn>
      <xrefOut>
         <mapping>
            <xrefElement>CountryName</xrefElement>
            <xrefPath>Countries/Name</xrefPath>
         </mapping>
      </xrefOut>
   </CrossRef>
</parameters>
```

For further information on the *crossreferencing* plugin, see section *Important plug-ins* and section *Example of the crossreferencing plug-in and its parameters*.

3.5.1.8. Managing Processes

An authorized user can also execute, import/export, copy/paste and delete the listed Processes from *Talend Open Studio for MDM*.

It is also possible to import and share MDM complete projects or only a data model or part of the data model from the community web page, **Talend Exchange**. For further information, see section *Projects/objects on Talend Exchange*.

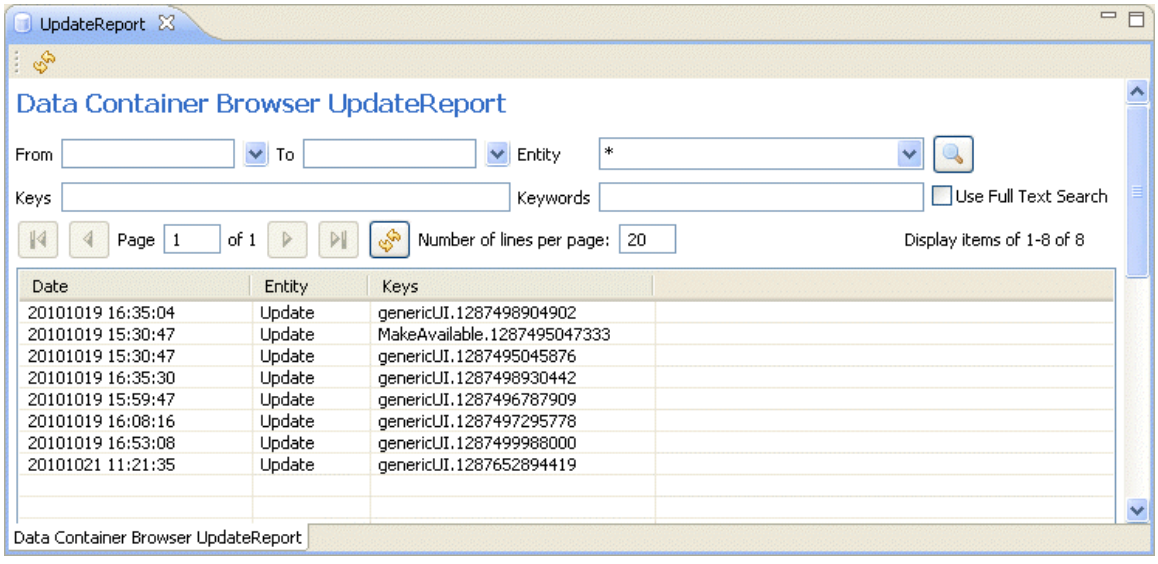
How to test a Process in the Studio

After creating your Process in *Talend Open Studio for MDM*, this Process is usually called by **Talend** MDM web service. However, it is possible to run the Process from within the Studio itself.

Prerequisite(s): You have already connected to the MDM server from *Talend Open Studio for MDM*. At least one Process exists.

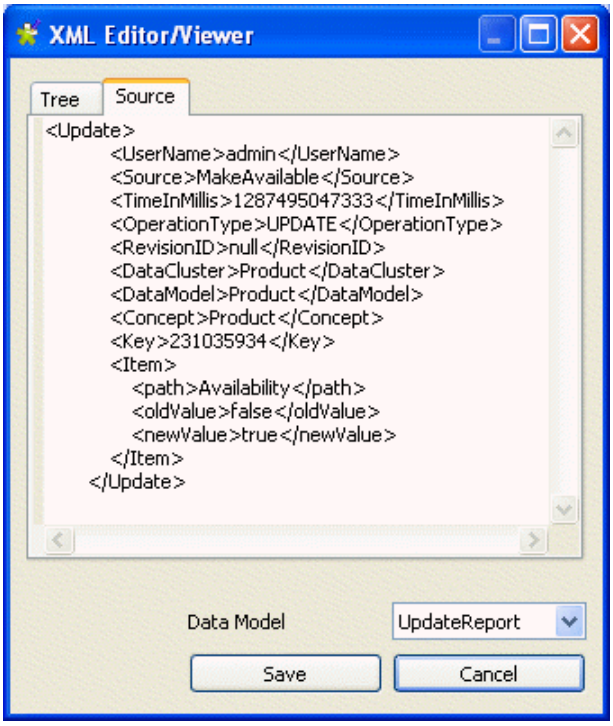
To execute a process in the Studio:

1. In the **MDM Repository** tree view, expand the **Data Container** and **System** nodes and then double-click **UpdateReport** to open the corresponding editor.



2. Click the  icon to display the list of all the documents generated upon a create/update/delete operation done on any data record.

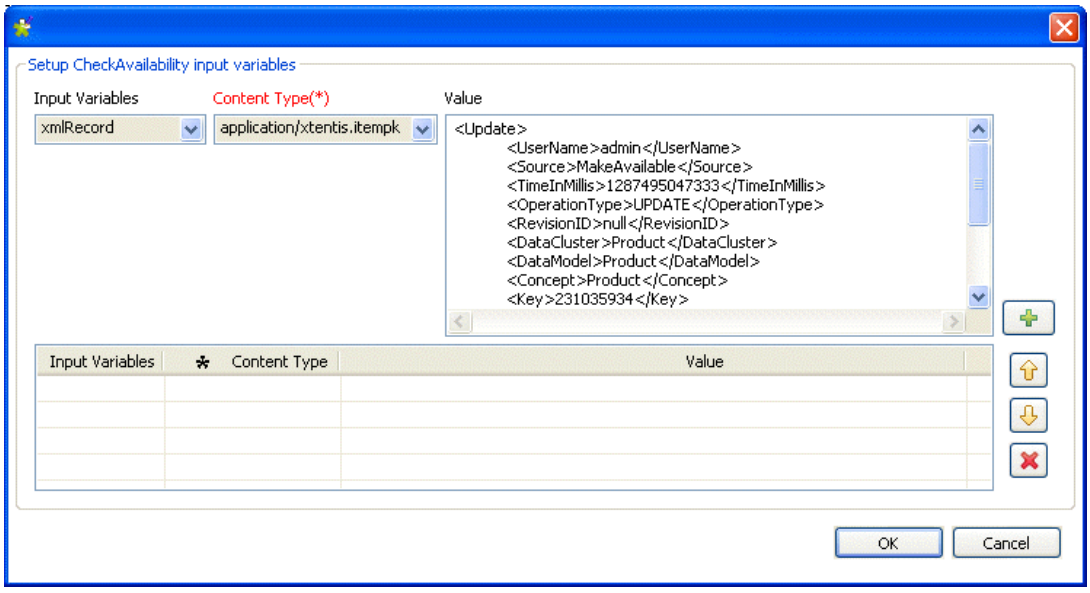
3. Double-click the document you want to test in order to display its detail in a dialog box.



4. From the **Source** view, select the XML text and then click **Cancel** to close the dialog box.

5. Double-click the Process you want to test and then click the  icon in the upper right corner of the process editor.

   A dialog box displays.



6. From the **Input Variables** list, select *_DEFAULT_* and select *text/xml* from the **Content Type** list.

7. Paste the document XML text in the **Value** table and then click the  icon to list the set variables and information in the lower part of the dialog box.

8. Click **OK** to run the Process.

A progress information bar displays and then a result dialog box opens. Here you can browse the latest value of all the Process pipeline as well as which records are sent to the Data Container.

This result will show that your Process works correctly. What you must do now is to define a Trigger in order to call the process from that Trigger. For further information, see section *Creating a Trigger*.

How to export Processes

From *Talend Open Studio for MDM*, you can export one or multiple Processes in order to exchange them between:

- two different MDM Servers or Repositories,
- two different Versions from the same/different MDM Servers or Repositories, for example.

The steps to export one or multiple Processes are similar to those for any other data object in the **MDM Repository** tree view. For detailed information on how to export Processes, see section *How to export data models*.

How to import Processes

From *Talend Open Studio for MDM*, you can import Processes created on other MDM servers, in different Versions on the same MDM server, or in different MDM Repositories into the current MDM Repository.

The steps to import one or multiple Processes are similar to those for any other data object in the **MDM Repository** tree view. For detailed information on how to import Processes, see section *How to import data models*.

How to edit a Process

You can open a Process you have already created to check its settings and/or edit the defined parameters.

Prerequisite(s): You have already connected to the MDM server from *Talend Open Studio for MDM*. At least one Process exists.

To edit a Process:

1. In the **MDM Repository** tree view, expand the **Processes** node.
2. Right-click the Process you want to edit and select **Edit** from the contextual menu.

   The editor opens on the selected Process in the workspace.
3. Modify the Process parameters as needed and then click the save icon on the toolbar or press **Ctrl + S** on your keyboard to save your changes.

The selected Process is modified accordingly.

If you try to update a Process that has been modified by somebody else after you have retrieved it from the database, a warning message displays to warn you that saving your modifications will overwrite the other user's changes.

How to copy/paste a Process

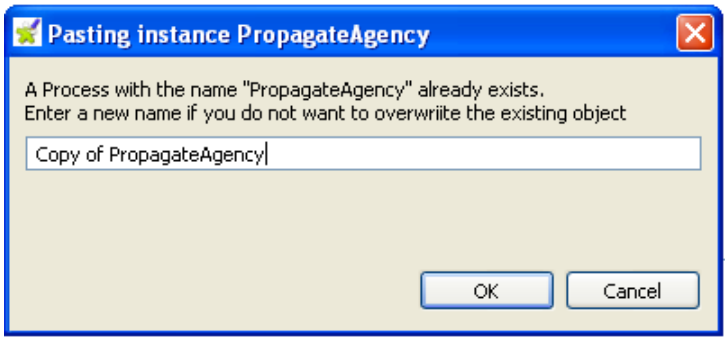
To avoid creating one or multiple Processes from scratch, you can copy an existing one in the **MDM Repository** tree view and modify its parameters to have a new Process.

Prerequisite(s): You have already connected to the MDM server from *Talend Open Studio for MDM*. At least one Process exists.

To copy/paste a Process, do the following:

1. In the **MDM Repository** tree view, expand the **Processes** node.
2. Right-click the Process you want to duplicate and select **Copy** from the contextual menu.
3. Right-click the **Processes** node and select **Paste** from the contextual menu.

   A dialog box displays prompting you to enter a name for the new Process.



4. Enter a name for the new Process and click **OK** to validate the changes and close the dialog box.

   The new Process is listed under the **Processes** node in the **MDM Repository** tree view.

How to duplicate a Process

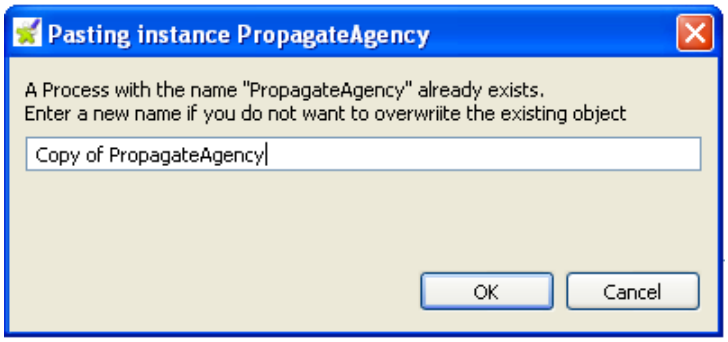
To avoid creating one or multiple Processes from scratch, you can duplicate an existing one in the **MDM Repository** tree view and modify its parameters to have a new Process.

Prerequisite(s): You have already connected to the MDM server from *Talend Open Studio for MDM*. At least one Process exists.

To duplicate a Process, do the following:

1. In the **MDM Repository** tree view, expand the **Processes** node.
2. Right-click the Process you want to duplicate and select **Duplicate** from the contextual menu.

   A dialog box displays prompting you to enter a name for the new Process.



3. Enter a name for the new Process and click **OK** to validate the changes and close the dialog box.

   The new Process is listed under the **Processes** node in the **MDM Repository** tree view.

You can also duplicate the data object if you drop it onto its parent node in the **MDM Repository** tree view.

How to delete a Process

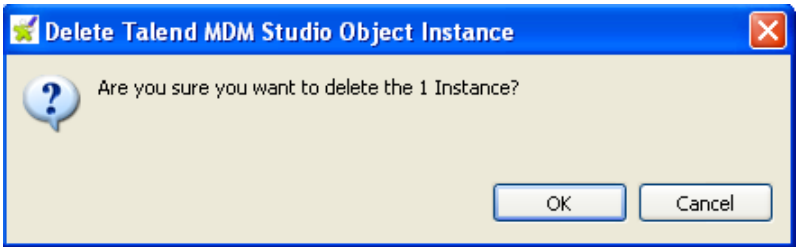
You can delete any of the Processes you create through a simple right-click on the selected item.

Prerequisite(s): You have already connected to the MDM server from *Talend Open Studio for MDM*. At least one Process exists.

To delete a Process:

- In the **MDM Repository** tree view, expand the **Processes** node.
- Right-click the Process you want to delete and select **Delete** from the contextual menu.

  A confirmation dialog box displays prompting you to confirm the deletion operation or to cancel it.



- Click **OK** to close the dialog box.

  The selected Process is deleted from the **MDM Repository** tree view.

3.5.2. Triggers

Triggers are used to link the MDM Hub and the Data Models you define in the Studio. You can use Triggers to call specific Processes under certain conditions.

The Triggers you define in *Talend Open Studio for MDM* enable different services to be undertaken on specific data records.

Then, whenever a business user creates or modifies a data record from *Talend MDM Web User Interface*, for example, a Trigger kicks in and the relevant event-based Process is undertaken by the Studio.

3.5.2.1. Creating a Trigger

The section below explains how to create a Trigger from scratch, however, *Talend Open Studio for MDM* enables you as well to automatically generate Triggers based on the Jobs listed under the **Job Designs** node in the **MDM Repository** tree view. You can then do any modifications in the generated Trigger, if required. For further information, see section *Generating a job-based Trigger*.

Triggers are rules for transforming data. You can easily define Triggers on specific business entities/elements from *Talend Open Studio for MDM*.

Parameters to set when defining Triggers include:

- selecting the business entity you want to Trigger the Process on,
- setting conditions on its content,
- deciding the execution mode,
- selecting the service to Trigger,
- setting the service parameters.

Prerequisite(s): You have already connected to the MDM server from *Talend Open Studio for MDM*.

For this example, a data model called *Product* has been created in *Talend Open Studio for MDM*. This data model has two business entities: *Product* and *ProductFamily*. Several attributes have been created in the *Product* entity including *Price* and *Family*. For further information about how to create a data model, see section *Setting up a data model*.

A Process has been created and named *Call_Job*. One service has been defined in this Process to launch a validation Job whenever a price of an item that belongs to a specific family has been changed through *Talend MDM Web User Interface*. For further information about how to create such a Process, see section *How to create a Process to enrich data on the fly*.

What is left to be done now is to create a Trigger and configure it to launch the *Call a Job from MDM* Process that will by turn launch the **Talend** price validation Job.

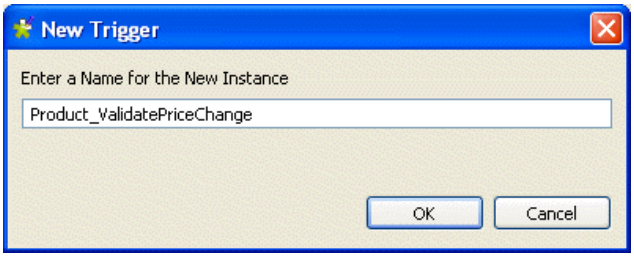
The sub-sections below explain in detail the procedures necessary to complete the creation of such a Trigger.

How to define the business entities on which to trigger a specific Process

To create a Trigger, you must first define the business entities in a selected data model on which you want to trigger a specific Process. You also need to define the execution mode. To do this:

1. In the **MDM Repository** tree view, expand **Event Management** and then right-click **Triggers** and select **New** from the contextual menu.

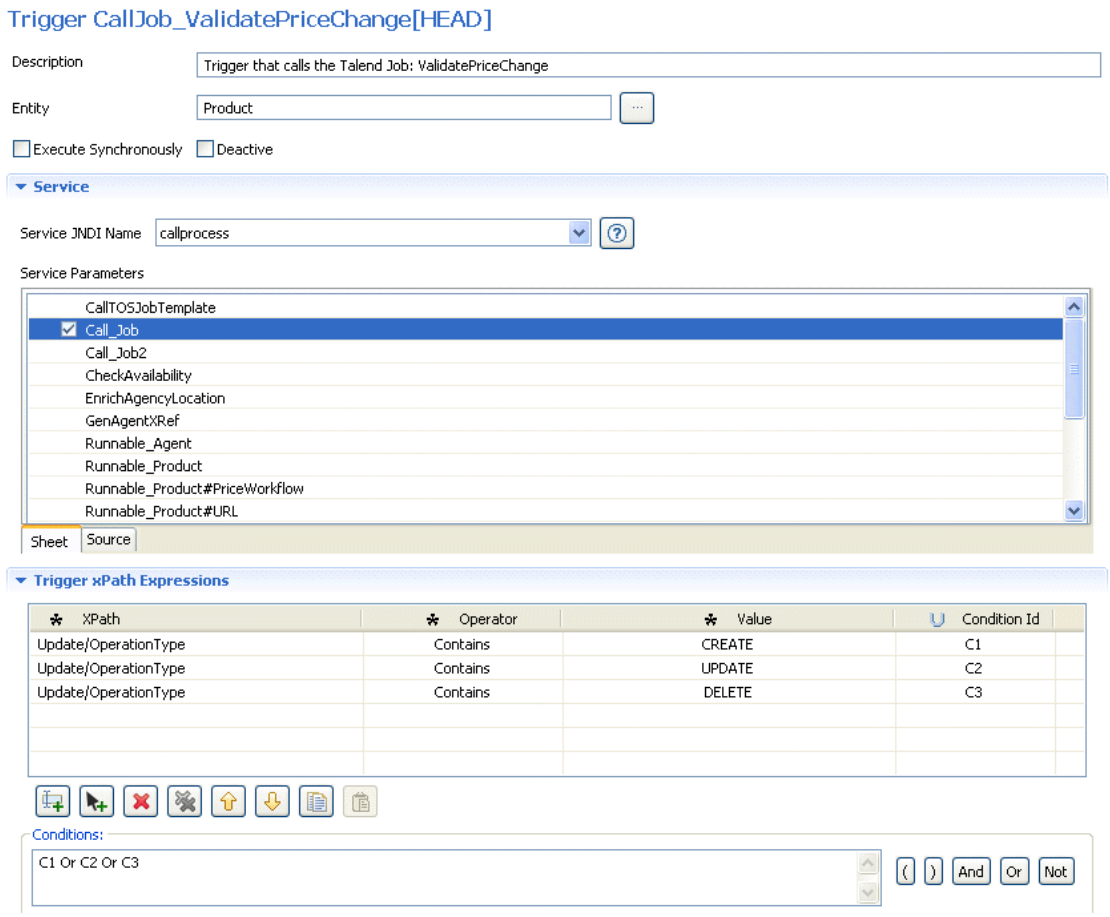
   The **[New Trigger]** dialog box displays.



2. Enter a name for the new Trigger and then click **OK** to close the dialog box.

   An editor for the newly created Trigger opens in the workspace.

   No spaces are allowed in the Trigger namel. The OK button in the dialog box will be unavailable as long as there is a space in the instance name.



3. In the **Description** field, enter a description to identify the Trigger being created.

   In this example, the Trigger will launch a Process that will by turn launch the Job *ValidatePriceChange_0.1*.

4. Click the three-dot button next to the **Entity** field to open a dialog box where you can select the business entity you want to trigger the Process on, *Product* in this example.

5. In the open dialog box, click the **Data Models** arrow and select the data model that holds the business entity you want to run the Trigger on.

6. Select the business entity in the list and click **Add** to close the dialog box.

   The selected business entity name displays in the **Entity** field.

7. Select the check box next to the execution mode you want to use:

Select	
Execute Synchronously	immediately executes the triggered Process. In most cases, this implies that an open connection from a source system will not be released until the data record has been pushed to the destination system and the connection released by the destination system.
Deactivate	To put the defined Trigger on hold.

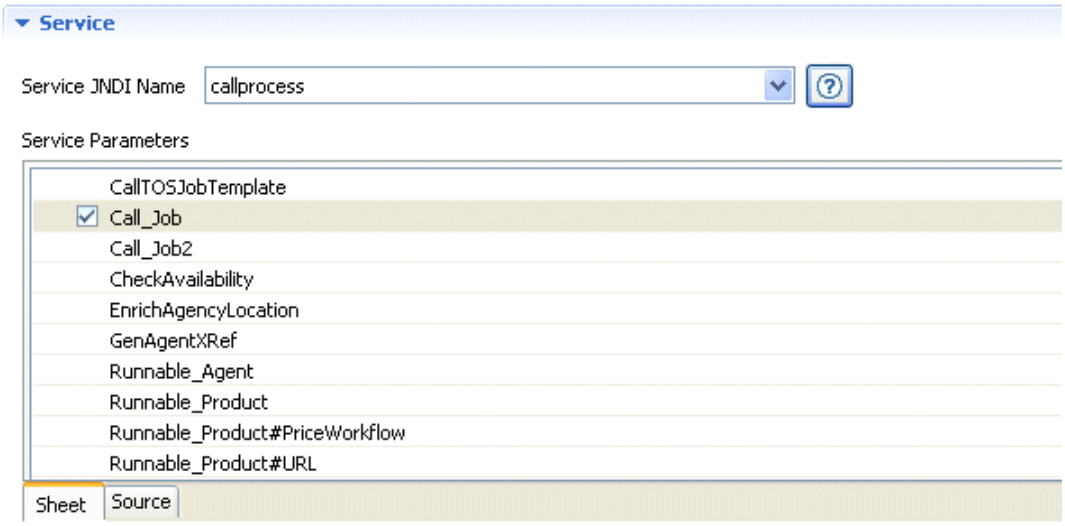
How to select the service to Trigger and set the service parameters

After defining the business entity on which you want to trigger a specific Process, you need to select a service to run on the defined business entity and set the service parameters.

To select a service to trigger and set the parameters, do the following:

1. In the **Service** area, click the **Service JNDI Name** arrow and select from the list the service you want to kick off by the Trigger, *callprocess* in this example.

The services you can find in the list are pre configured MDM services. Click the  button to open a dialog box that gives information about what the service does and its parameters.

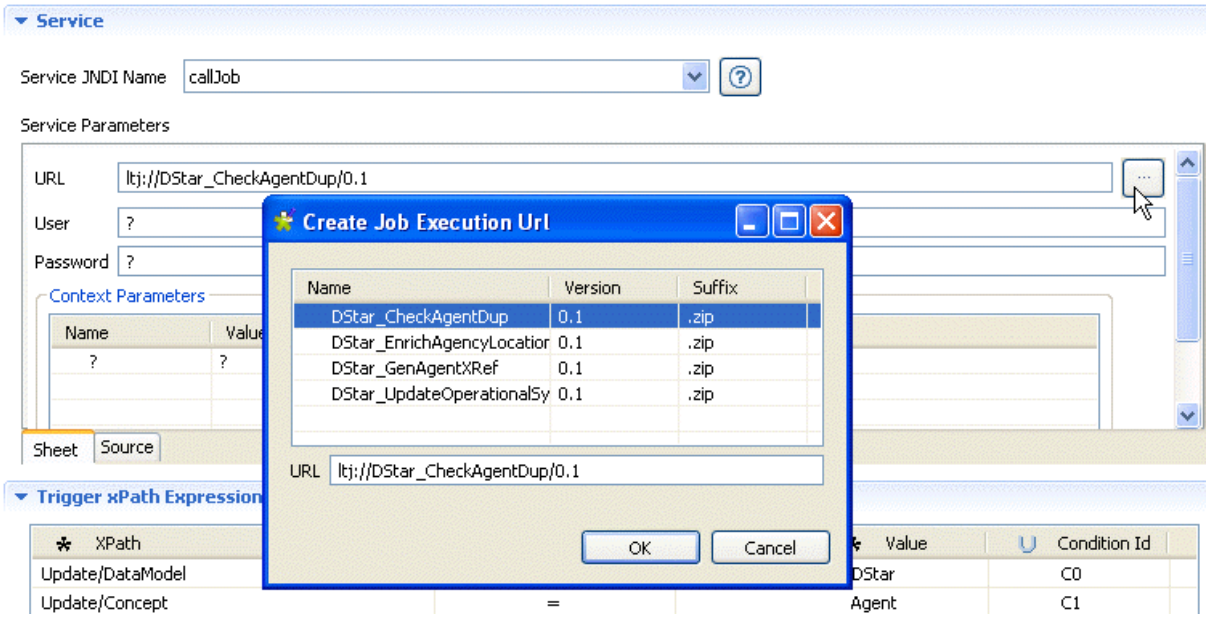


2. In the **Service Parameters** area, define the parameters of the service you select in the list.

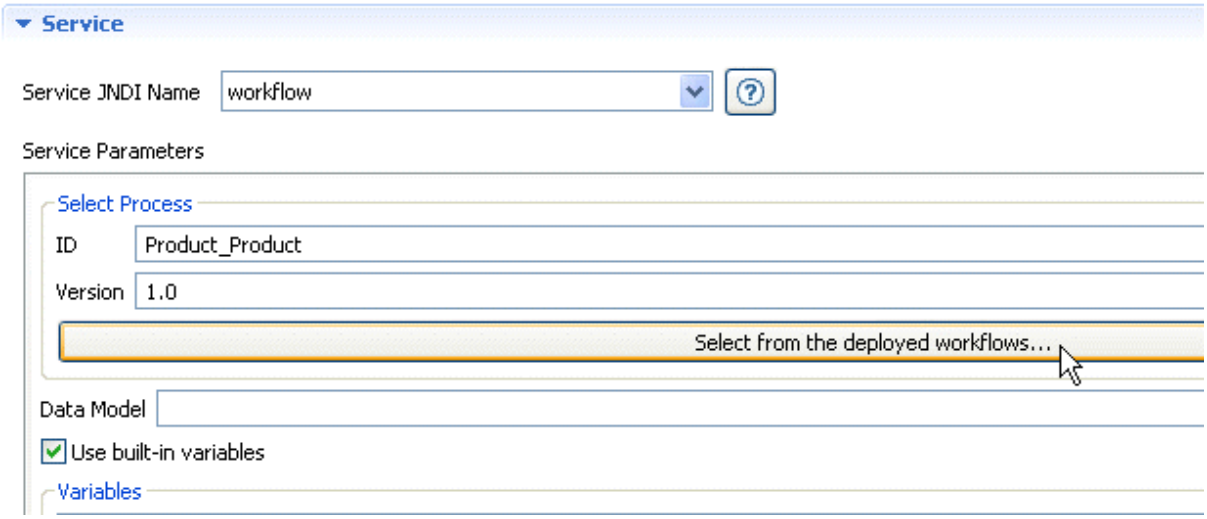
   When you select the *callprocess* service, you just need to click the name of the Process you want to trigger, *Call_Job* in this example.

The *callprocess* service is one of the most important available services on the list. When you select this service, all Processes you already created in *Talend Open Studio for MDM* are listed in the **Service Parameters** area and you can call any of them to transform and cross-reference master data. For more information on Processes, see section *Processes*.

The parameters you need to define in the **Service Parameters** area differ according to the selected service in the service list. *Talend Open Studio for MDM* simplifies this task for you by creating the underlying XML code for the proper execution of the trigger. When you select the *callJob* service, fir example, you can access a list of all deployed Jobs directly from the Trigger editor.



When you select the *workflow* service, for example, you can access a list of all deployed workflows directly from the Trigger editor and this simplifies the process of creating a Trigger.



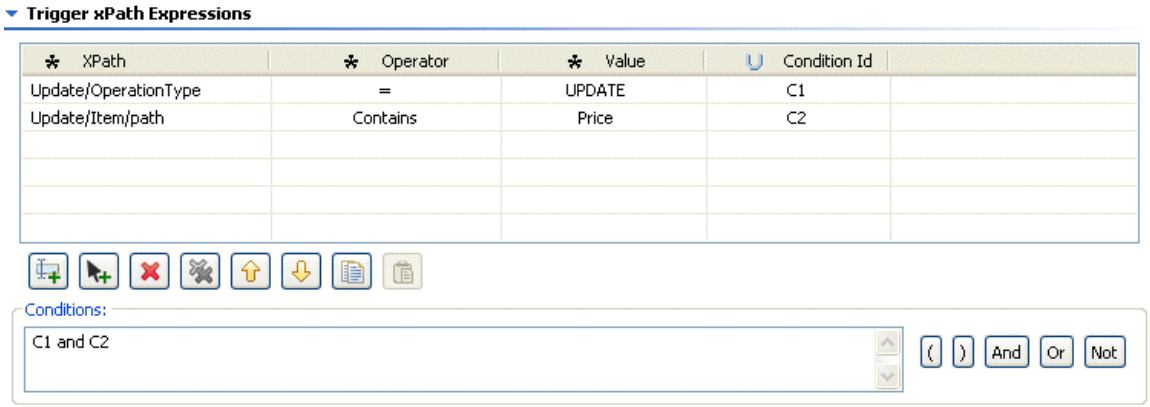
For certain services, you can switch between a graphical editor and a text editor upon clicking the **Sheet** and **Source** tabs respectively. You can define the service parameters in either editors and any modifications you do in one of the two editors will be reflected in the other.

How to set conditions for the Trigger

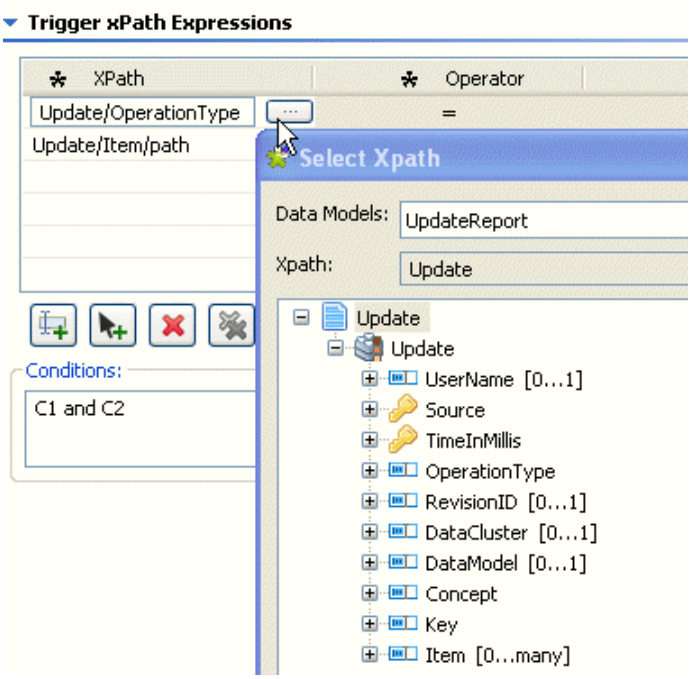
After selecting a pre-configured service to run on the selected business entity and defining the service parameters, you must set conditions on the content of the selected business entity.

To configure the conditions for this Trigger, do the following:

1. In the **Trigger xPath Expressions** area, click the  button to add a new *XPath* line to the table where you can set the first condition.



2. Click in the new line, and then click the three-dot button to open a dialog box where you can select the entity and/or element on which you want to define conditions, *Update/OperationType* in this example.



3. Click in the **Operator** column and select an operator from the list.

4. In the **Value** column, enter a value for each of the selected business entities/elements, the value is usually the name of the entity or element. Enter the *UPDATE* value in this example (case sensitive).

5. In the **Condition Id** column, enter a unique identifier for the condition you want to set on the selected business entity/element, *C1* in this example.

   This first condition means that the Trigger will be executed only on the records in the *Update* entity when there is an update operation.You may want to activate this Trigger only if the update operation is done on the *Price* attribute. So, follow the same steps to define this second condition on the *Item/path* attribute using the *Price* value and *C2* condition.

6. In the **Conditions** area, enter the query you want to undertake on the data record using the condition IDs you set earlier, *C1 and C2* in this example.

   The Trigger created in the above procedure means that you want the Trigger to be executed on a record in the *Update* entity only when the *Price* attribute has been changed.

   When the conditions match the content of the data record, the selected service is triggered on the content of the data records that match the set conditions.

7. Click the save icon on the toolbar or press **Ctrl + S** on your keyboard to save your changes.

This newly created trigger is listed under the **Trigger** node in the **MDM Repository** tree view.

3.5.2.2. Managing Triggers

An authorized user can also import/export, copy/paste and delete created Triggers from *Talend Open Studio for MDM*.

It is also possible to import and share MDM complete projects or only a data model or part of the data model from the community web page, **Talend Exchange**. For further information, see section *Projects/objects on Talend Exchange*.

How to export Triggers

From *Talend Open Studio for MDM* you can export one or multiple Triggers in order to exchange them between:

- two different MDM Servers or Repositories,
- two different Versions from the same/different MDM Servers or Repositories, for example.

The steps to export one or multiple Triggers are similar to those for any other data object in the **MDM Repository** tree view. For detailed information on how to export data containers, see section *How to export data models*.

How to import Triggers

From *Talend Open Studio for MDM*, you can import Triggers into the current MDM Repository that have been created in other MDM Repositories or in different Versions of the current MDM Repository.

The steps to import one or multiple Triggers are similar to those for any other data object in the **MDM Repository** tree view. For detailed information on how to import data containers, see section *How to import data models*.

How to edit a Trigger

You can open a Trigger you have already created to check its settings and/or edit the defined parameters.

Prerequisite(s): You have already connected to the MDM server in *Talend Open Studio for MDM*. At least one Trigger exists.

To edit a Trigger:

1. In the **MDM Repository** tree view, expand the **Trigger** node.
2. Right-click the Trigger you want to edit and select **Edit** from the contextual menu.

   An editor opens on the selected Trigger in the workspace.
3. Modify the Trigger parameters as needed and then click the save icon on the toolbar or press **Ctrl + S** on your keyboard to save your changes.

The selected Trigger is modified accordingly.

If you try to update a Trigger that has been modified by somebody else after you have retrieved it from the database, a warning message displays to warn you that saving your modifications will overwrite the other user's changes.

How to copy/paste a Trigger

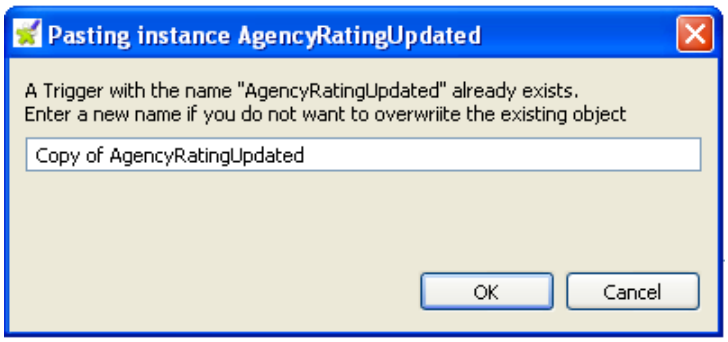
To avoid creating one or multiple Triggers from scratch, you can copy an existing one in the **MDM Repository** tree view and modify its parameters to have a new Trigger.

Prerequisite(s): You have already connected to the MDM server in *Talend Open Studio for MDM*. At least one Trigger exists.

To copy/paste a Trigger, do the following:

1. In the **MDM Repository** tree view, expand the **Trigger** node.
2. Right-click the Trigger you want to copy and select **Copy** from the contextual menu.
3. Right-click the **Trigger** node and select **Paste** from the contextual menu.

A dialog box displays prompting you to enter a name for the new Trigger.



4. Enter a name for the new Trigger and click **OK** to validate the changes and close the dialog box.

   The new Trigger is listed under the **Trigger** node in the **MDM Repository** tree view.

How to duplicate a Trigger

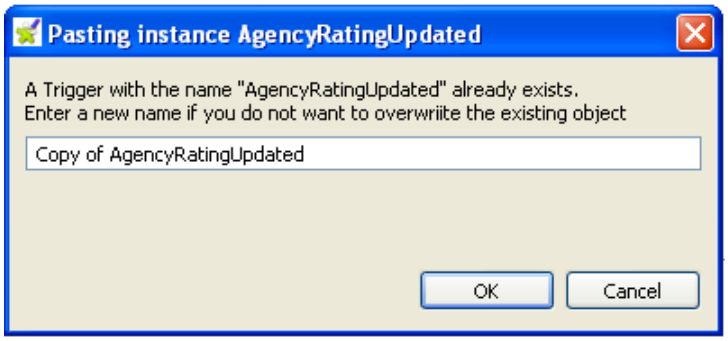
To avoid creating one or multiple Triggers from scratch, you can duplicate an existing one in the **MDM Repository** tree view and modify its parameters to have a new Trigger.

Prerequisite(s): You have already connected to the MDM server in *Talend Open Studio for MDM*. At least one Trigger exists.

To duplicate a Trigger, do the following:

1. In the **MDM Repository** tree view, expand the **Trigger** node.
2. Right-click the Trigger you want to duplicate and select **Duplicate** from the contextual menu.

   A dialog box displays prompting you to enter a name for the new Trigger.



3. Enter a name for the new Trigger and click **OK** to validate the changes and close the dialog box.

   The new Trigger is listed under the **Trigger** node in the **MDM Repository** tree view.

You can also duplicate the data object if you drop it onto its parent node in the **MDM Repository** tree view.

How to delete a Trigger

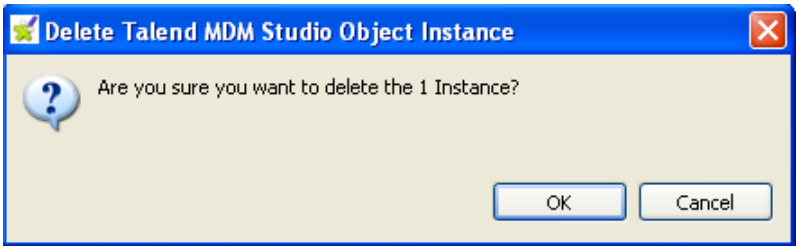
You can delete any of the Triggers you create through a simple right-click on the selected item.

Prerequisite(s): You have already connected to the MDM server in *Talend Open Studio for MDM*. At least one Trigger exists.

To delete a Trigger:

1. In the **MDM Repository** tree view, expand the **Trigger** node.
2. Right-click the Trigger you want to delete and select **Delete** from the contextual menu.

   A confirmation dialog box displays prompting you to confirm the deletion operation or to cancel it.



3. Click **OK** to close the dialog box.

   The selected Trigger is deleted from the **MDM Repository** tree view.

3.6. Job Designs

All data integration Jobs created in the **Integration** perspective of *Talend Open Studio for MDM* are shared with the **MDM** perspective, i.e. they are automatically listed under the **Job Designs** node in the **MDM Repository** tree view. For more information on the different tools integrated in the MDM Studio, see section *A comprehensive set of tools*.

Talend Open Studio for MDM enables you to deploy these data integration Jobs on the MDM server in .war or .zip files. Several possibilities are available for this Job deployment:

- through a manual import/export operation. For further information, see section *Deploying Jobs manually on the MDM server*.
- through automatic deployment from the **Integration** perspective. For further information, see section *How to deploy Jobs from the Integration perspective*.
- through automatic deployment from the **MDM** perspective. For further information, see section *How to deploy Jobs from the MDM perspective*.

The **Job Designs** node in the **MDM Repository** tree view groups all data integration Jobs under the **Source Jobs** folder and all data integration Jobs that have been deployed on the MDM server under the **Deployed Jobs** folder.

You can also run a Job directly from the **MDM** perspective through a right-click on the Job in the **Source Jobs** folder.

3.6.1. Deploying Jobs manually on the MDM server

An authorized user can export/import an archive for a data integration Job from the **Integration** perspective into the **MDM** perspective. A job-based Process or a job-based Trigger can then be automatically generated from the imported archive. For more information, see section *Generating a job-based Process* and section *Generating a job-based Trigger*.

Once the job-based Process is generated, it can be attached to a Trigger which will allow, when kicked off, the data integration operation to be carried out on master data.

For more information on creating and managing Triggers, see section *Triggers*.

3.6.1.1. How to export Job scripts

From the **Integration** perspective in *Talend Open Studio for MDM*, you can export one or more Jobs in order to import them into the **MDM** perspective.

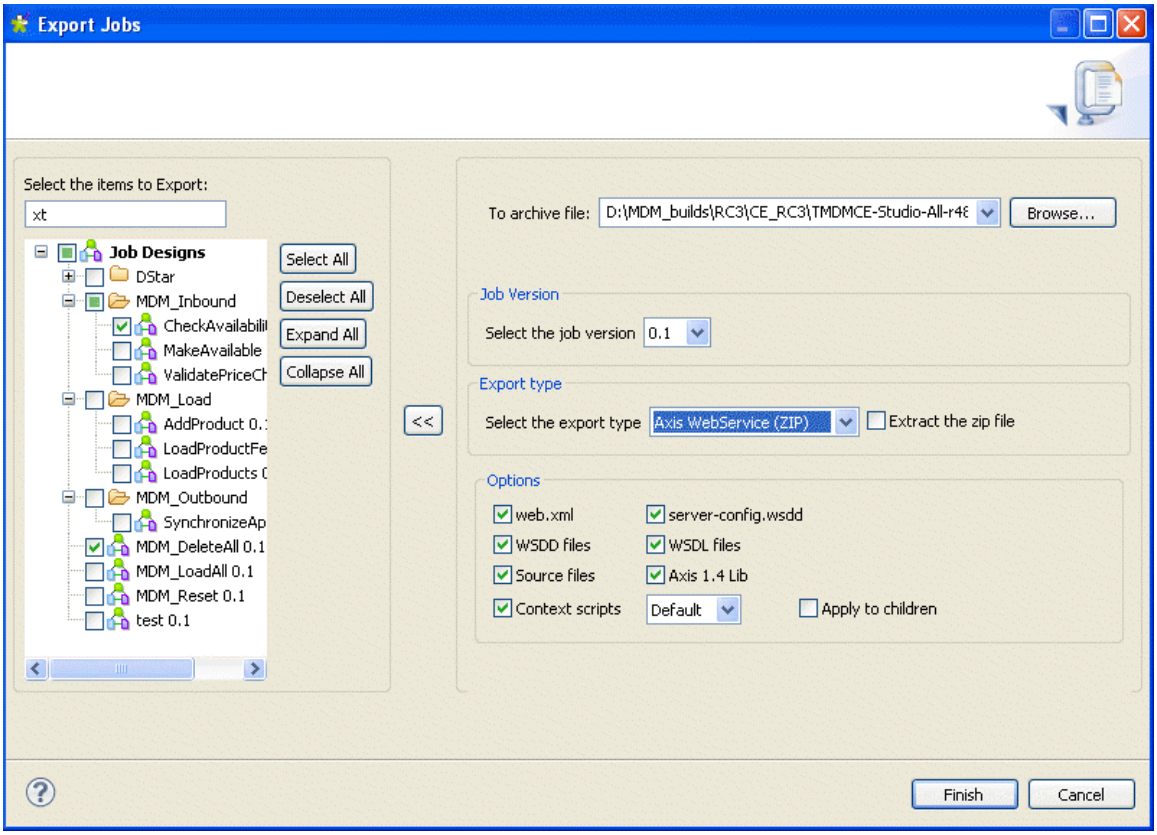
The export Job script feature allows you to deploy and execute the Job on the MDM server. Exporting a Job script adds to an archive all the files required to execute the Job, including the .bat and .sh files along with the possible context-parameter files or relative files.

Prerequisite(s): You have already connected to the MDM server from *Talend Open Studio for MDM*. At least one data integration Job is created in the **Integration** perspective.

To export one or more data integration Jobs:

1. In the **Integration** perspective, expand the **Job Designs** node in the **Repository** tree view and browse to the Job you want to export.
2. Right-click the Job and select **Export Job** from the contextual menu.

   The **[Export Jobs]** dialog box displays.



You can show/hide a tree view of all created Jobs directly from the **[Export Job Scripts]** dialog box by clicking the  and the  buttons respectively. The Jobs you selected earlier in the **Integration** perspective display with selected check boxes. This flexibility allows you to modify the selection of items to be exported directly from the dialog box without having to close it and go back to the **Integration** perspective in *Talend Open Studio for MDM*.

3. In the **To archive file** field, browse to where you want to export the archive file.
4. In the **Job Version** area, select the version number of the Job you want to export if you have created more than one version.
5. From the Select the export type list, select a type for the exported file.
6. Click **Finish** to validate your changes, complete the export operation and close the dialog box.

A file of the defined type holding the job script is created in the defined place.

3.6.1.2. How to import a Job archive

Talend Open Studio for MDM enables you to import Jobs created in the **Integration** perspective into the **MDM** perspective. Once imported, you can automatically generate Processes based on these Jobs. For more information on how to generate a job-related action, see section *Generating a job-based Process*.

Prerequisite(s): You have already connected to the MDM server from *Talend Open Studio for MDM*. You have exported one or more Jobs in .war files from the **Integration** perspective.

To import one or more Jobs into the **MDM** perspective, do the following:

1. In the **MDM Repository** tree view, right-click the **Job Designs** node and select **Import Items** from the contextual menu.

   The **[Import Repository Items]** dialog box displays.

2. In the dialog box, select the file including the exported Job.

3. Click **Finish** to import the Job and close the dialog box.

   The imported Job displays under the **Job Designs** node in the **MDM Repository** tree view.

3.6.2. Deploying Jobs automatically on the MDM server

Talend Open Studio for MDM provides additional deployment capabilities that enables an authorized user to deploy a Job from the **Integration** or the **MDM** perspectives onto the MDM server directly without the need of the import/export operation.

A job-based Process or a job-based Trigger can then be automatically generated from the deployed Job. For more information, see section *Generating a job-based Process* and section *Generating a job-based Trigger*.

Once the job-based Process is generated, it can be attached to a Trigger which will allow, when kicked off, the data integration operation to be carried out on master data.

For more information on creating and managing Triggers, see section *Triggers*.

3.6.2.1. How to deploy Jobs from the Integration perspective

You can deploy a Job from the **Integration** perspective onto the MDM server directly without the need of the import/export operation. The Jobs will be deployed on the MDM server to which you connect from within the MDM Studio. For further information, see section *Launching Talend Open Studio for MDM and connecting to the MDM server*.

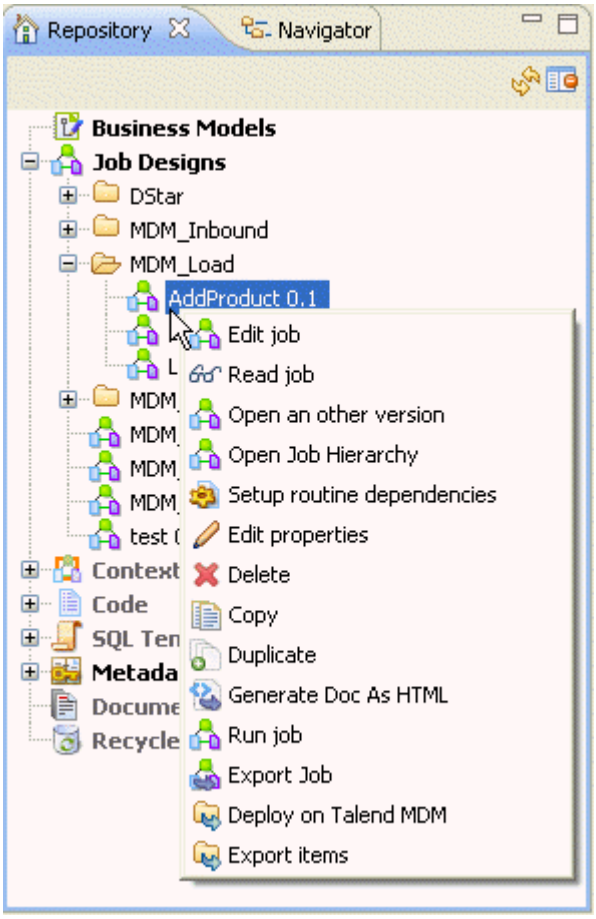
As the **Job Designs** node shows up directly within the **MDM Repository** tree view, you can also deploy Jobs on the MDM server from the **MDM** perspective. For further information, see section *How to deploy Jobs from the MDM perspective*.

Prerequisite(s): You have already connected to an MDM server from *Talend Open Studio for MDM*. The server on which you want to deploy the Job is up and running.

To deploy a Job on an MDM server, do the following:

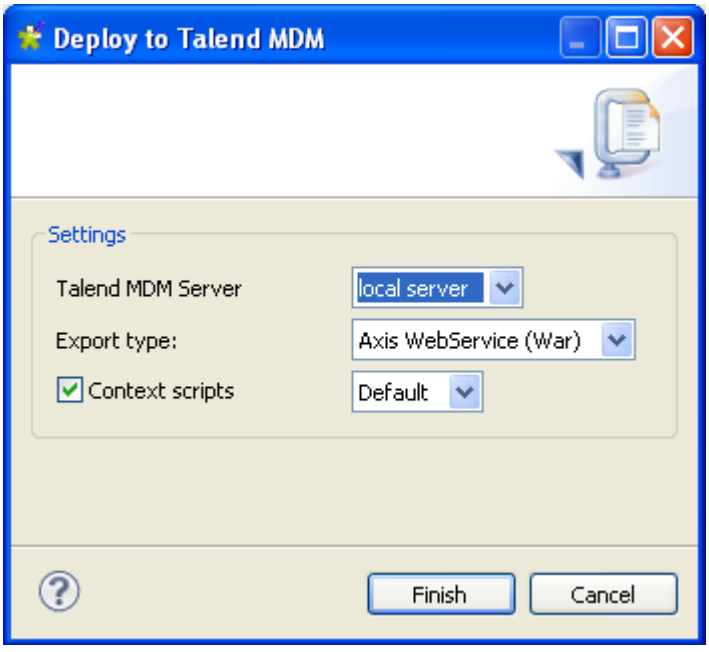
1. From the **Integration** perspective and in the **Repository** tree view, expand the **Job Designs** node and then right-click the Job you want to deploy.

If you want to deploy multiple Jobs at the same time, press and hold down the **Ctrl** key while you click each of the Jobs you want to deploy.



2. From the contextual menu, select **Deploy to Talend MDM**.

   The **[Deploy to Talend MDM]** dialog box displays.



3. From **Talend MDM Server** list, select the MDM server on which you want to deploy the Job.

4. From the **Export type** list, select the export type, and then click **Finish**.

   A confirmation message displays.

5. Click **OK** to close the message and the dialog box.

3.6.2.2. How to deploy Jobs from the MDM perspective

As data integration Jobs are shared between the **Integration** and the **MDM** perspectives, you can deploy any of these Jobs on the MDM server directly from the **MDM** perspective.

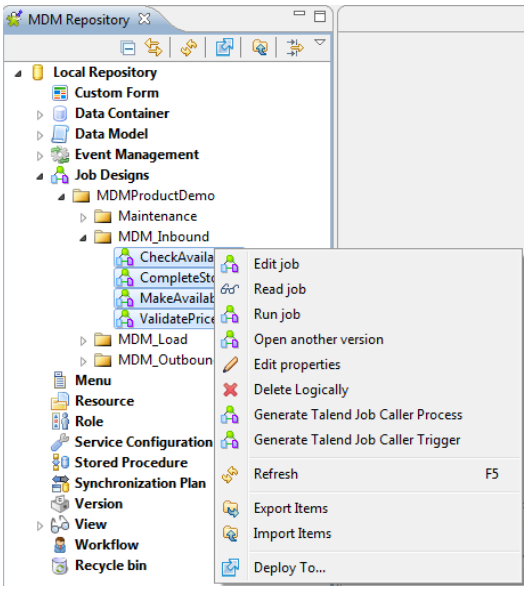
Prerequisite(s): You have already connected to the MDM server from *Talend Open Studio for MDM*. At least one data integration Job has been created in the **Integration** perspective.

To deploy a Job on a specific MDM server, do the following:

1. From the **MDM** perspective and in the **MDM Repository** tree view, expand the **Job Designs** node and then right-click the Job you want to deploy.

   If you want to deploy multiple Jobs at the same time, press and hold down the **Ctrl** key while you click each of the Jobs you want to deploy.



2. From the contextual menu, select **Deploy To...**.

   The **[Deploy to Talend MDM]** dialog box displays.

3. If appropriate, select the server where you want to deploy the objects in the **[Select server location definition]** window.

4. In the **[Deploy to Talend MDM]** window, specify the Settings for the deployment:

   - Export type: Choose between Distributed (War) or Hosted (Zip).

     Deploy the Job as a Zip file if you want it to be preloaded and run directly in the MDM Server, which reduces latency. Deploy the Job as a War file if you want to embed it in a web service, for instance as a way of deploying the Jobs across different servers for the purposes of load balancing.

   - Leave the Context scripts checkbox selected.

5. Click **Finish** to deploy your objects.

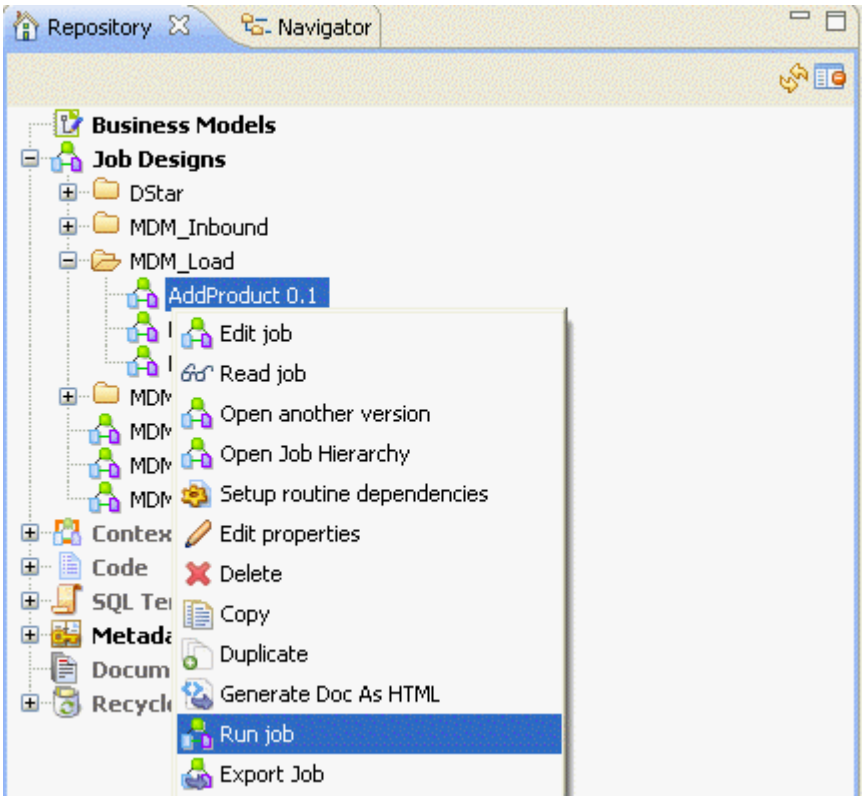
3.6.3. Running Jobs

You can run a Job in *Talend Open Studio for MDM* either from the **Integration** perspective or directly from the **MDM** perspective.

Prerequisite(s): You have already connected to the MDM server from *Talend Open Studio for MDM*. At least one data integration Job has been created in the **Integration** perspective and deployed on the MDM server.

To run a Job from the **Integration** perspective, do the following:

1. From the **Integration** perspective and in the **Repository** tree view, expand the **Job Designs** node and then right-click the Job you want to run.

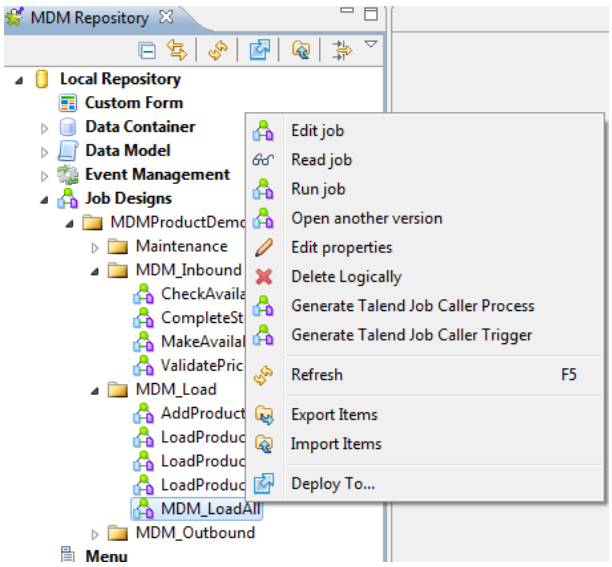


2. From the contextual menu, select **Run job**.

   The selected Job is executed and the execution results are shown in the **Run** console.

To run a Job from the **MDM** perspective, do the following:

1. From the **MDM** perspective and in the **MDM Repository** tree view, expand the **Job Designs** node and then right-click the Job you want to run.



2. From the contextual menu, select **Run job**.

   The selected Job is executed and the execution results are shown in the **Run** console.

3.6.4. Generating a job-based Process

Talend Open Studio for MDM enables you to automatically generate Processes based on the deployed Jobs listed under the **Job Designs** node in the **MDM Repository** tree view.

Once the job-based Process is generated, you can attach it to a Trigger which will initiate, when kicked off, the data integration process defined in the Job.

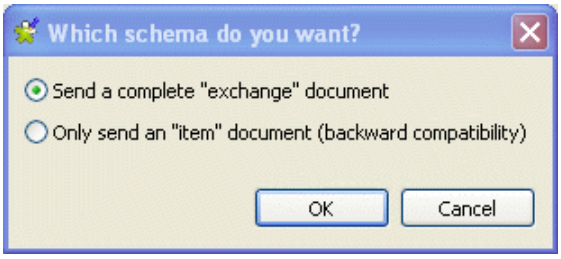
For more information on creating and managing Processes, see section *Processes*.

Prerequisite(s): You have already connected to the MDM server from *Talend Open Studio for MDM*. At least one data integration Job is created in the **Integration** perspective and deployed on the MDM server.

To generate a Job-based Process:

1. In the **MDM Repository** tree view, expand the **Job Designs** node.
2. Right-click the Job on which you want to generate a Process and then select **Generate Talend Job Caller Process**.

   A dialog box displays.

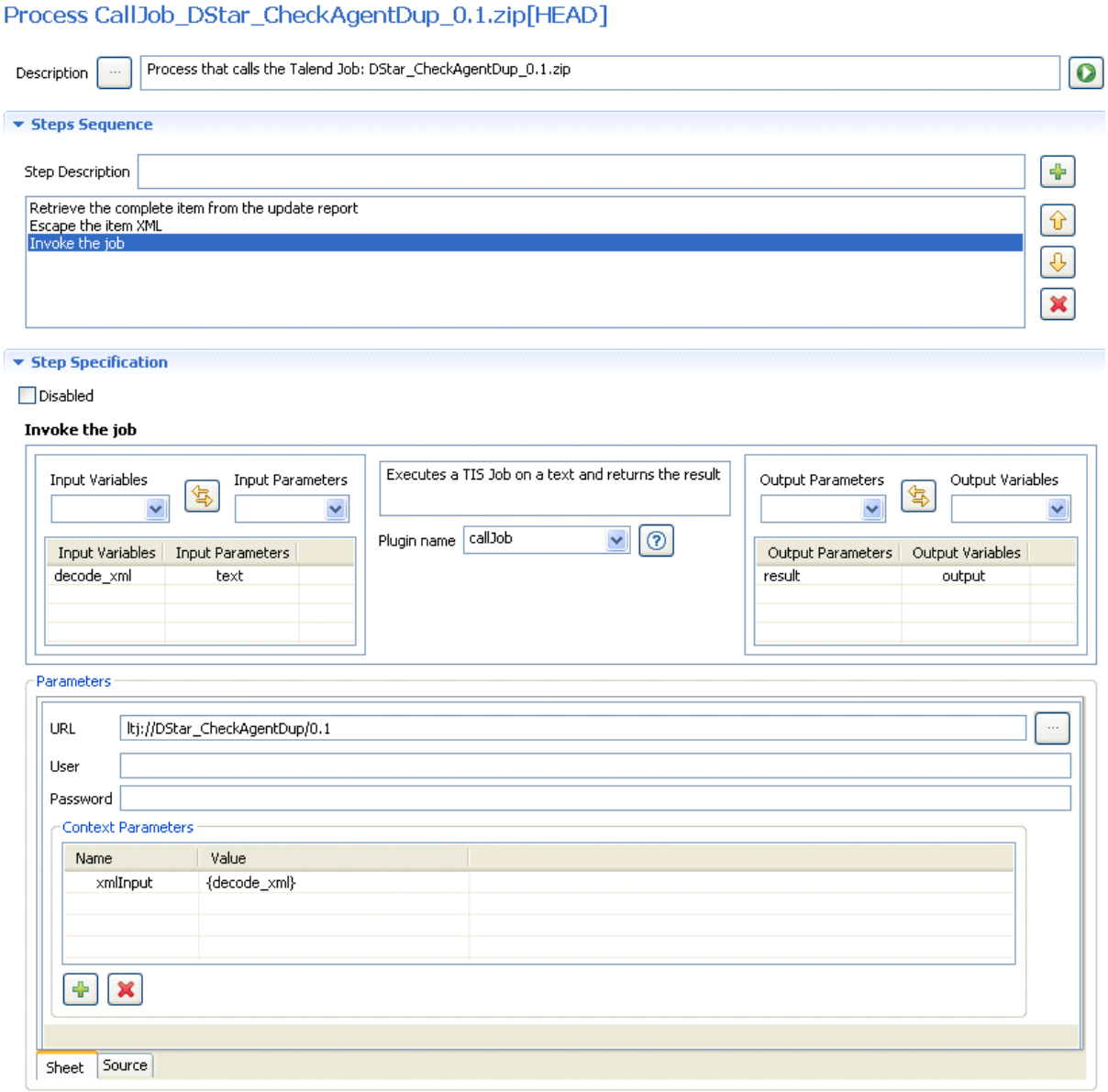


The option selected by default is to send a complete "exchange" document to the MDM server. This document will have the Update Report in addition to the item itself. However, if you want your Process to be compatible with older MDM versions (backward compatibility), you must select the Only send an "item" document option.

3. Click **OK** to validate your schema choice and close the dialog box.

The generated Process holding the name of the generated Job displays under **Event Management - Process** in the **MDM Repository** tree view.

4. If required, double-click the generated job-based Process to open an editor in the workspace that lists the Process parameters.



In the **Parameters** area, you can always switch between two views: **Sheet** and **Source**. The first enables you to add/modify parameters using fields and rows in a table, while the second display the parameters in its XML format where you can add/modify the parameters directly in the XML text.

The generated job-based process includes one basic step to call the Job and two additional steps as the following:

First step: to retrieve the complete XML record through the *XSLT* plug-in.

Second step: to decode XML in order to remove the escape function you used in the XSLT document and thus send a real XML document to the Job.

Final step: to send the XML document to a **Talend** Job.

Usually, a Process can have only the step to call the Job, the *calljob* plug-in, if the complete XML record is not needed in the task you want to accomplish. For further information, see section *How to create a Process to enrich data on the fly*.

From this editor, you can modify any of the Process parameters, if required.

3.6.5. Generating a job-based Trigger

Talend Open Studio for MDM enables you to automatically generate Triggers based on the Jobs listed under the **Job Designs** node in the **MDM Repository** tree view.

Once the Job-based Trigger is generated, it will initiate, when kicked off, the data integration Process defined in the Job.

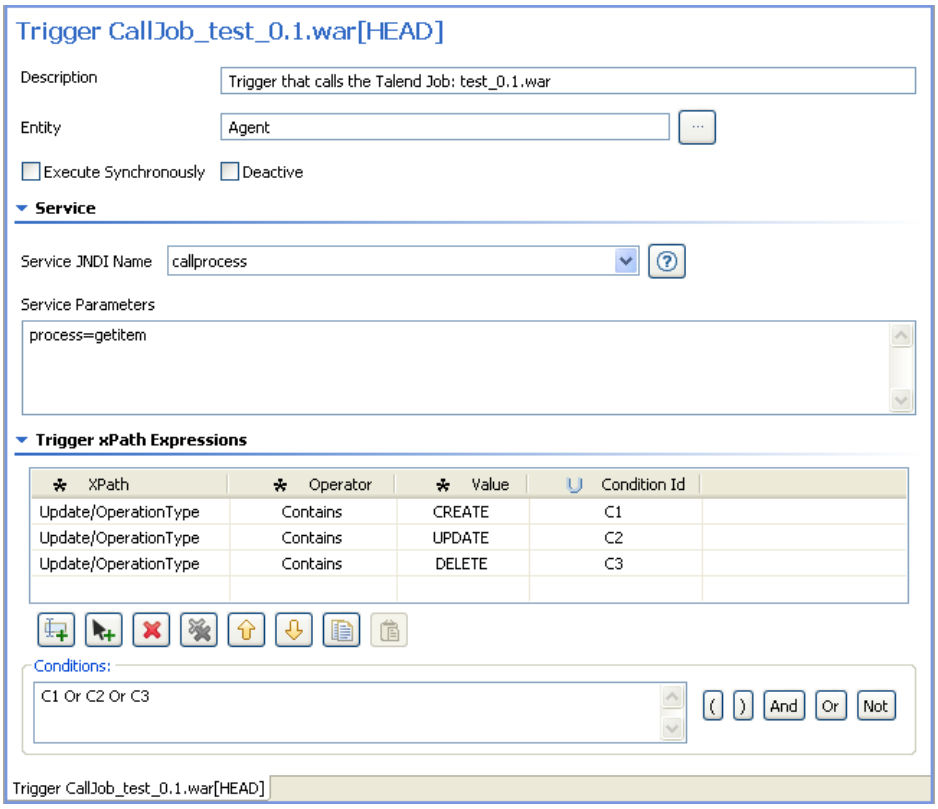
For more information on creating and managing Triggers, see section *Triggers*.

Prerequisite(s): You have already connected to the MDM server from *Talend Open Studio for MDM*. At least one Job is created in the **Integration** perspective.

To generate a Job-based Trigger:

1. In the **MDM Repository** tree view, expand the **Job Designs** node.
2. Right-click the Job on which you want to generate a Trigger and then select **Generate Talend Job Caller Trigger**.

   The generated Trigger holding the name of the generated Job displays under **Event Management - Trigger** in the **MDM Repository** tree view.
3. If required, double-click the generated job-based Trigger to open an editor in the workspace that lists the Trigger parameters.



From this editor, you can modify any of the Trigger parameters, if required.

Chapter 4. Advanced subjects

This chapter provides the information you need to carry out some advanced MDM procedures including using XQuery instructions on master data and importing into your current Studio MDM projects/objects created by **Talend** community and uploaded on **Talend Exchange**.

Before starting any of these MDM management procedures, you need to be familiar with the *Talend Open Studio for MDM* Graphical User Interface (GUI). For more information, see chapter *Talend MDM: Concepts & Principles* and appendix *Talend Open Studio for MDM management GUI*.

4.1. Stored Procedures

The stored procedure feature in *Talend Open Studio for MDM* allows authorized users to create and store XQuery instructions. These XQuery instructions support input parameters and are used to query master data grouped in the MDM Hub.

The input parameters of the XQuery instructions are defined as a % sign followed by a number. For example: *%0, %1, %2,%n*.

A stored procedure can be executed directly from *Talend Open Studio for MDM* or attached to a specific report in the Web User Interface. For more information, see *Talend MDM Web User Interface User Guide*.

You need to be comfortable with the XQuery language to use stored procedures.

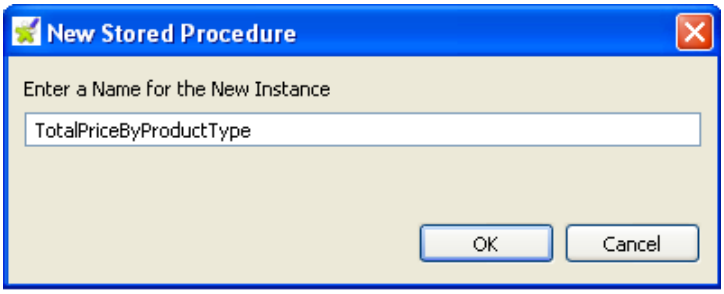
4.1.1. Creating a stored procedure

Prerequisite(s): You have already connected to the MDM server from *Talend Open Studio for MDM*. You have the appropriate user authorization to create a stored procedure.

To create a stored procedure, do the following:

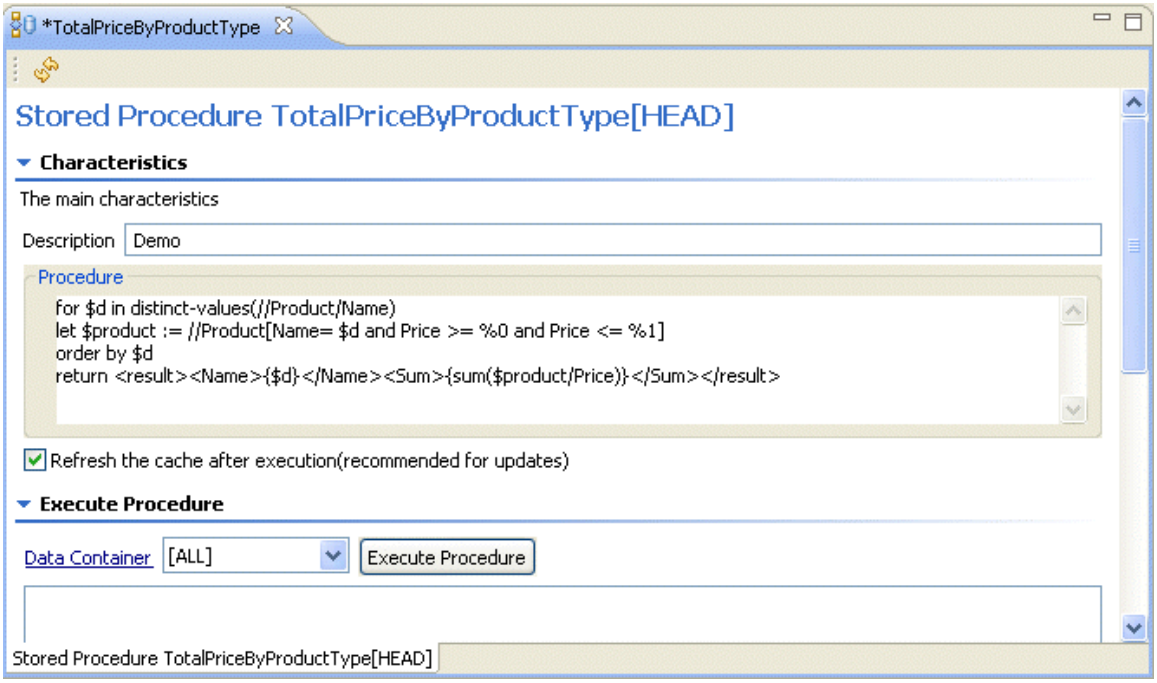
1. In the **MDM Repository** tree view, right-click **Stored Procedure** and select **New** from the contextual menu.

   The **[New Stored Procedure]** dialog box displays.



2. Enter a name for the new stored procedure and then click **OK** to close the dialog box.

3. In the **MDM Repository** tree view, expand the **Stored Procedure** node and click the new stored procedure you created to open it in the workspace.



4. In the **Description** field, enter a description for the stored procedure you want to create.

5. In the **Procedure** area, enter the XQuery syntax to address the data records pertained in a specific entity.

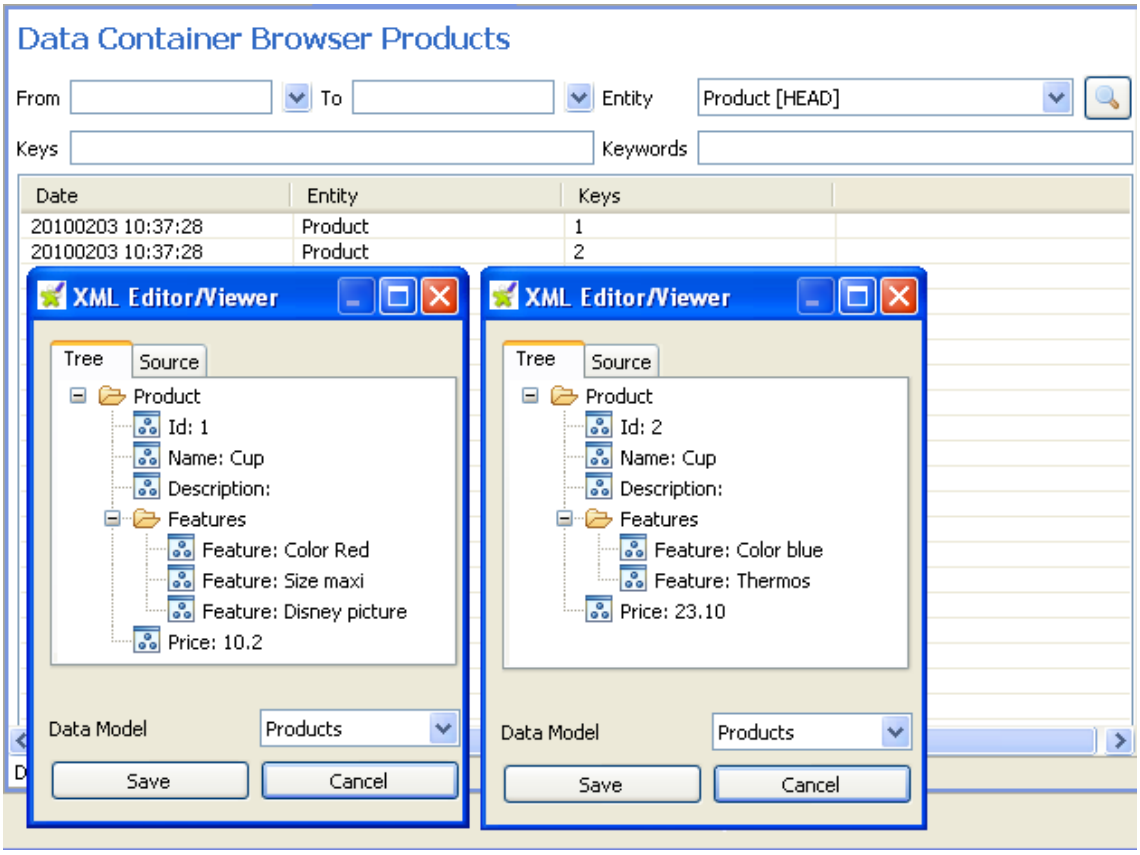
   For more information on the XQuery language, see http://exist.sourceforge.net/xquery.html.

   The stored procedure must have certain syntax if it is to be used in generating summary reports on data stored in the MDM Hub through the Web User Interface. Each result must have the following syntax: <result><Field1>val1</Field1>...<Fieldn>valn</Fieldn></result>. The "Field#" are the fields defined in the report page in the Web User Interface. For more information, see Talend MDM Web User Interface User Guide.

6. Select the **Refresh the cache after execution** check box if you want to refresh the cache after the execution of the Xquery of the stored procedure.

   This is necessary as a stored procedure may perform an insert/update/replace Xquery that is executed directly at the database level. Because of the cache, the change may not be reflected.

   The sample XQuery code that shows in the above captures calculates the sum of the prices of all the records, two cups in this example, listed under the *Product* entity.



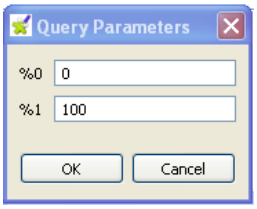
7. Click the save icon on the toolbar or press **Ctrl + S** on your keyboard to save the procedure.

To execute the stored procedure, do the following:

1. In the **Execute Procedure** section of the stored procedure editor, select the data container holding the data records on which you want to execute the query.

2. Click **Execute Procedure**.

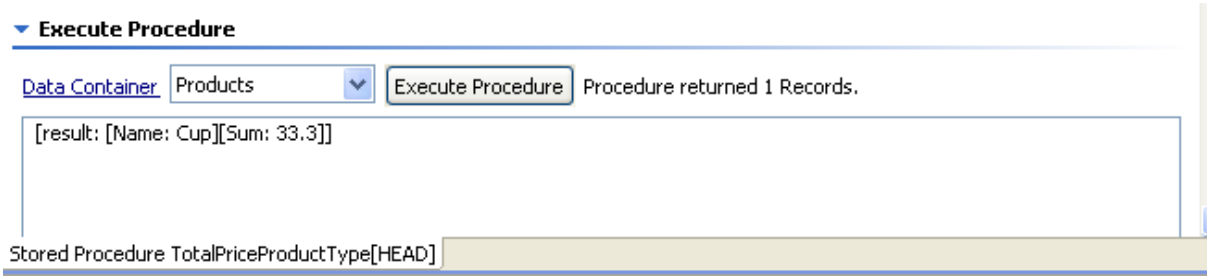
   A dialog box opens prompting you to set the query parameters.



3. Set the minimum and maximum parameters in the corresponding fields. Here we want to calculate cup prices that are between 0 and 100.

4. Click **OK**.

   The result of the operation carried out by the XQuery on the defined data records displays in the lower half of the stored procedure editor.



In this example, the XQuery code used on the *Product* entity calculates the sum of the prices of the listed cups.

4.1.2. Managing stored procedures

An authorized user can also import/export, copy/paste, duplicate and delete stored procedures from *Talend Open Studio for MDM*.

4.1.2.1. How to export stored procedures

From *Talend Open Studio for MDM* you can export one or more stored procedures in order to exchange them between:

- two different MDM Servers or Repositories,
- two different Versions from the same/different MDM Servers or Repositories, for example.

The steps to export one or multiple stored procedures are similar to those for any other data object in the **MDM Repository** tree view. For detailed information on how to export stored procedures, see section *How to export data models*.

4.1.2.2. How to import stored procedures

From *Talend Open Studio for MDM*, you can import stored procedures created on other MDM servers, in different Versions on the same MDM server, or in different MDM Repositories into the current MDM Repository.

The steps to import one or multiple stored procedures are similar to those for any other data object in the **MDM Repository** tree view. For detailed information on how to import stored procedures, see section *How to import data models*.

4.1.2.3. How to edit a stored procedure

You can open a stored procedure you have already created to check its settings and/or edit the defined parameters.

Prerequisite(s): You have already connected to the MDM server from *Talend Open Studio for MDM*. At least one stored procedure exists.

To edit a stored procedure, do the following:

1. In the **MDM Repository** tree view, expand the **Stored Procedure** node.

2. Double-click the stored procedure you want to edit and select **Edit** from the contextual menu.

   An editor opens on the selected stored procedure in the workspace.

3. Modify the stored procedure settings as required and then click the save icon on the toolbar or press **Ctrl + S** on your keyboard to save your changes.

The selected stored procedure is modified accordingly.

4.1.2.4. How to copy/paste a stored procedure

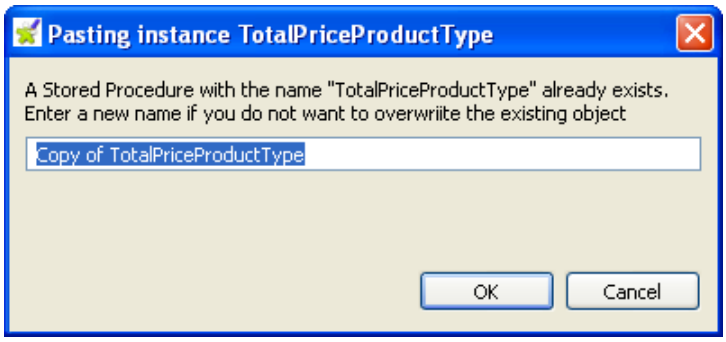
To avoid creating one or multiple stored procedures from scratch, you can copy an existing one in the **MDM Repository** tree view and modify its settings to have a new procedure.

Prerequisite(s): You have already connected to the MDM server from *Talend Open Studio for MDM*. At least one stored procedure exists.

To copy/paste a stored procedure, do the following:

1. In the **MDM Repository** tree view, expand the **Stored Procedure** node.
2. Right-click the procedure you want to duplicate and select **Copy** from the contextual menu.
3. Right-click the **Stored Procedure** node and select **Paste** from the contextual menu.

   A dialog box displays prompting you to enter a name for the new stored procedure.



4. Enter a name for the new stored procedure and click **OK** to validate the changes and close the dialog box.

   The new stored procedure is listed under the **Stored Procedure** node in the **MDM Repository** tree view.

4.1.2.5. How to duplicate a stored procedure

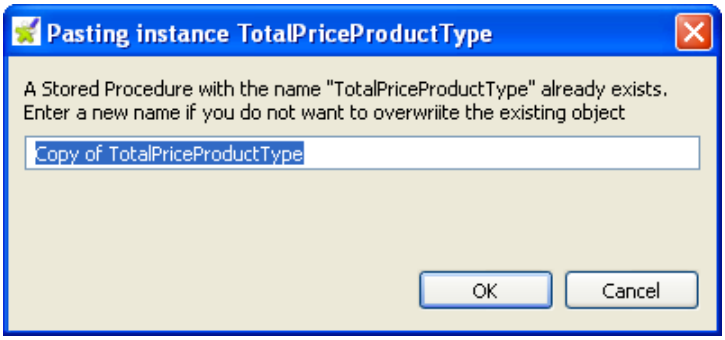
To avoid creating one or multiple stored procedures from scratch, you can duplicate an existing one in the **MDM Repository** tree view and modify its settings to have a new procedure.

Prerequisite(s): You have already connected to the MDM server from *Talend Open Studio for MDM*. At least one stored procedure exists.

To duplicate a stored procedure, do the following:

1. In the **MDM Repository** tree view, expand the **Stored Procedure** node.
2. Right-click the procedure you want to duplicate and select **Duplicate** from the contextual menu.

   A dialog box displays prompting you to enter a name for the new stored procedure.



3. Enter a name for the new stored procedure and click **OK** to validate the changes and close the dialog box.

   The new stored procedure is listed under the **Stored Procedure** node in the **MDM Repository** tree view.

You can also duplicate the stored procedure if you drop it onto its parent node in the **MDM Repository** tree view.

4.1.2.6. How to delete a stored procedure

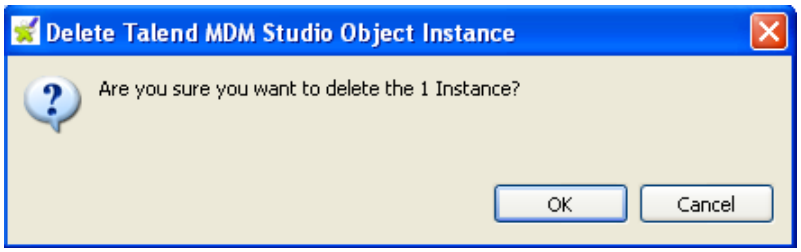
You can delete any of the stored procedures you create through a simple right-click on the selected item.

Prerequisite(s): You have already connected to the MDM server from *Talend Open Studio for MDM*. At least one stored procedure exists.

To delete a stored procedure, do the following:

1. In the **MDM Repository** tree view, expand the **Stored Procedure** node.
2. Right-click the procedure you want to delete and select **Delete** from the contextual menu.

   A confirmation dialog box displays prompting you to confirm the deletion operation or to cancel it.



3. Click **OK** to close the dialog box.

   The selected stored procedure is deleted from the **MDM Repository** tree view.

4.2. Projects/objects on Talend Exchange

Talend Open Studio for MDM enables you to import MDM projects/objects created by **Talend** community and uploaded on **Talend Exchange** into your current Studio.

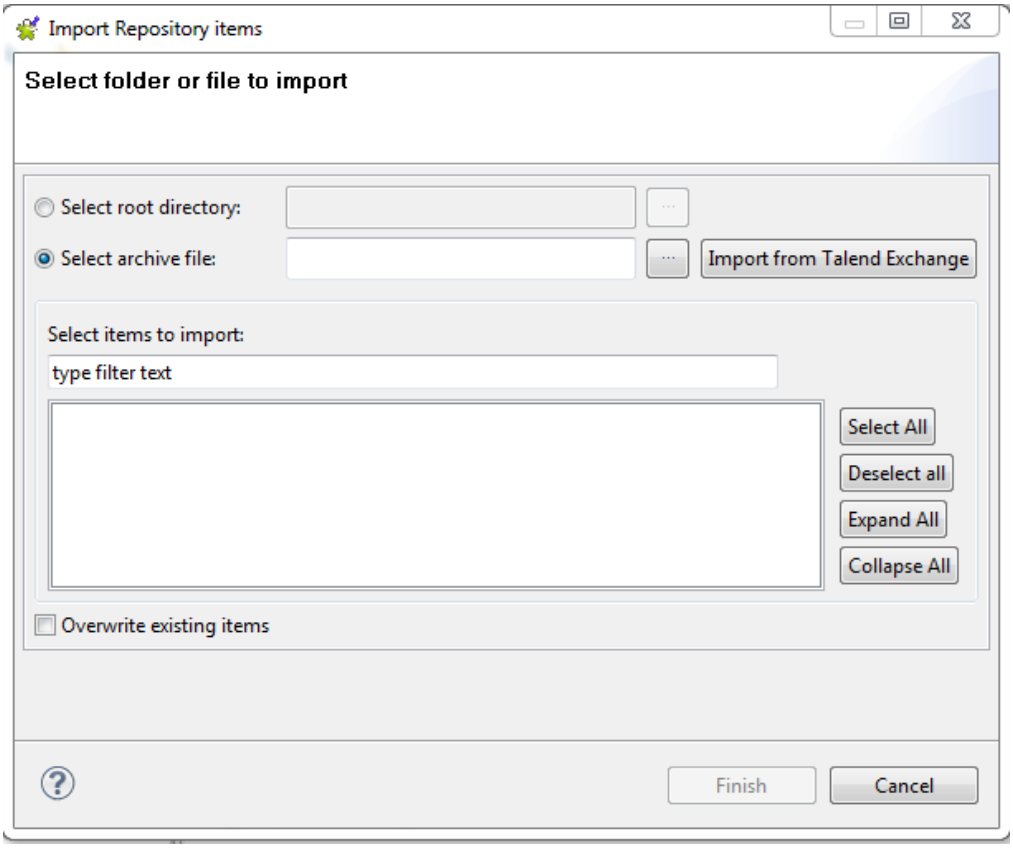
4.2.1. Importing data projects from Talend Exchange

From *Talend Open Studio for MDM* you can share **Talend** community MDM projects by importing them from **Talend Exchange** into your local MDM repository.

To import one or multiple MDM projects from **Talend Exchange** into your local MDM repository, do the following:

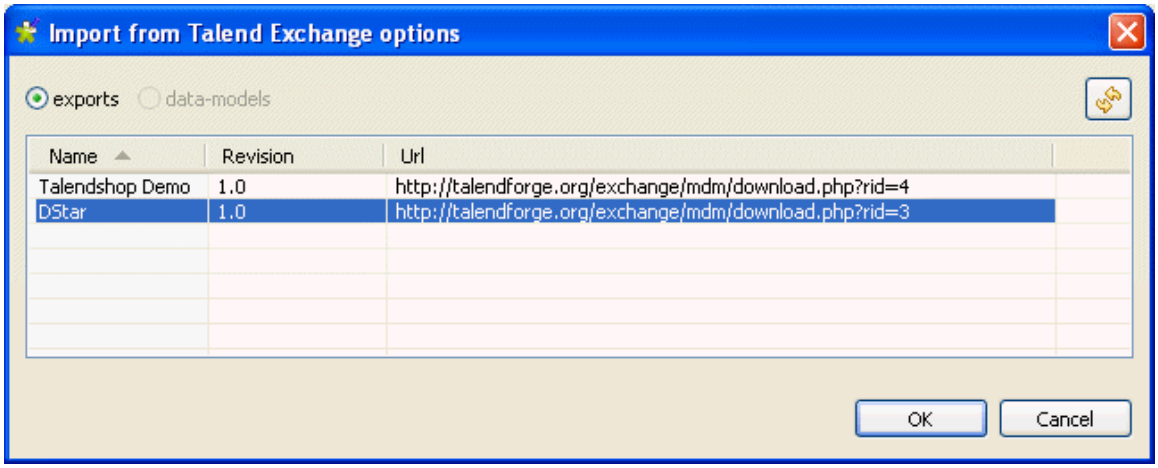
1. In the **MDM Repository** tree view, right-click any of the data objects in the view and then select **Import Items** from the contextual menu.

   The **[Import Repository items]** dialog box displays.



2. Select the **Select archive file** option and then click **Import from Talend Exchange**.

   A dialog box displays and lists all the community MDM projects uploaded on **Talend Exchange**.



3. Select from the list the MDM project you want to import into your current MDM Studio and then click **OK**.

   All MDM data objects of the selected project are shown in the **[Import Repository items]** dialog box and selected by default.

4. Click **Finish** to validate the import operation and close the dialog box.

All data objects in the selected MDM project are exported and listed under the corresponding nodes in the **MDM Repository** tree view of your Studio.

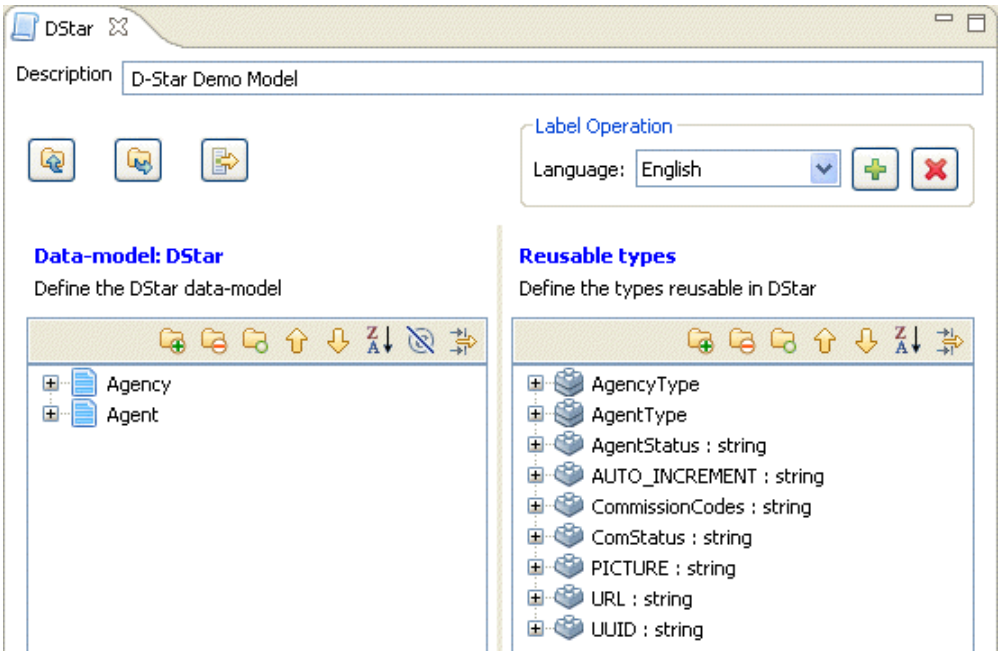
4.2.2. Importing the xsd schema for a specific data model from Talend Exchange

From *Talend Open Studio for MDM* you can share **Talend** community MDM data object (data model) or part of the data object (simple or complex types that can be reused in a data model) through importing it from **Talend Exchange** into your local MDM repository.

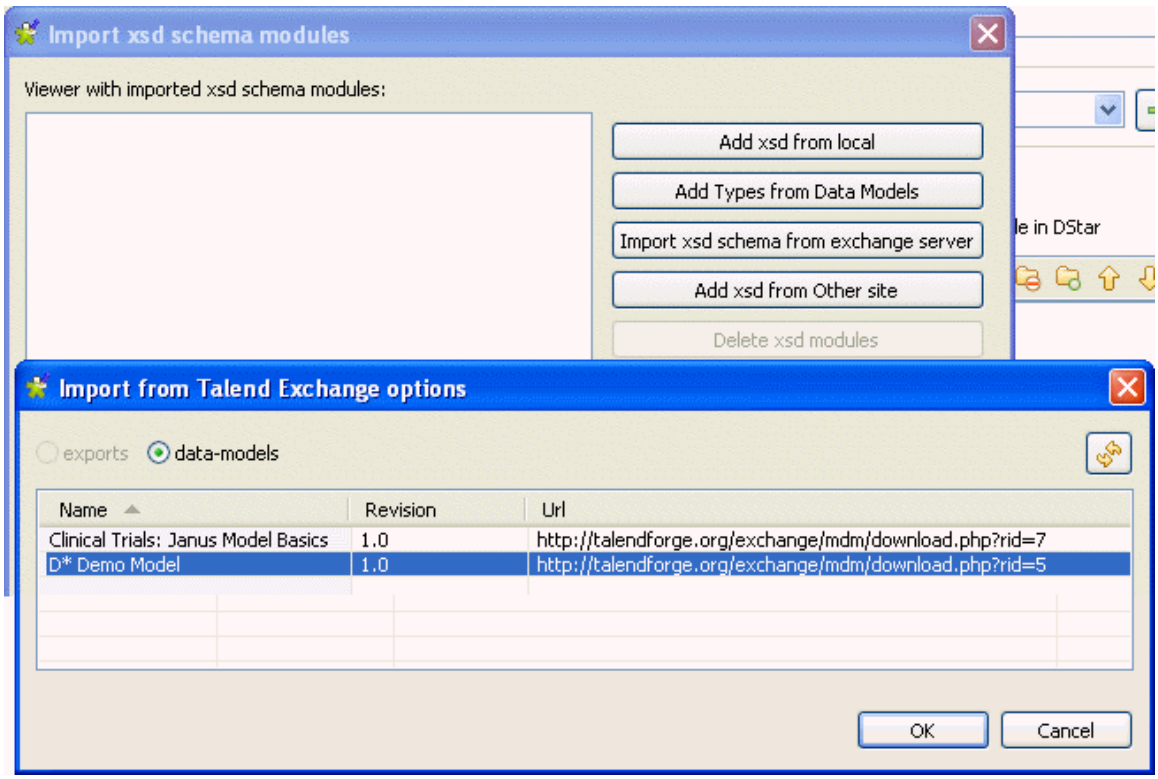
To import a data object or part of the data object from **Talend Exchange** into your local MDM repository, do the following:

1. In the **MDM Repository** tree view, expand **Data Model** and then double-click any of the data models in the list.

   The data model editor displays.



2. Click  to display the **[Import xsd schema module]** dialog box.



3. Click **Import xsd schema from exchange server**.

   The **[Import from Talend Exchange modules]** dialog box displays listing all data models uploaded on **Talend Exchange**.

4. Select from the list the data model(s) you want to import into your current MDM Repository and then click **OK**.

   The imported xsd schema module(s) are listed in the **[Import from Talend Exchange modules]** dialog box.

   If required, double-click the URL to open the selected data model page in **Talend Exchange** and view more information about the data model.

   If you want to delete any of the data models from the list, select the data model and click **Delete xsd modules**.

5. Click **OK** to validate the import operation and close the dialog box.

Appendix A. *Talend Open Studio for MDM* management GUI

This appendix describes the Graphical User Interfaces (GUI) of *Talend Open Studio for MDM*.

A.1. Main window of *Talend Open Studio for MDM*

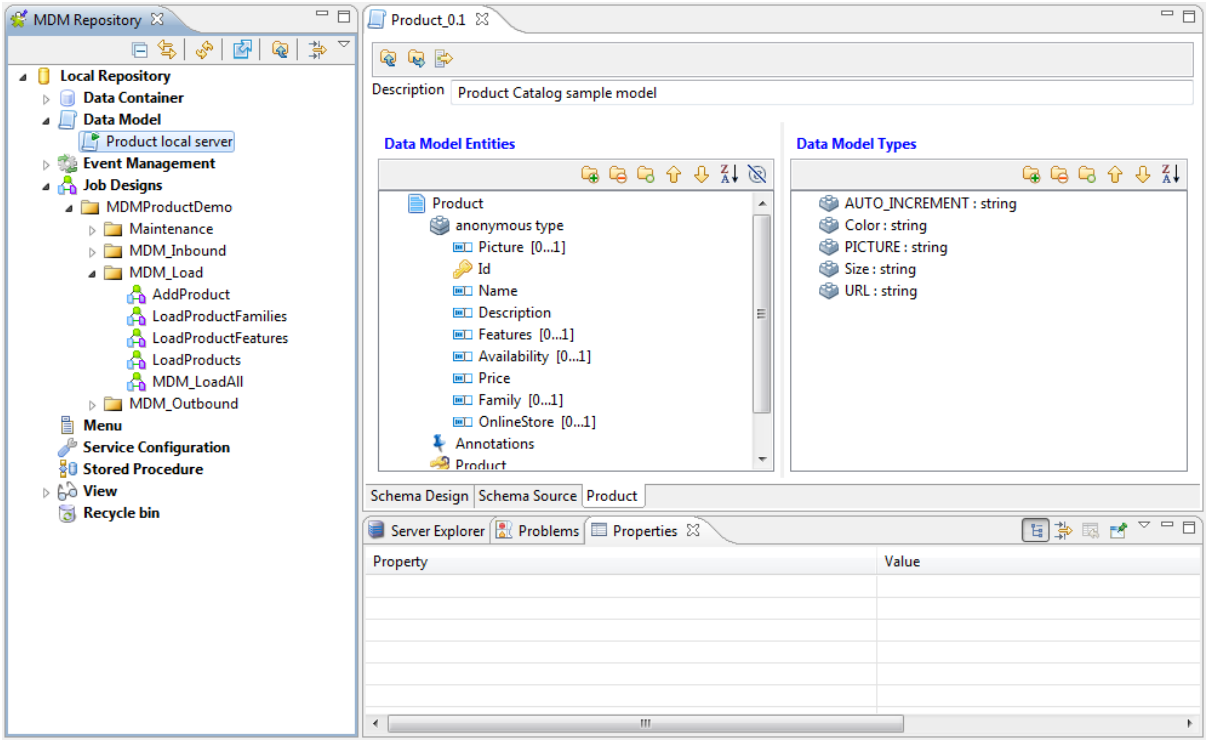
Talend Open Studio for MDM main window is the interface from which you can carry out administration tasks and set the parameters of the platform.

For more information about possible administration procedures within *Talend Open Studio for MDM*, see chapter *Setting data governance rules*, and chapter *Advanced subjects*.

The *Talend Open Studio for MDM* main window is divided into:

- a menu bar,
- a toolbar,
- a tree view area,
- a workspace,

The figure below illustrates an example of a *Talend Open Studio for MDM* main window.



A.2. Menu bar of *Talend Open Studio for MDM*

The menu bar headers and submenus help you perform operations in open editors and pages, interact with an application, or access help.

The table below describes various management items available in the menu.

Menu	Menu item	Description
File	**Close**	Closes the current open view in the workspace.

Menu	Menu item	Description
	Close All	Closes all open views in the workspace.
	Save	Saves any changes done in the current open view.
	Save All	Saves any changes done in all open views.
	Print	Unavailable option.
	Switch project...	Closes the current session and launches another one to enable you to open a different project in the Studio.
	Edit project properties	Opens a dialog box where you can customize the settings of the current project. For more information, see *Talend Open Studio for Data Integration User Guide*.
	Import...	Opens a wizard that helps you to import different types of resources (files, items, preferences, XML catalogs, etc.) from different sources.
	Export...	Opens a wizard that helps you to export different types of resources (files, items, preferences, breakpoints, XML catalogs, etc.) to different destinations.
	Exit	Closes the main window
	Open Files	Opens a file stored locally
Edit	**Undo**	Deletes the last action done in the workspace
	Redo	Redoes the last action done in the workspace
	Cut	Cuts selected object in the workspace
	Copy	Copies the selected object in the workspace
	Paste	Pastes the previously copied object in the workspace
	Delete	Deletes the selected object in the workspace
	Select All	Selects all components present in *Talend Open Studio for Data Integration* workspace
Window	**Perspective**	**Component Designer:** Opens the component designer
		Profiling: Opens *Talend Open Studio for Data Quality* perspective
		Data Explorer: Opens the data explorer perspective
		Integration: Opens *Talend Open Studio for Data Integration* perspective
		Master Data Management: Opens *Talend Open Studio for MDM* perspective
		Debug: Opens the debug perspective
	Show View...	Opens the **[Show View]** dialog box which enables you to display different views on *Talend Open Studio for MDM*
	Maximize Active View or Editor...	Maximizes the current perspective
	Preferences	Opens the **[Preferences]** window which enables you to set your preferences in *Talend Open Studio for Data Integration*
Help	**Welcome**	Opens a welcoming page which has links to *Talend Open Studio for Data Integration* documentation and **Talend** practical sites
	Help Contents	Opens the Eclipse help system documentation
	About	Displays: - the software version you are using - Installation details that include: detailed information on your software configuration that may be useful if there is a problem, detailed information about plug-in(s) and detailed information about the current Studio features
	Export logs	Opens the **[Export Logs]** wizard that helps to export to an archive file logs and system configuration

Menu	Menu item	Description
	Software Updates	-**Find and Install...:** Opens the **[Install/Update]** wizard that helps search for updates of currently installed features, and search of new features to install -**Manage Configuration...:** Opens the **[Product Configuration]** window where you can manage configuration for the current Studio
	View bookmarks	Opens a bookmark panel that holds some useful links. These links enable you to easily access specific information related to the usage of the current Studio

A.3. Toolbar of *Talend Open Studio for MDM*

The toolbar contains icons that provide you with quick access to the most commonly used operations performed from the *Talend Open Studio for MDM* main window.

The table below describes the toolbar icons and their functions.

Icon	Function
	Saves modifications
	Opens a list of available perspectives

A.4. Tree view of *Talend Open Studio for MDM*

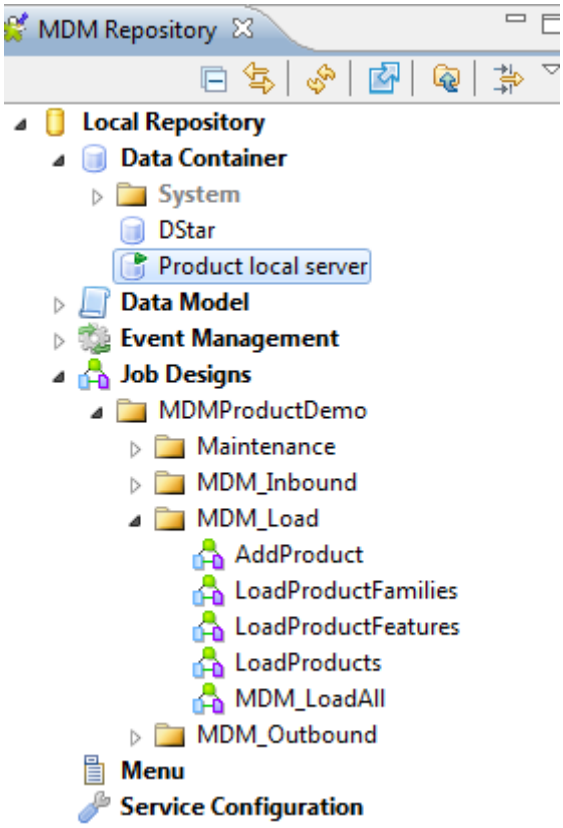
The **MDM Repository** tree view area shows nodes for manageable MDM objects including **Data Models**, **Data Containers**, **Processes**, **Views**, and **Triggers**.

If you right-click any of these nodes or any instance under the node, you display a contextual menu with multiple options.

The table below lists some of the available management options for MDM objects and instances.

Option	Description
Export Items	Exports one or multiple Data Containers from *Talend Open Studio for MDM* to a specified folder.
Import Items	Imports one or multiple Data Containers from a specified folder to the current *Talend Open Studio for MDM*.
New	Creates a new MDM object.
Create Category	Creates a new folder.
Edit properties	Edits the selected item.
Remove from Repository	Moves the selected item to the Recycle bin.
Copy	Copies the selected object(s).
Paste	Pastes the selected object(s).
Duplicate	Duplicates the selected object(s).

The figure below shows an **MDM Repository** tree view.



A.5. Workspace of *Talend Open Studio for MDM*

This area contains:

- nothing if no MDM object is open,
- parameter values or information about the open MDM object.

When you double-click an MDM object in the **MDM Repository** tree view area, the relevant page opens in the *Talend Open Studio for MDM* workspace. You can then browse items in the open MDM object or set parameters for the open MDM object depending on its type.

For detailed information about actions you can do on different MDM objects open in the workspace, see chapter *Setting data governance rules*, and chapter *Advanced subjects*.

Appendix B. MDM system routines

This appendix gives you an overview of the MDM system routines that you can use to factorize code and thus optimize data processing and improve job capabilities.

B.1. Accessing/managing MDM system routines

To access the MDM system routines, click on **Code > Routines > system** in the **Integration** perspective of *Talend Open Studio for MDM*.

You can customize system routines, create/edit user routines and libraries and call a routine from a Job. For more detailed information, see *Talend Open Studio for Data Integration User Guide*.

B.2. MDM Routines

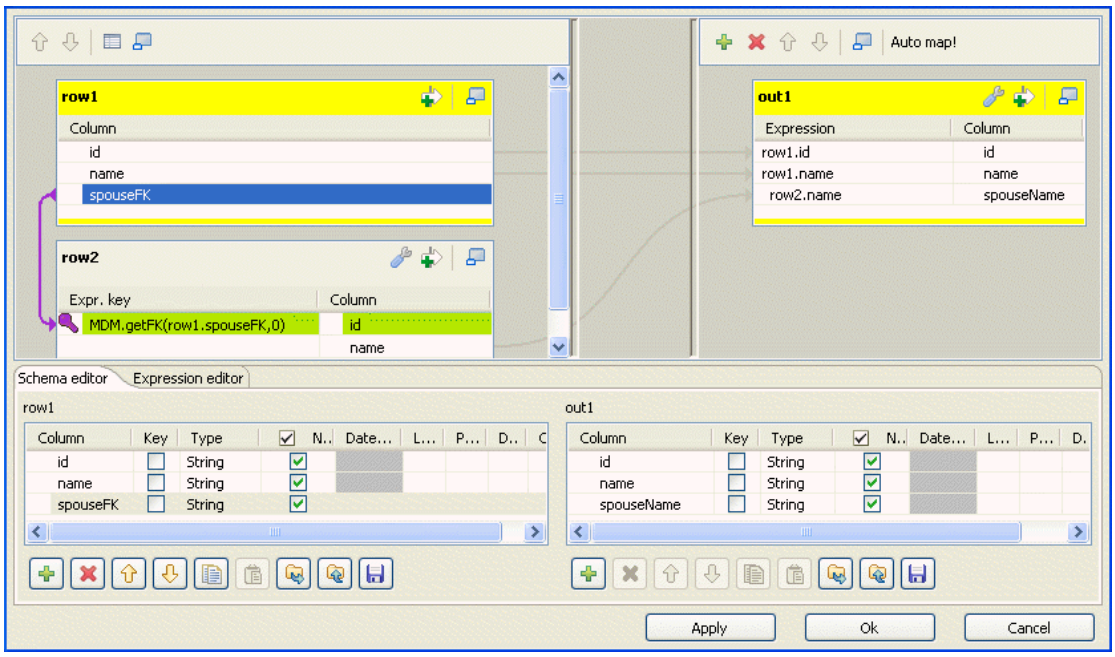
The table below lists some useful MDM routines including those for mangling Foreign Keys. In MDM, you usually need to mangle foreign keys to accommodate to compound keys.

To access these routines, double-click **MDM** in the **system** folder in the **Integration** perspective. The **MDM** category contains several routines including **getFK** and **createFK**:

Routine	Description	Syntax
getFK	Returns one of the FK components by position in a mangled FK.	`MDM.getFK` (FKs,0)
createFK	Returns the FK string by a single key (String singleKey).	`MDM.createFK("0")`
createFK	Returns the Fk string by a key list (String[] keys).	`MDM.createFK {"0","1"})`
getRepeatingElement	Gets repeating element in xmlString according to the xpath & position.	`MDM.getRepeatingElement`
hasRepeatingElement	Checks repeating elements in xmlString according to xpath & text;	`MDM.hasRepeatingElement`
listRepeatingElement	Lists repeating elements in xmlString according to xpath & delimiter.	`MDM.listRepeatingElement`
addRepeatingElement	Adds repeating elements in xmlString according to xpath & text.	`MDM.addRepeatingElement`
createReturnMessage	Generates an <error code="X">msg</ error> fragment.	`MDM.createReturnMessage {example} genErrMsg("test message",0)`
getNodeList	Gets a nodelist from an xPath.	`MDM.getNodeList`
parse	Parses the xml.	`MDM.parse`
nodeToString	Generates an xml string from a node with or without the xml declaration.	`MDM.nodeToString`

B.2.1. How to return one component of a mangled foreign key

The **getFK** routine allows you to return one component of a mangled foreign key using **tMap**, for example:



You must use the **getFK** MDM routine in the lookup table in the **Map Editor**. Using the **getFK** routine in the editor, you can add/remove/update the foreign key components. In the above capture, the routine helps you to return the *name* component in the mangled foreign key.

You must do the same in a reversed case and use the **createFK** routine to return the foreign key string by a single key.