EBX5 Add-on for Data Exchange

User Guide

GA 2.0.0

Last update: September 10, 2014

Table of contents

Overview

EBX5's Data Exchange add-on allows you to export and import data. You can initiate import and export in the following ways:

- A 'by default' mode allows the end-users–with limited IT knowledge–to easily export and import data when source and target data structures are similar. Because this mode uses a direct mapping between the source and the target, a specific data mapping configuration isn't needed.
- When the source and target data structures are different, the extended mode allows IT staff to manage user-defined data mapping configurations. This mode is used to change the name of the field from the source to the target, to transform the data during the data flow processes, to aggregate or split data values, etc.

Data mapping configuration is based on basic concepts of Tables and Fields. When a company wants to strengthen business representation of data mapping configurations, Tables and Fields can be connected to business concepts also known as Object Classes and Properties, respectively. The addition of this business level improves the data lineage audit and encourages actual information governance.

Special notation key

✔	Important recommendation for the use of the feature

✘	This feature is not yet available in the current release

Overview of updates

GA 2.0.0

User 's operations for XML import-export and data transfer

- Export and Import in XML based on a 'by default' data mapping.
- Export and Import in XML based on a user-defined data mapping.
- Data transfer between EBX tables relying on the same data model.
- Data transfer between EBX tables relying on different data models and based on a user-defined data mapping.

Extended operations to manage user-defined data mapping configuration

- These operations are located in the Data Exchange's configuration data set. They are used by IT specialists to create and maintain user-defined data mapping configurations between source and target applications. These configurations allow end-users to import, export and transfer data.
- Creation of the application portfolio for XML and EBX types.
- From an EBX type application, automatic generation of the Tables, Fields, Object Classes and Properties declaration.
- From an XML type application, automatic generation of the Tables and Fields declaration with XML paths (based on an XML sample that is provided as input parameter of the operation).
- Automatic detection of any misaligned items between a data mapping configuration and its underling EBX application.
- Automatic data mapping configuration between two EBX type applications sharing the same Object Class and Property items.

1. Functional scope

EBX5's Data Exchange Add-On relies on a repository that collects all data mapping configurations. This repository automatically updates when a user executes a mapping operation that invokes the default data mapping. For instance, during export execution for the 'Client' table in the 'Sales' data set, the Data Exchange repository is enriched with new source and target 'Sales' applications of EBX and 'Default XML' types, respectively. The data mapping configuration between the source fields (EBX paths) and the target fields (XML nodes) are created automatically. The next time this export process is executed, the repository updates automatically if needed (changes in the data structure).

This data can be reused to create user-defined data mapping configurations. Indeed, the repository can be enriched manually if default mappings do not meet business requirements. For example, when the names of the metadata from the source are different from the target, a user-defined data mapping configuration must be declared.

1.1. Semantic model

Data mapping configurations can be enriched by creating links from Tables and Fields to corresponding business concepts also known as Object Class and Property items. A single Object Class can be linked to many tables, and a Property can be linked to many fields. An Object Class is a container of data, predominantly a Table, but can include a group of fields or a complex data type. A property is a business abstraction of a table's field. This vocabulary comes from the ISO11179 standard also used in the EBX5's Information Governance add-on.

The use of the semantic model is not mandatory to create data mapping configurations or to enable import, export and data transfer. Once the semantic model is configured, it facilitates the management of the data mapping configurations as follows:

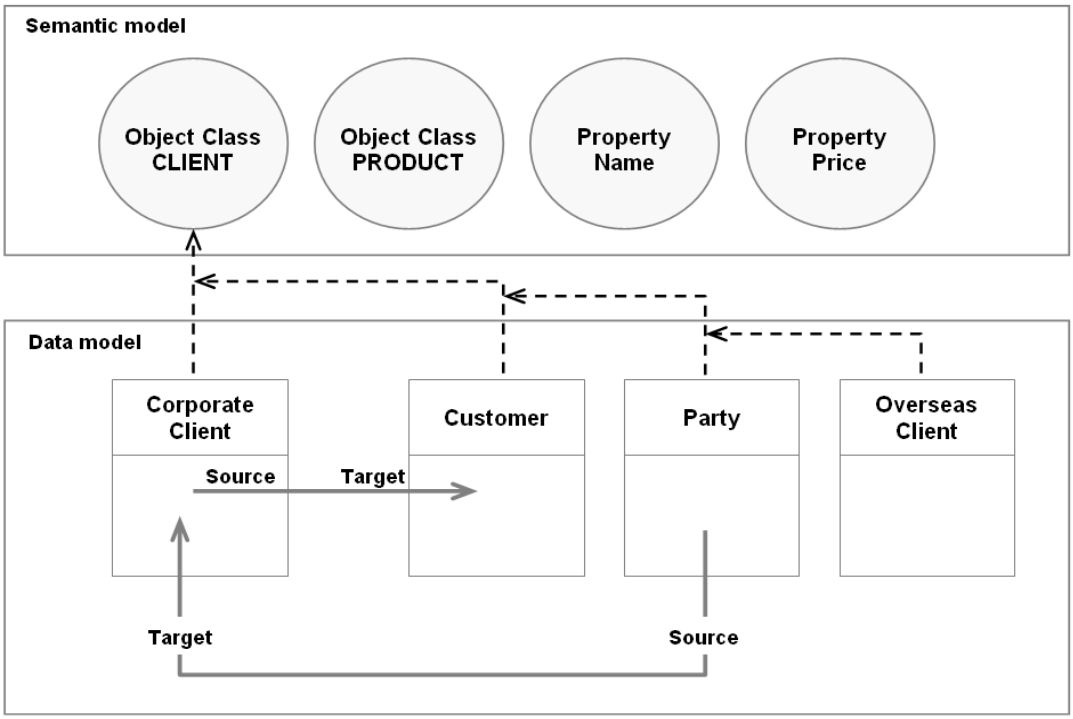
- Business data lineage.
- Automatic data mapping.
- Integration with the governance process.

1.1.1. Business data lineage

Data lineage shows a global view of how a data is transformed and conveyed between applications playing the roles of producers and consumers. When data lineage applies solely to the logical data model level, it is not easy to enforce a full understanding of the transformation. For instance, two tables can have indirect data flow processes that cannot be figured out using just the logical level:

- There are four tables: A, B, C and D. There is a data mapping from A to B and another mapping from C to D. Data lineage only recognizes two possible ways for data to flow A->B and C->D. But from a business point of view, table 'D' and table 'A' have similar significance. Even though a logical data mapping between A and D doesn't exist, the business data lineage must represent the global linking meaning between the four tables. It could be considered that there is a missing data mapping configuration between A and D.

In the following example, the 'CLIENT' Object Class is linked to the 'Corporate Client', 'Customer', 'Party' and 'Overseas Client' tables. From this Object Class, it is easy to get a full data lineage applied to the business concept of 'Client', including the information that the table 'Overseas Client' is not involved to feed the 'Customer' table.

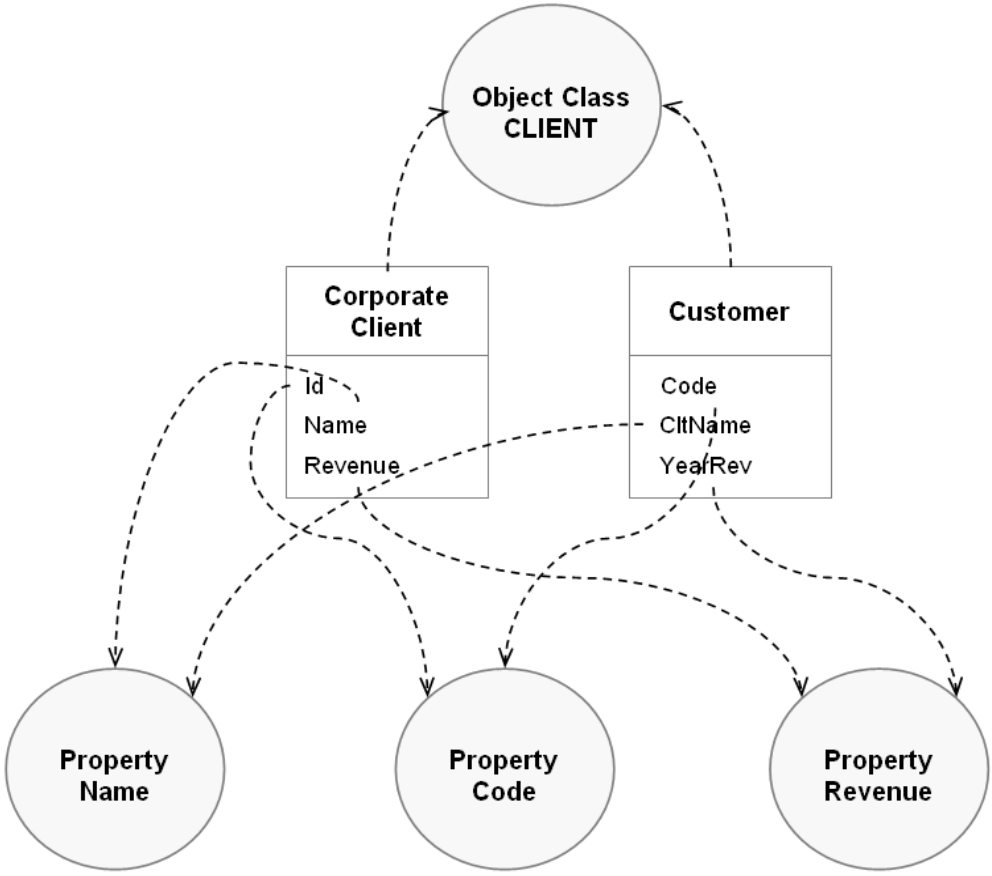


✕	In the current version of the add-on, it is possible to link the Tables and Fields with the Object Class and Properties items, but the UI to display the data lineage is not yet available.

1.1.2. Automatic data mapping

The semantic model is also used to create automatic data mappings between tables and fields sharing the same Object Class and Property items or a part of them. As illustrated below, the two tables 'Corporate Client' and 'Customer' share the same 'CLIENT' Object Class, meaning that they can be mapped with each other. The 'Id' and 'Code' fields are linked to the same 'Name' Property, meaning that they can also be mapped with each other.

Based on this link between the data model (Tables and Fields) and the semantic model (Object Classes and Properties), the add-on can automatically generate the data mapping configuration between source and target tables, such as, in the example below, the 'Corporate Client' table and the target 'Customer' table.



✔	You can refer to the appendix 'Automatic data mapping for transferring data' to get a use case relying on a semantic data model.

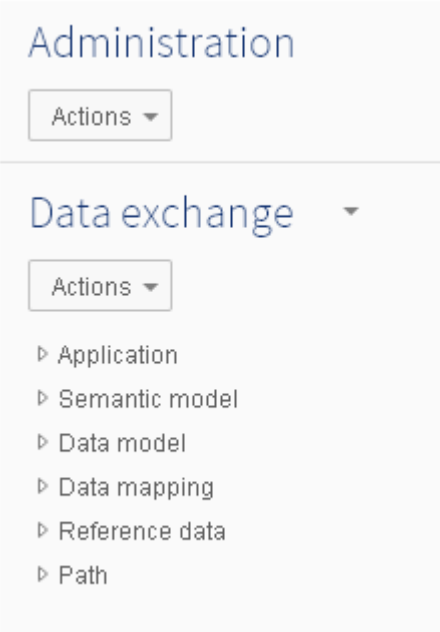
1.1.3. Integration with the governance process

In the Information governance field, EBX5's Information Governance add-on allows you to manage all metadata of any data asset, such as: data models, workflow, rules, data spaces, applications, etc. This add-on uses Object Class and Property items as concepts to arrange the metadata and govern their definitions.

It is also possible to declare the parties and their roles involved in each Object Class and Property. For instance, if the 'Sales' application is referenced as the 'Consumer' of the 'Client' Object Class, it should be forbidden to declare this application as a source for a export process in Data Exchange.

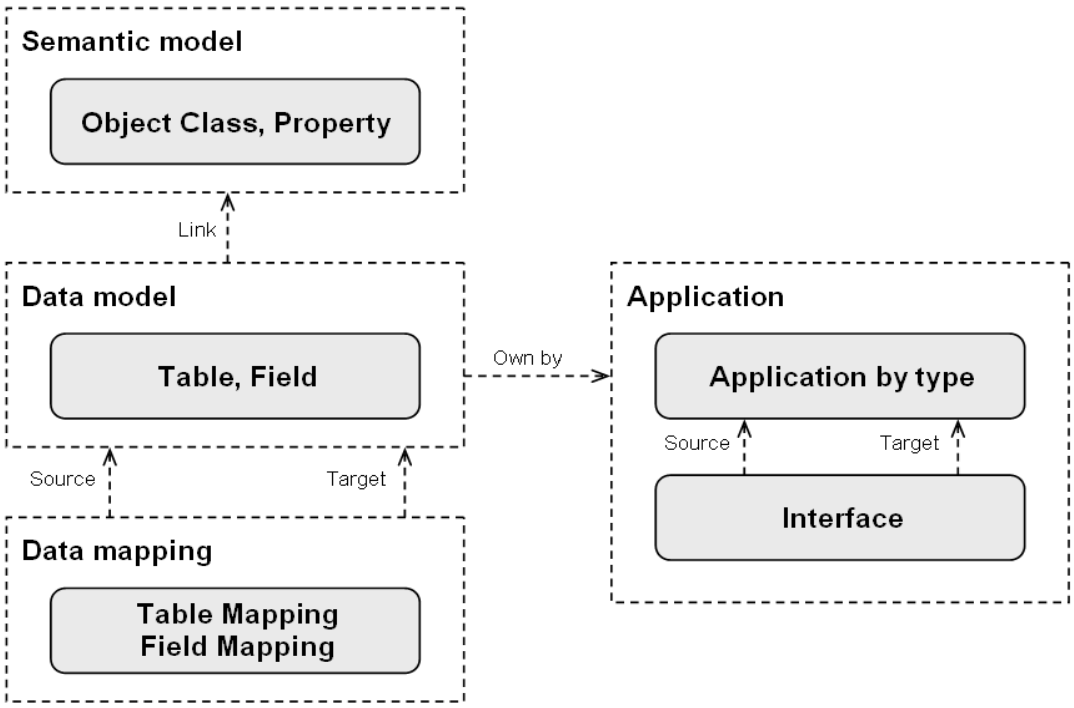
✕	The integration between the add-ons Information Governance and Data Exchange will be available in the further version

1.2. Key definitions



The table on the next page defines Data Exchange repository key concepts.

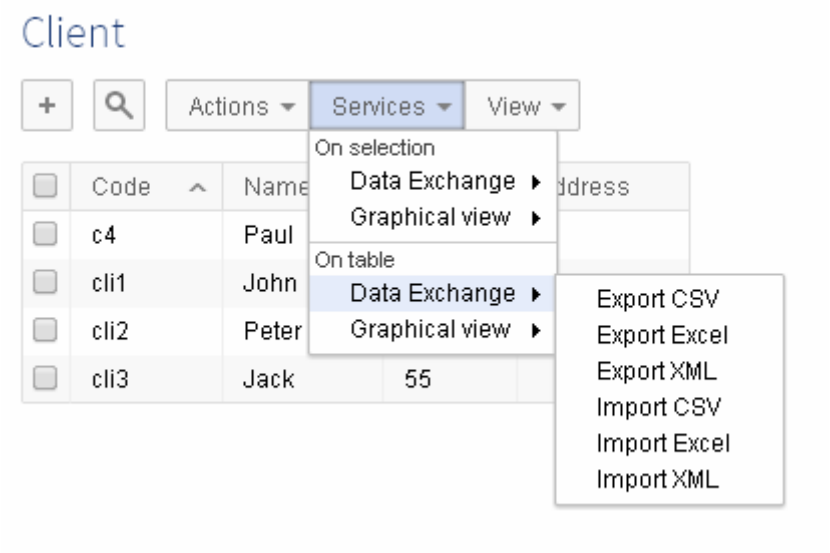
This repository is located under the EBX5 'Administration' tab in the 'Data exchange' data space. It contains the following domains: Application, Semantic model, Data model and Data mapping.



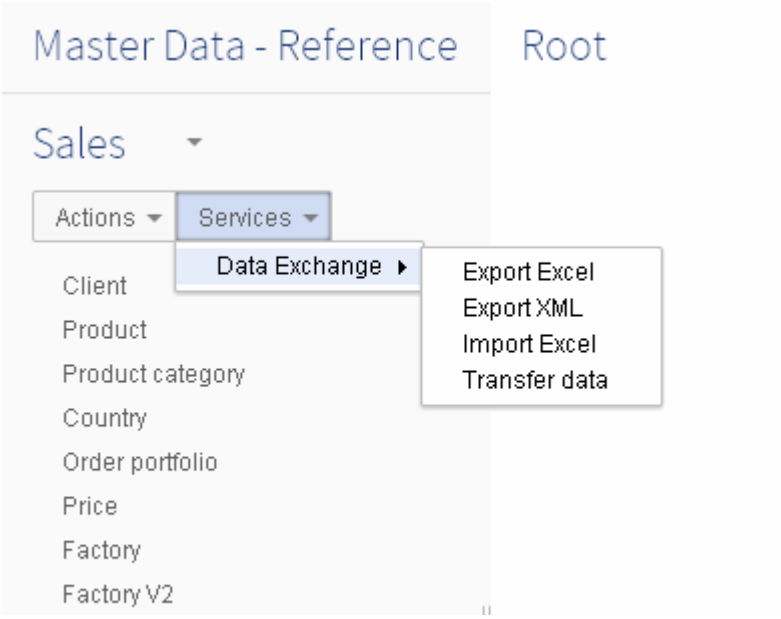
Concept	Definition
Semantic model	Using the semantic model is not mandatory to create data mapping configurations. However, it provides an improved data lineage view. Also, automatic data mapping between tables and fields–sharing the same Object Classes and properties–uses the semantic model.
Object Class	An Object Class is a container that holds metadata such as a table, a group of fields or a complex data type. An Object Class can be linked to one to many Tables.
Property	A Property is a business abstraction of a table's field, group of fields or complex data type. A Property can be linked to one to many Object Classes and Fields.
Data model	A data model contains the Tables and Fields that comprise an application. If it is an EBX type application (referenced as a data set), it corresponds to the logical data model. When it is an XML type application, it corresponds to the XML paths.
Table	When referring to an EBX type application, this is the table in the related logical data model. When referring to an XML type application, this is an XML path.
Field	When referring to an EBX type application, this is the field in the related logical data model. When referring to an XML type application, this is an XML path.
Application	The application level declares the portfolio of source and target applications involved in the import, export and transfer of data.
Application by type	An application is declared for one to many format types: • EBX: the application corresponds to a data model in EBX5—referenced through a data set. • Default XML: the application corresponds to a XML data structure fully managed by the add-on with default XML paths. • XML: the application corresponds to a XML data structure issued in the user-defined data mapping configurations.
Interface	An interface is comprised of a source and target application and enables data flow between the two. Import, export and transfer of data is not possible between applications without an interface definition. The add-on creates an interface when the end-user executes a default XML import or export.
Data mapping	This level contains the actual mapping configuration between tables and fields.
Table mapping	Defines the source and target tables.
Field mapping	Defines the sources and target fields.

2. User operations

At the table level, Data Exchange services allow you to export and import data in the CSV, Excel and XML formats.



At the data set level–in addition to being able to import and export in Excel and XML–Data Exchange services allow you to transfer data. The transfer copies data directly between EBX tables.



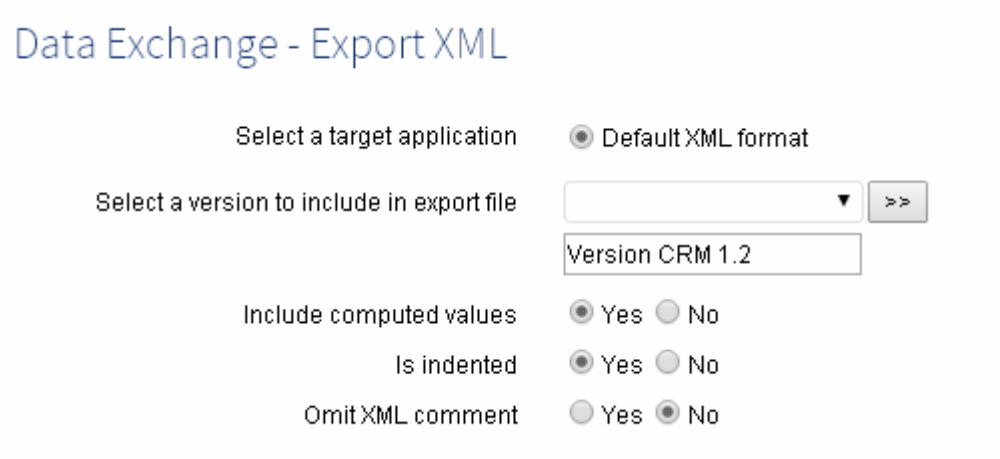
2.1. XML with the default data mapping

The following sections describe options when you use the 'Data Exchange' 'Export XML' and 'Import XML' services.

2.1.1. XML export

If the table is not defined in the data mapping configuration when you attempt to export to XML, 'Default XML format' displays as the only available target application. This default format, provided by the add-on, allows you to easily get an XML file and doesn't require a specific data mapping configuration.

If a field name has to be renamed when exporting to an XML file, or another instance arises where data in the source and target differ, you have to specify a data mapping configuration. In these types of cases, other target applications display as options (see the rest of this user guide).



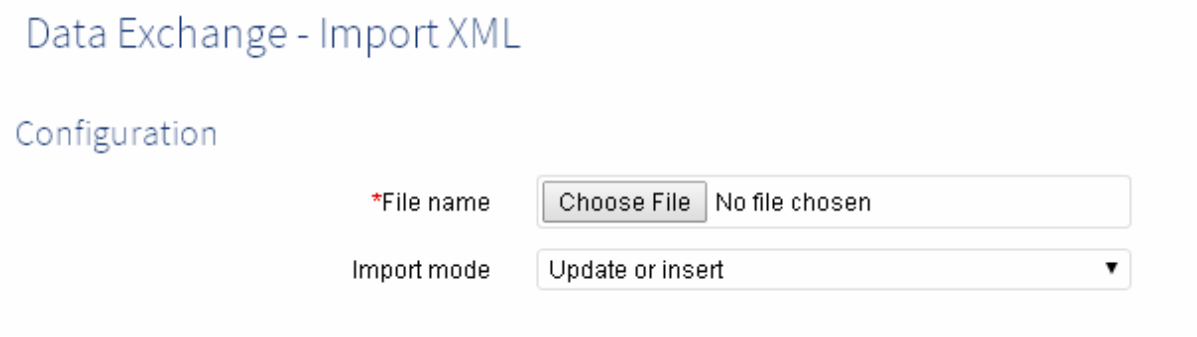
Option	Description
Select a version to include into the exported file	The version is for information only. It is inserted in the exported file as the 'appversion' XML tag.
Include computed value	Specifies whether or not computed values must be included in the export.
Is indented	Specifies whether or not the file should be indented to improve readability.
Omit XML comment	Specifies whether or not the generated XML comment that describes the data location and export date should be omitted from the file.

The exported XML file contains a standard header with the following data:

- The name of the application. By default, this is the name of the data set in which the table is located (a timestamps value is also added to build the name).
- The version that has been selected during the export configuration. If no specific version was specified, this will be blank.
- The export date.
- The type of the XML export, either 'Default XML' or 'XML' when a user-defined data mapping configuration is used.

2.1.2. XML import

Two import modes are available: 'Update or insert' or 'Replace all content'.

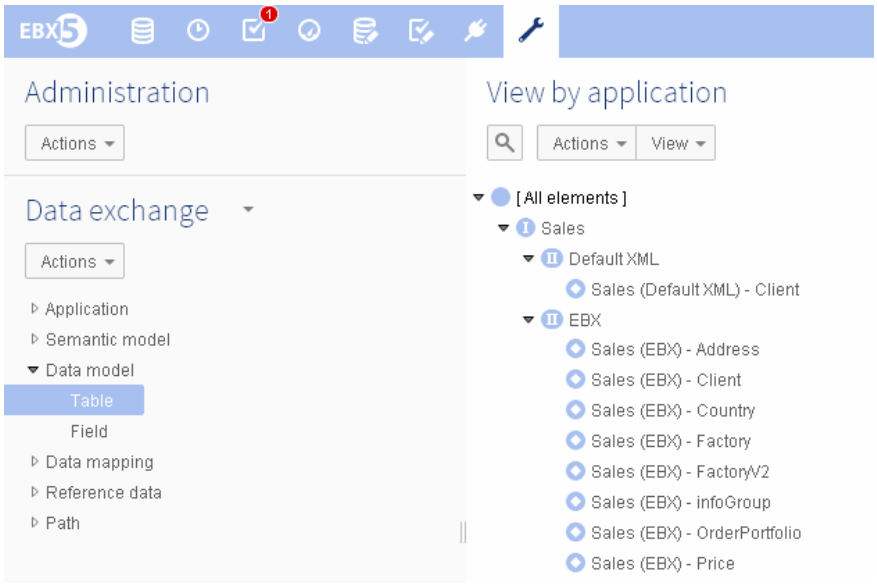


The add-on automatically identifies which data mapping to apply in order to import the data into the table. When the 'Default XML' is declared in the header of the imported XML file, then the default data mapping is applied. A UI result message provides an overview of the import process, as illustrated below.

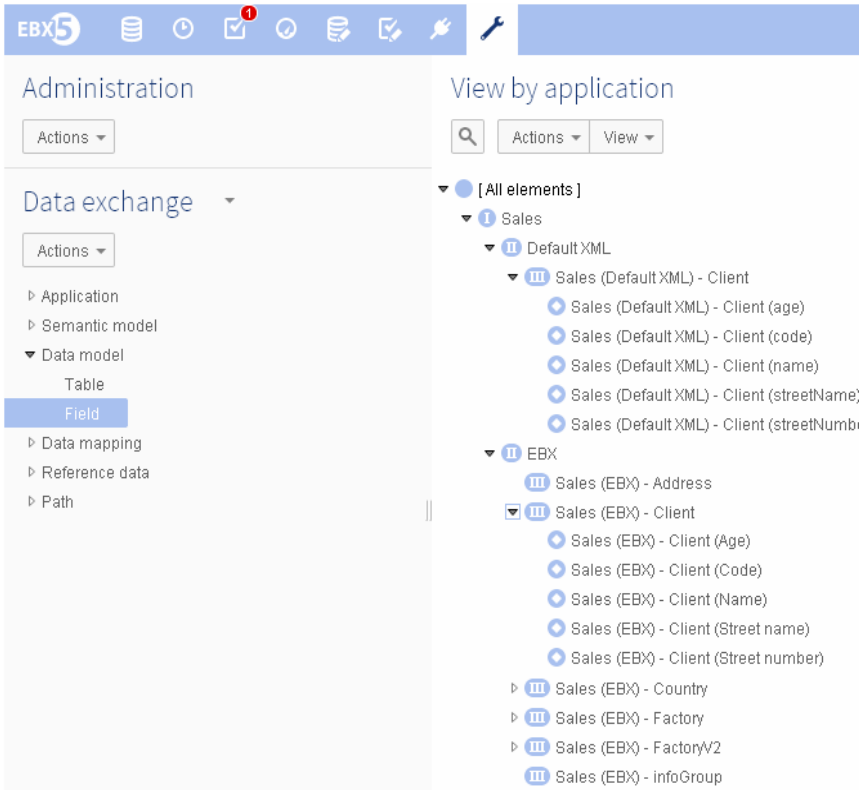
2.1.3. The default XML data mapping configuration

When the XML export execution process is based on the default XML configuration, the add-on generates the related data mapping configuration. It contains the correct XML paths used for each of the exported table's fields. Even though this default XML configuration cannot be modified, understanding its architecture can be of benefit to you. Indeed, if you need to configure a user-defined data mapping, the same architecture applies (see the rest of the user guide).

The following images give an overview of how this configuration is saved in the Data Exchange repository. You can access to this repository from the 'Administration' tab's 'Data Exchange' data space.



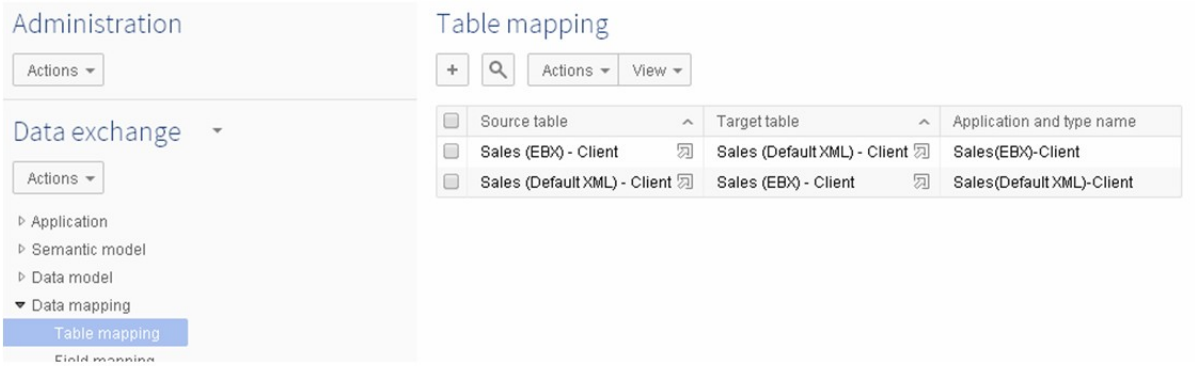
The 'Sales' (name of the data space where the table is located) application is created with the two application types, 'Default XML' and 'EBX'. For the 'Default XML' type, only the exported table is declared ('Client'). For the 'EBX' type, all tables and field groups located in the data space are declared.



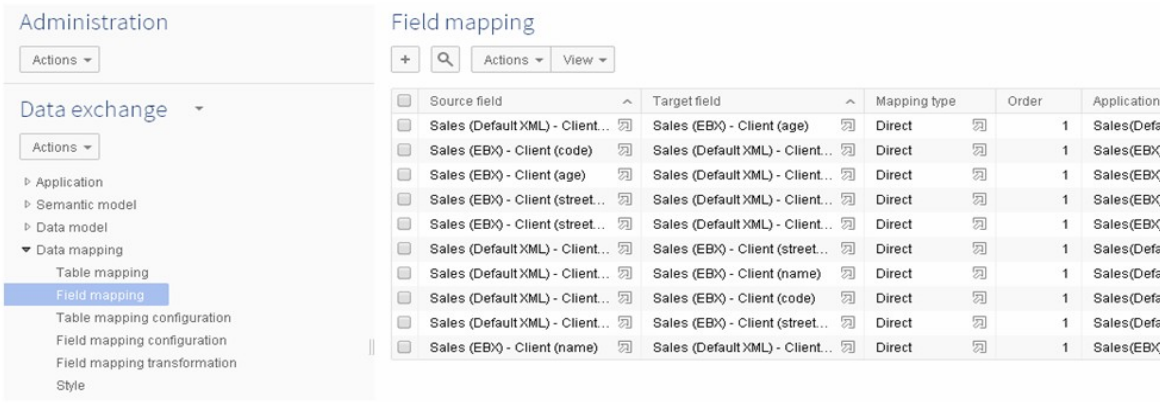
The declaration of the XML nodes for the 'Client' table is done with the default XML naming convention (direct reuse of the naming from the table in EBX).

The declaration of the fields for every table in EBX is also performed.

The 'Table mapping' is automatically provided. The source table has been declared as EBX for the export process. And the source table has been declared as 'Default XML' for the import process.



Then the add-on automatically creates the declaration for the 'Field mapping', thus providing the link between the source and the target file.



2.2. XML with a user-defined data mapping

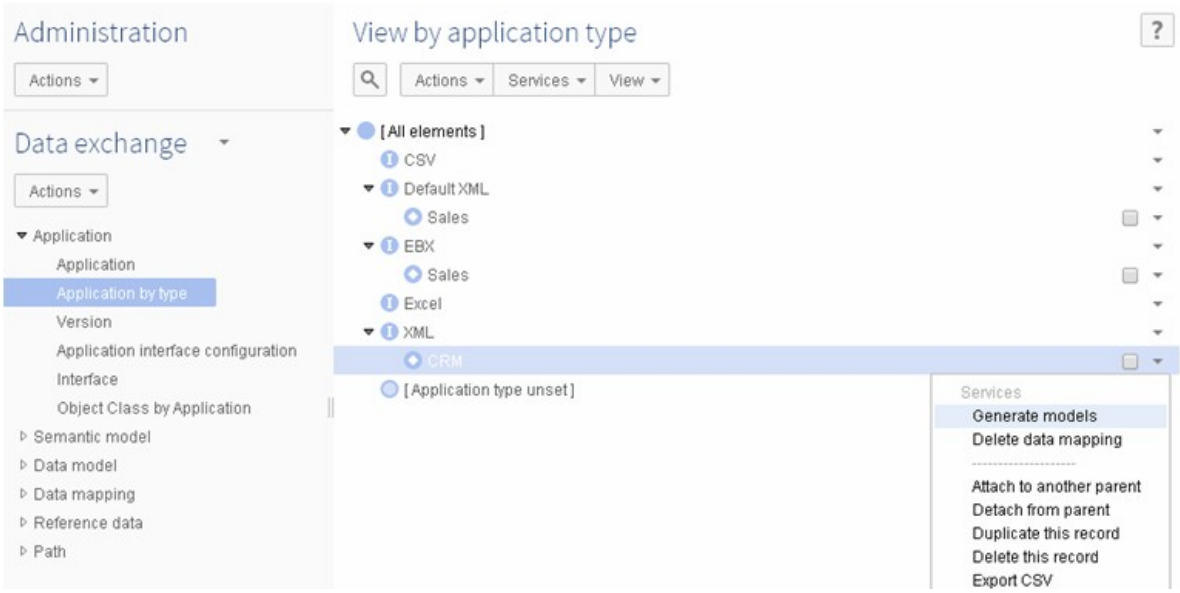
The following section describes a user-defined data mapping.

2.2.1. XML export

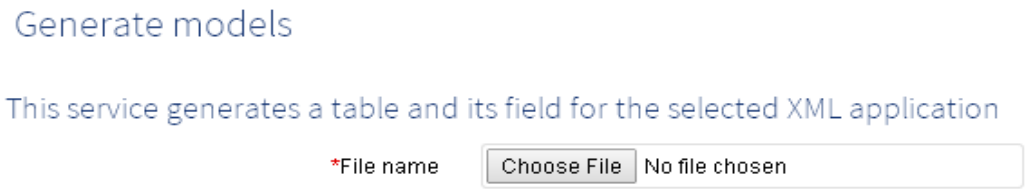
To export an XML file with a data structure that is not compliant with the 'Default XML' type, you can manually configure a user-defined data mapping.

To facilitate the configuration, the add-on can automatically analyze your XML file to generate the XML configuration as illustrated below. In most cases the XML file you want to get already exists, and you can reuse it as a template to automatically feed the configuration.

First, a new XML type application has been created manually. In the following image, this is the 'CRM' application. Since the type is XML (not 'Default XML'), the Data Exchange 'Generate models' service is available and allows you to get the XML configuration automatically.

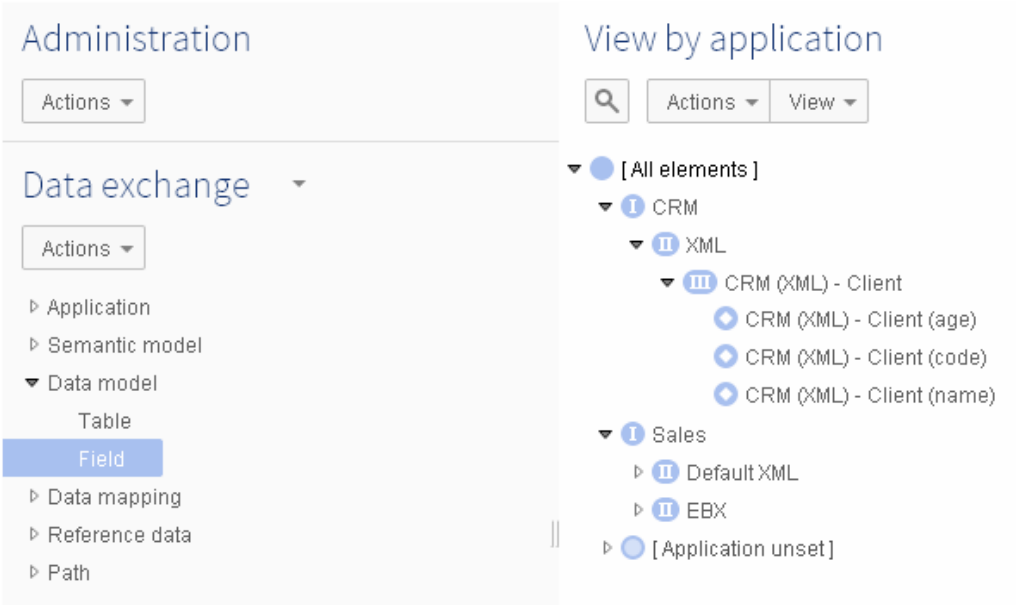


The add-on allows you to enter the XML file that will be used to create the XML configuration.

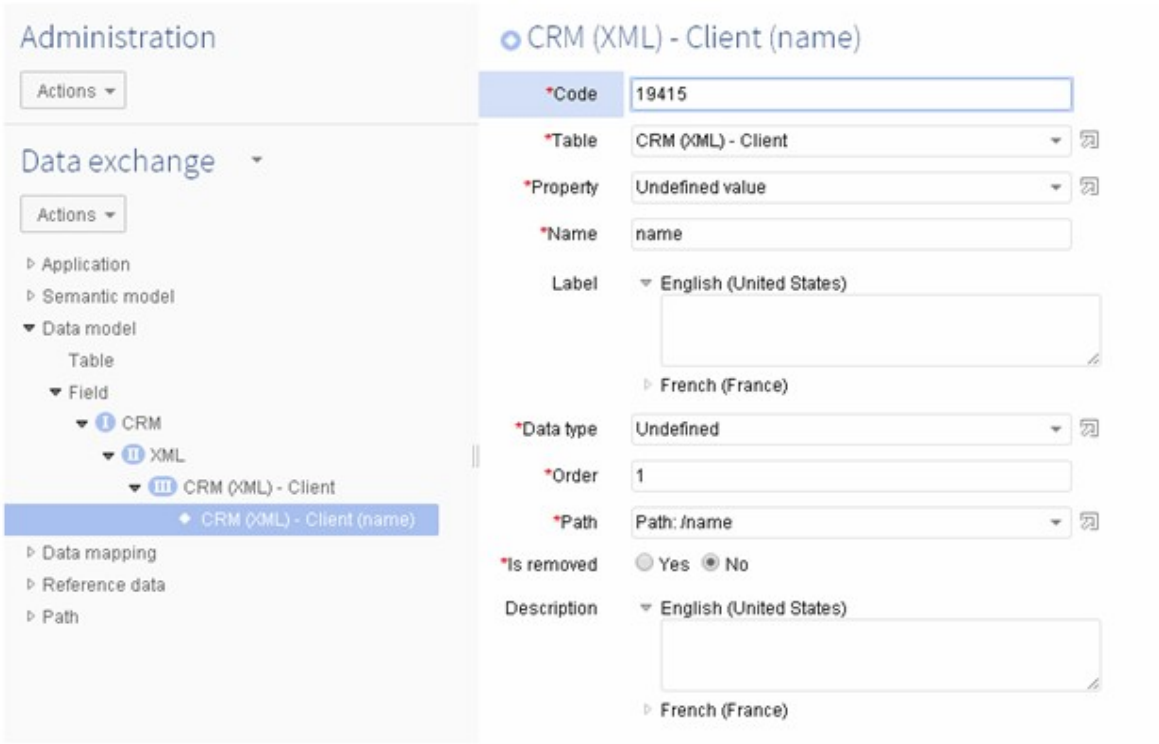


Then you select a node path in the XML file from which the XML configuration must be applied.

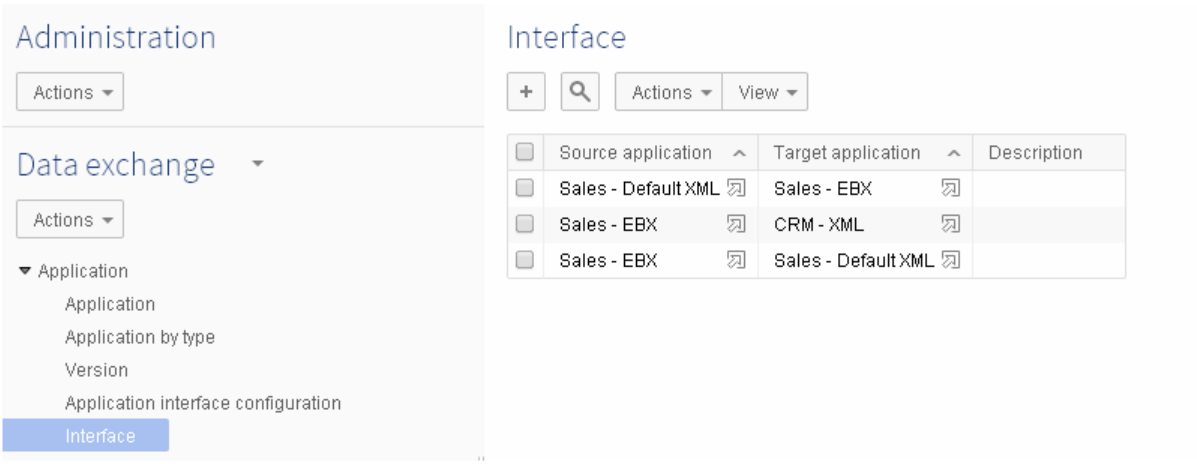
The result of the configuration is displayed below. The CRM XML type application is now declared with the fields corresponding to the XML tags existing in the file.



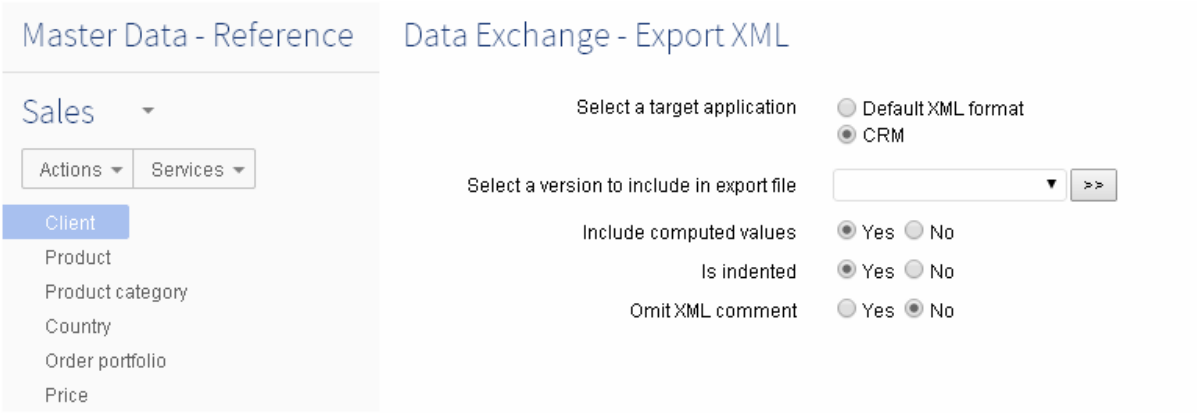
The configuration of a field with its XML path can be adapted manually as illustrated below. You can change the name, the path, etc. Conversely, in the case of a configuration based on the 'Default XML', the names and paths cannot be modified and are under add-on control.



To make the export process available from the EBX type application ('Sales' data space) to the CRM application, the 'Interface' between both must be declared as follows:

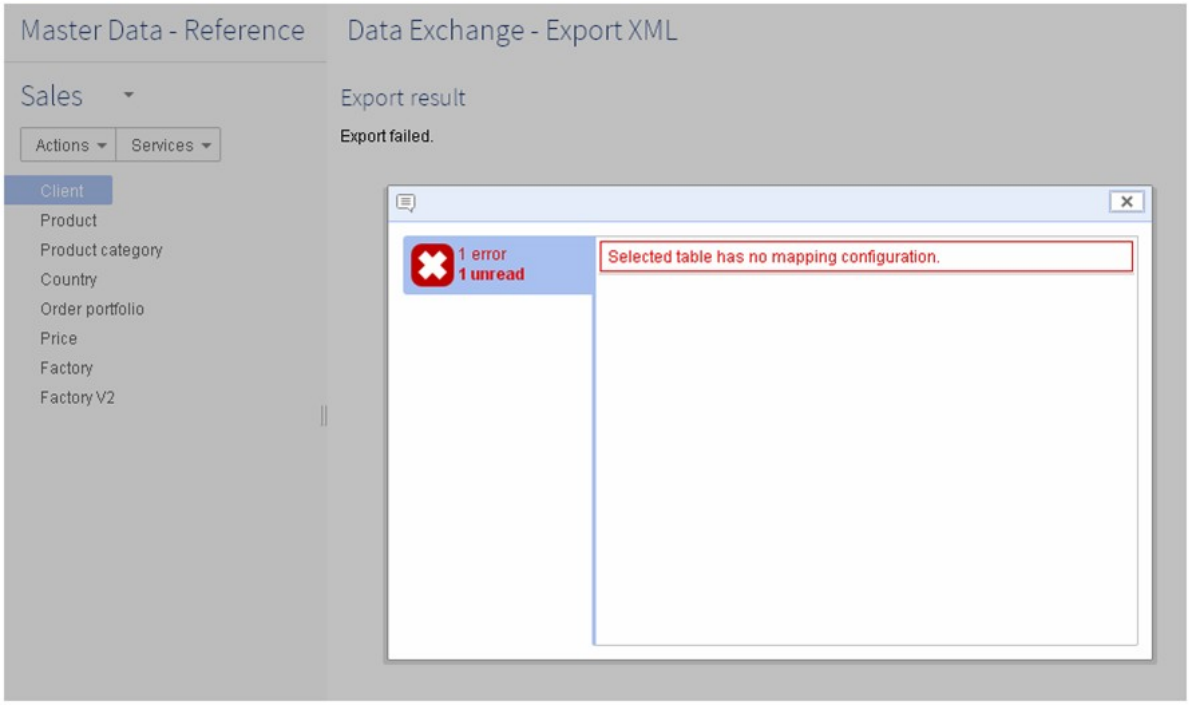


Now, the export UI displays the 'CRM' target application as an option:

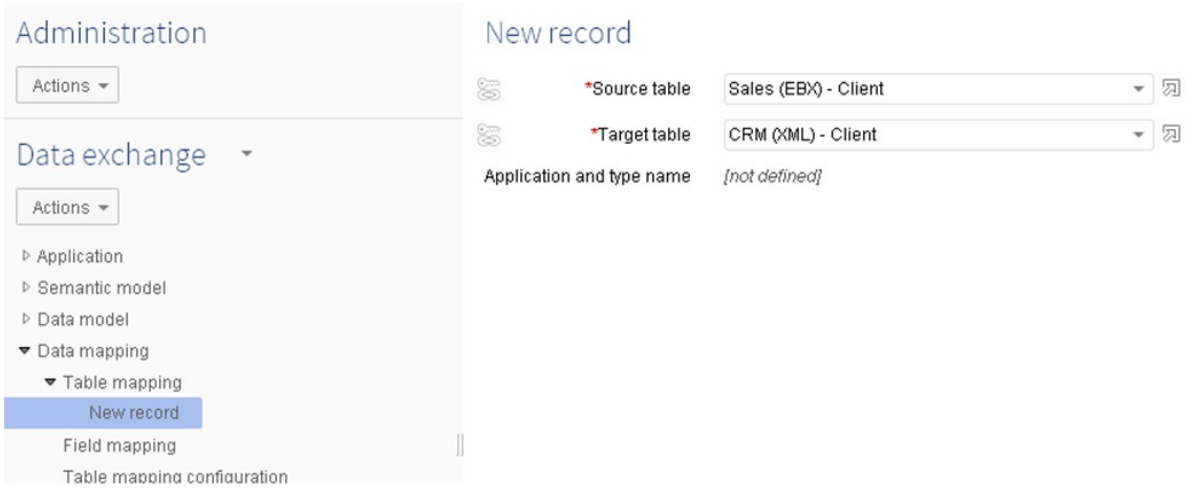


The options are already described in the previous section.

When executing the export process based on this configuration, the add-on raises this error:



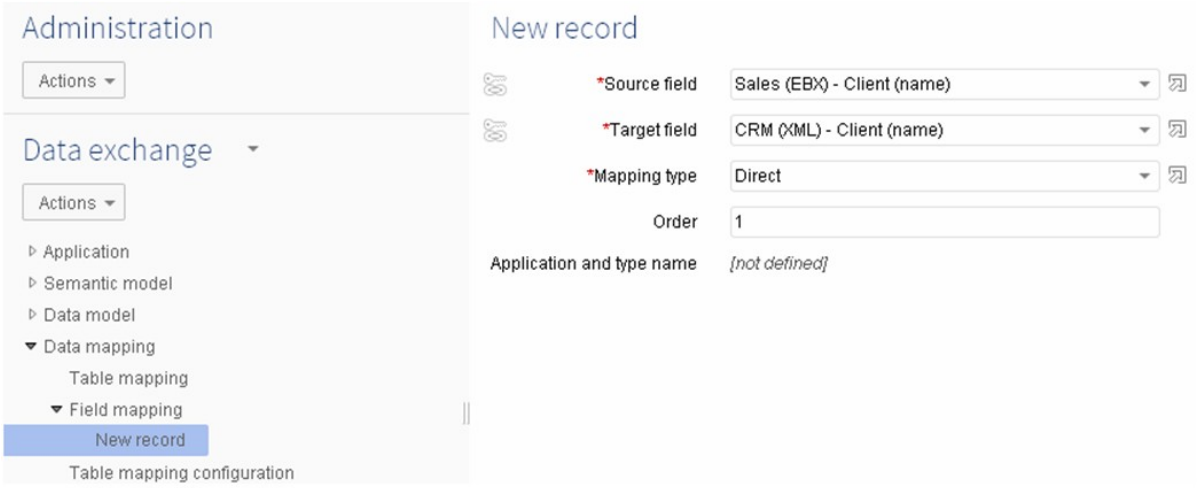
Indeed, even though the XML field configuration is done, the 'Table mapping' and 'Field mapping' are not yet configured. The 'Table mapping' is declared as follows:



A new execution of the export process entails a useless result because the fields are not exported until the 'Field mapping' is configured:

```
<?xml version="1.0" encoding="UTF-8" ?>
<!--XML content generated for /root/Client in data set Sale
<dataexchange>
        <header>
                <appname>CRM-1407409946284</appname>
                <appversion></appversion>
                <date>2014-08-07</date>
                <apptype>[ON] XML</apptype>
        </header>
        <root>
                <Client>
                </Client>
                <Client>
                </Client>
                <Client>
                </Client>
                <Client>
                </Client>
        </root>
</dataexchange>
```

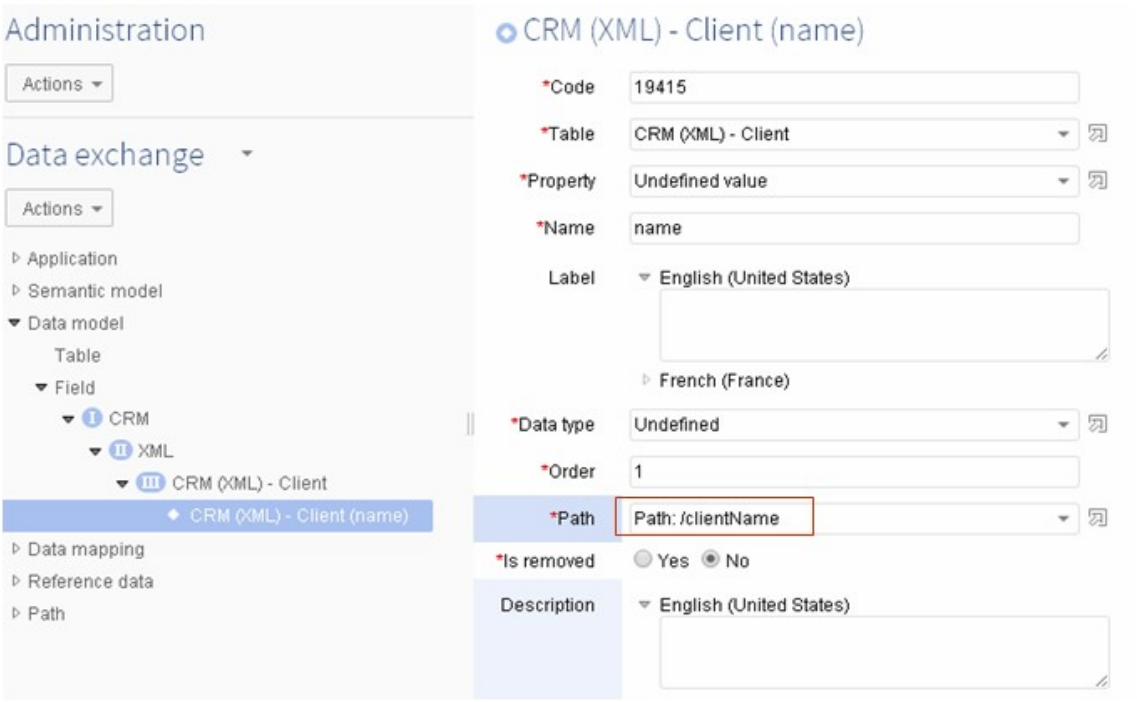
For every field, the mapping from the source to the target is declared as follows:



Now, the export process integrates the fields that have been configured (in this example only the name is declared in the 'Field mapping' table).

```
<?xml version="1.0" encoding="UTF-8" ?>
<!--XML content generated for /root/Client in data set Sales in
<dataexchange>
        <header>
                <appname>CRM-1407409946284</appname>
                <appversion></appversion>
                <date>2014-08-07</date>
                <apptype>[ON] XML</apptype>
        </header>
        <root>
                <Client>
                        <name>Paul</name>
                </Client>
                <Client>
                        <name>John</name>
                </Client>
                <Client>
                        <name>Peter</name>
                </Client>
                <Client>
                        <name>Jack</name>
                </Client>
        </root>
</dataexchange>
```

To change the path of a field, a direct modification in the path configuration can be done as illustrated below. The initial path '/Client' has been changed into '/clientName'.



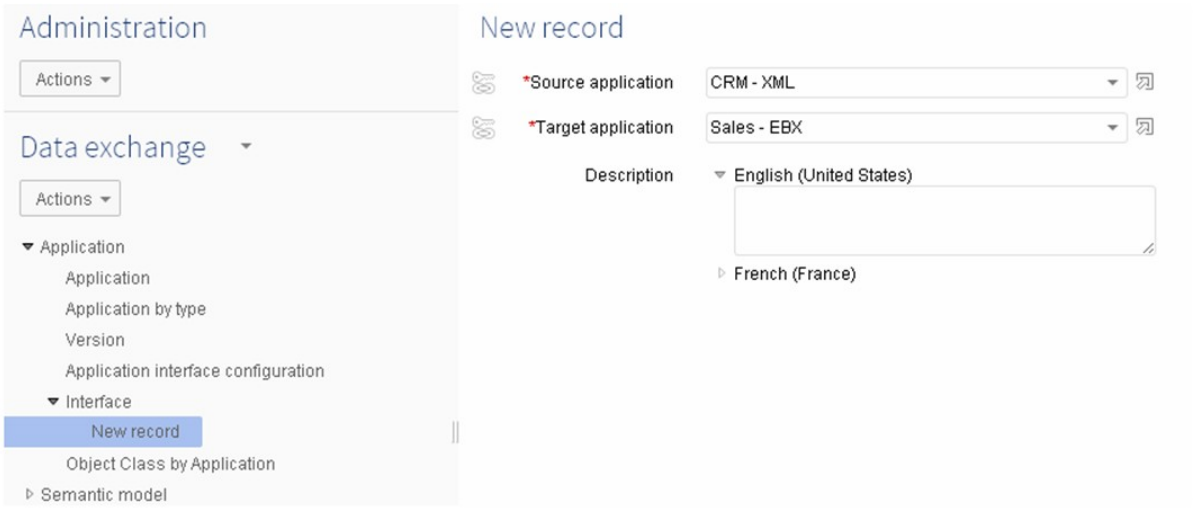
Now, an export process execution generates the following result (new path 'clientName'):

```
<?xml version="1.0" encoding="UTF-8" ?>
<!--XML content generated for /root/Client in data set Sales in
<dataexchange>
        <header>
                <appname>CRM-1407409946284</appname>
                <appversion></appversion>
                <date>2014-08-07</date>
                <apptype>[ON] XML</apptype>
        </header>
        <root>
                <Client>
                        <clientName>Paul</clientName>
                </Client>
                <Client>
                        <clientName>John</clientName>
                </Client>
                <Client>
                        <clientName>Peter</clientName>
                </Client>
                <Client>
                        <clientName>Jack</clientName>
                </Client>
        </root>
</dataexchange>
```

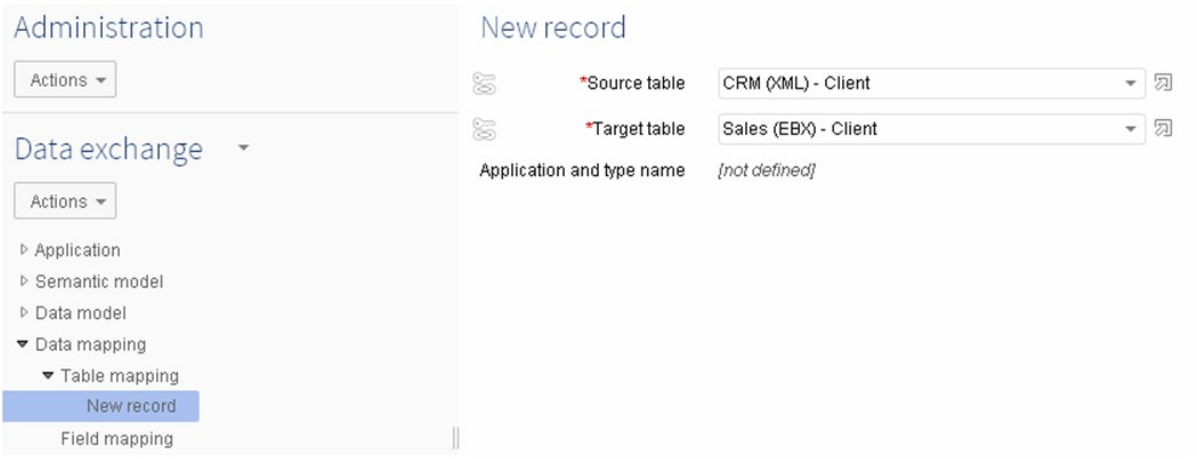
2.2.2. XML import

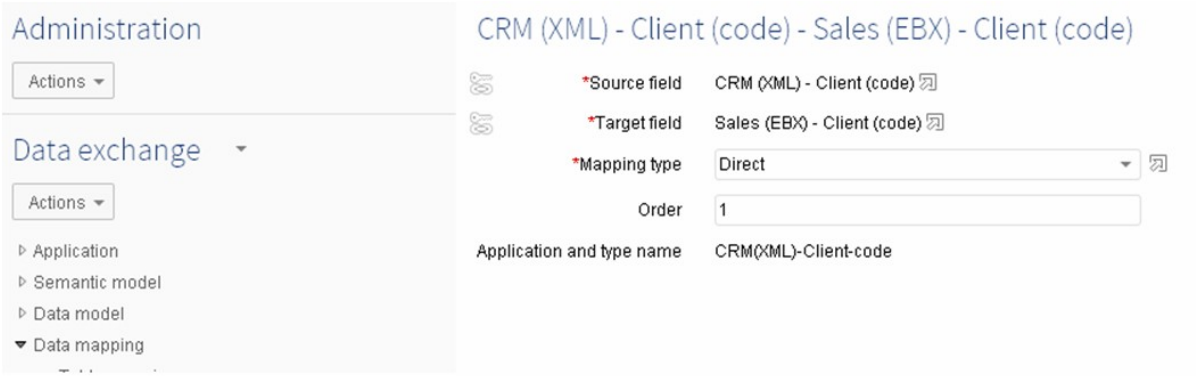
To import an XML file with a data structure that is not compliant with the 'Default XML' configuration, you can manually configure a user-defined data mapping using the same process described in the previous XML export section. You can use the XML import file as a template to automatically generate the XML configuration.

Then you create an 'Interface' from the source XML type application ('CRM') to the target EBX type application ('Sales').



The 'Table mapping' and 'Field mapping' must also be configured:



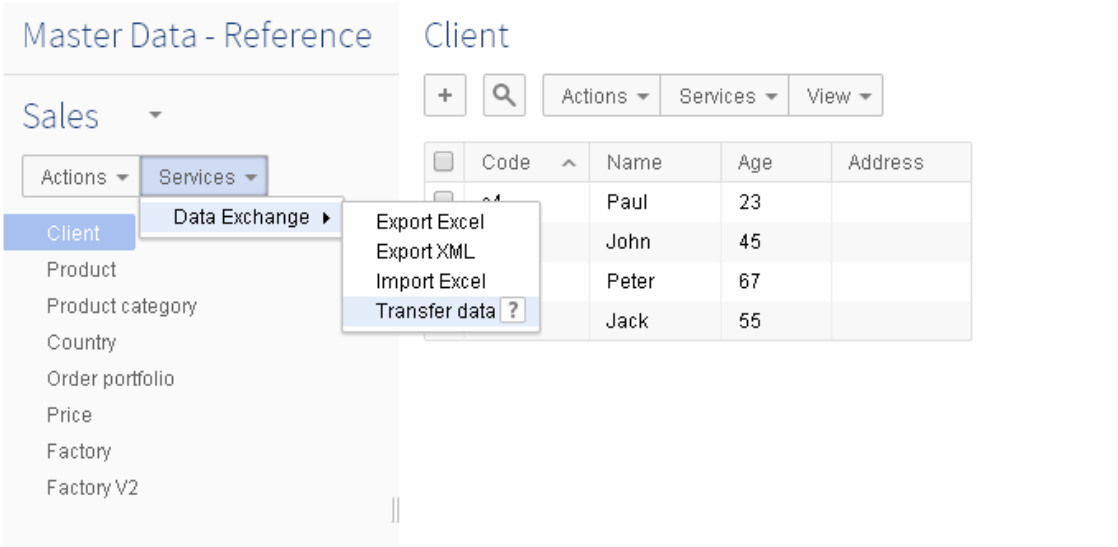


2.3. Data transfer

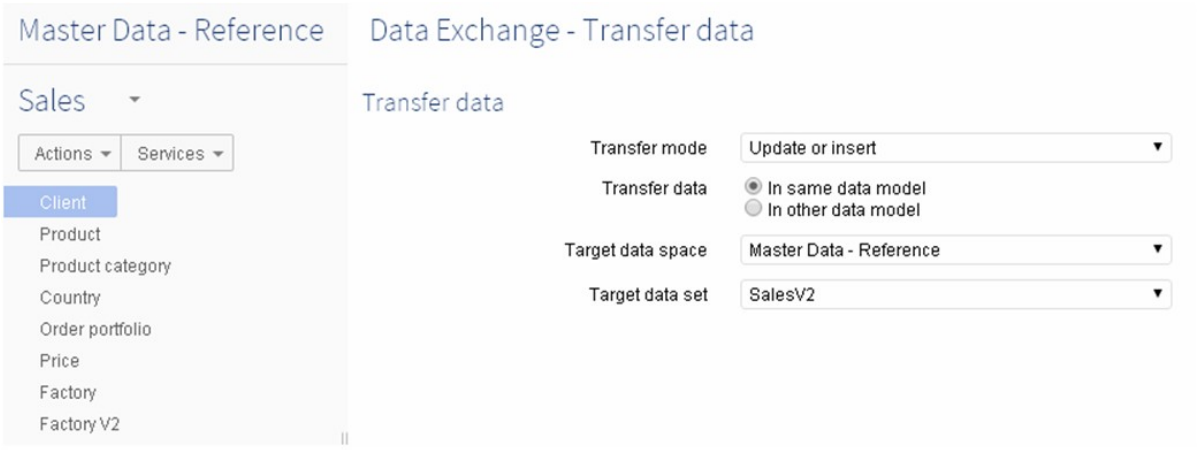
Data transfer is used to copy data from one EBX table to another. If the tables share the same data model, the add-on automatically manages the transfer. However, transferring data between tables that don't share the same data model requires a bespoke data mapping configuration.

2.3.1. Data transfer within tables sharing the same data model

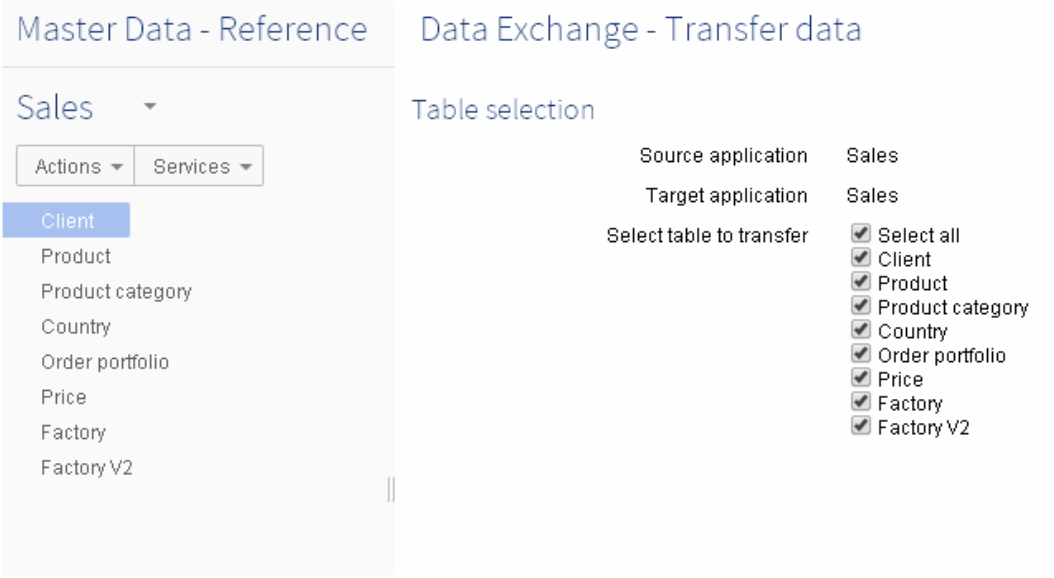
The data transfer is executed at the data set level. This example uses the 'Sales' data set:



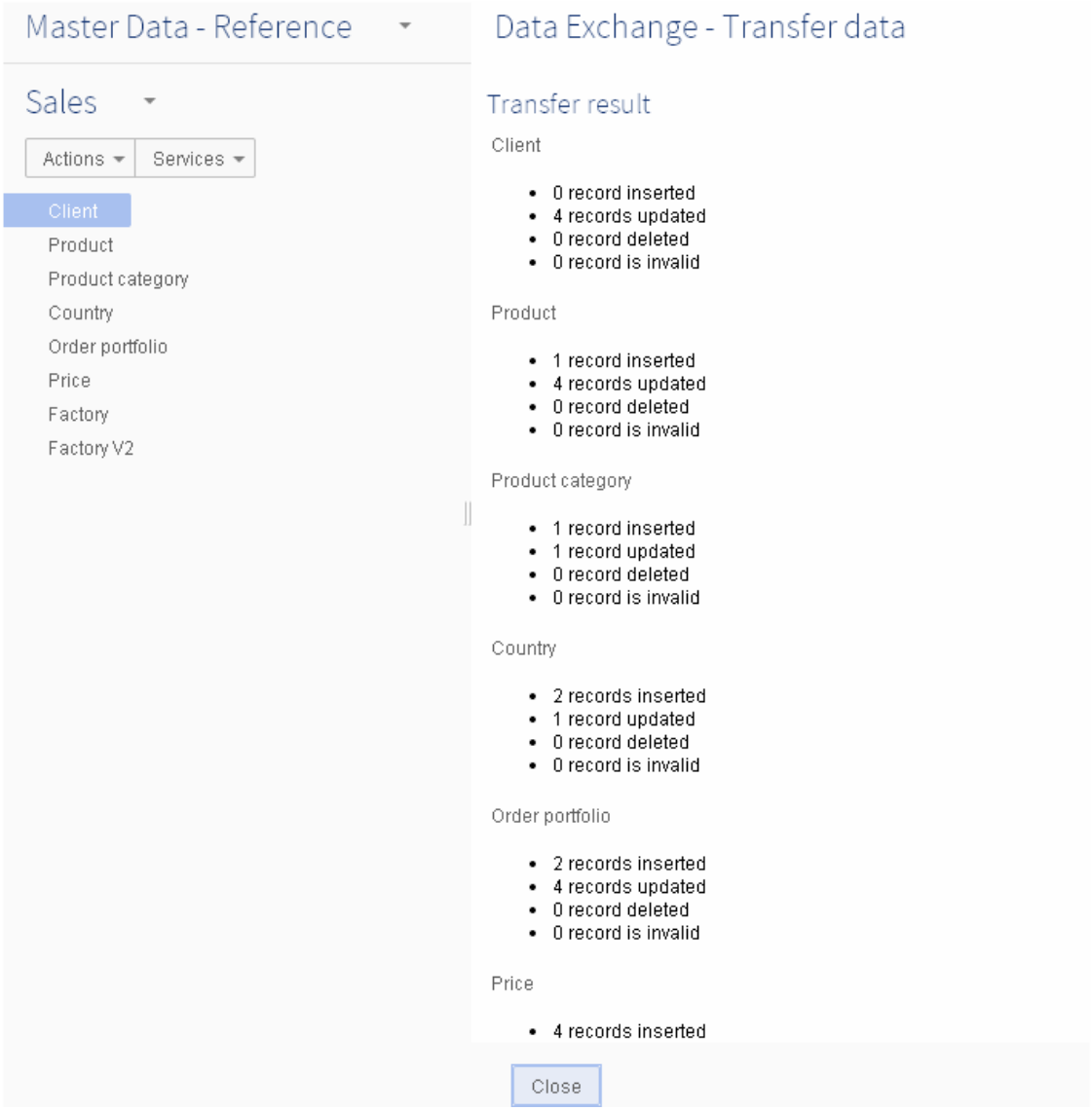
The 'In same data model' option allows you to get all the data spaces and data sets that are based on the same data model as the 'Sales' data set. By selecting this option, data transfer is fully managed by the add-on without requiring any user-defined data mapping configuration.



The data transfer can be done for one or several tables in the data set. You can select which tables to transfer as illustrated below:



This overview displays the data transfer result:

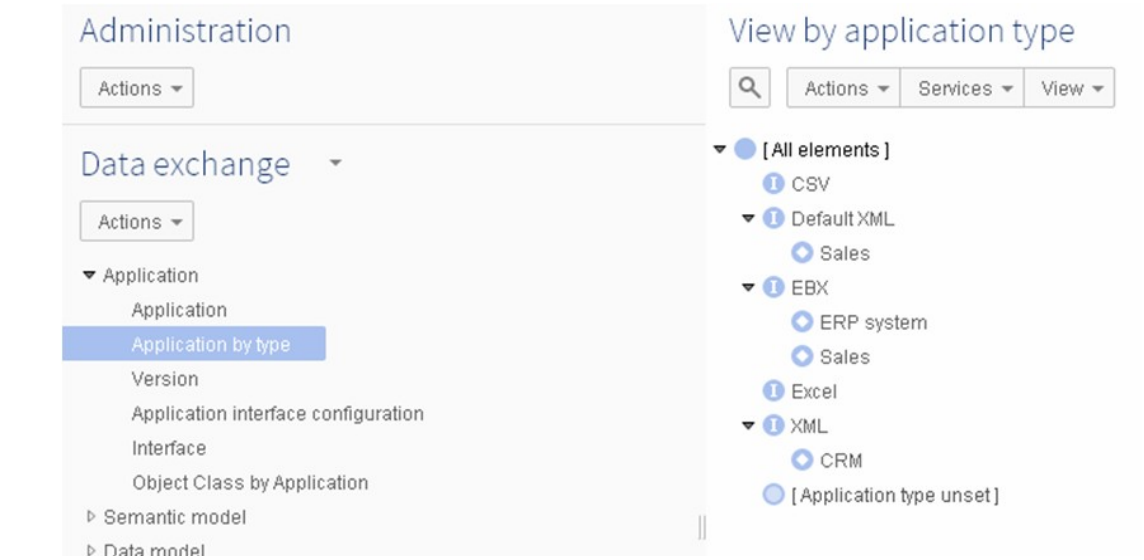


During the transfer process, the add-on does not generate any data mapping configuration in the repository.

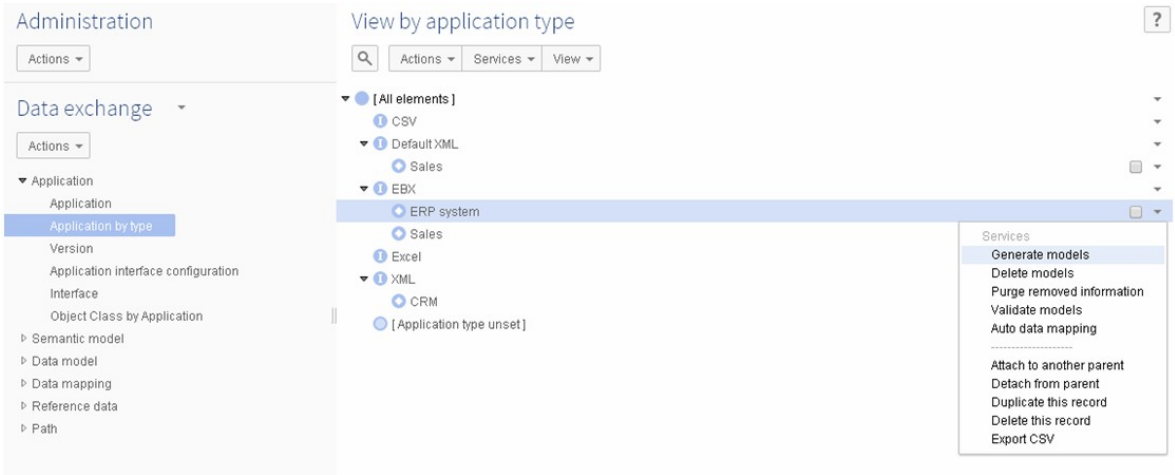
2.3.2. Data transfer within tables based on different data models

When the data transfer is applied between two different data models, a user-defined configuration in Data Exchange must be done, as described below.

First of all, the EBX type applications must be declared. The 'Sales' application has already been configured automatically during the previous XML data flow operations. Now, the new 'ERP system' XML type application is declared.

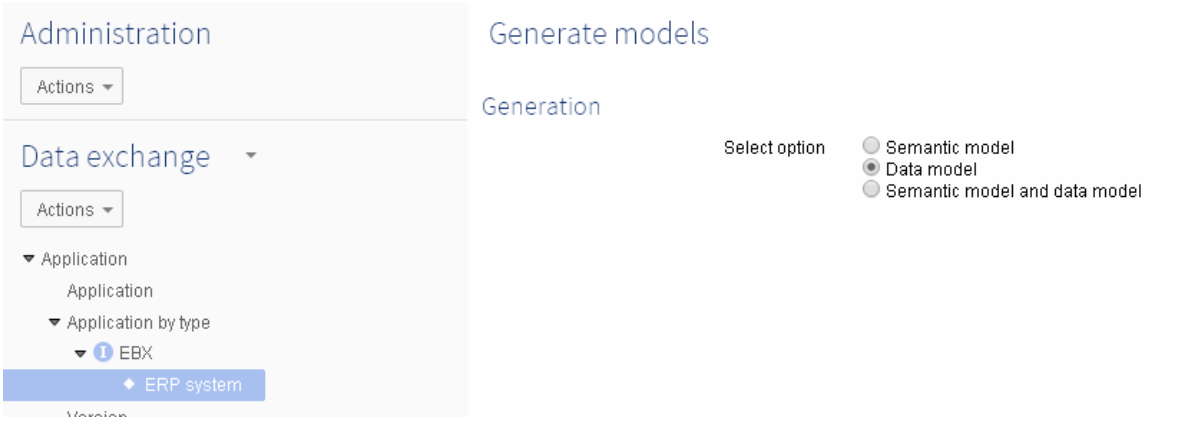


To automatically configure this new application's tables and fields, the 'Generate models' service is used:



The 'Generate models' service allows you to automatically retrieve the data model (tables and fields) configuration and the semantic model (Object class and Property).

In the current version of the add-on, the semantic data model makes the auto-data mapping between two different EBX data models possible.



An overview of the service result displays, as illustrated below:

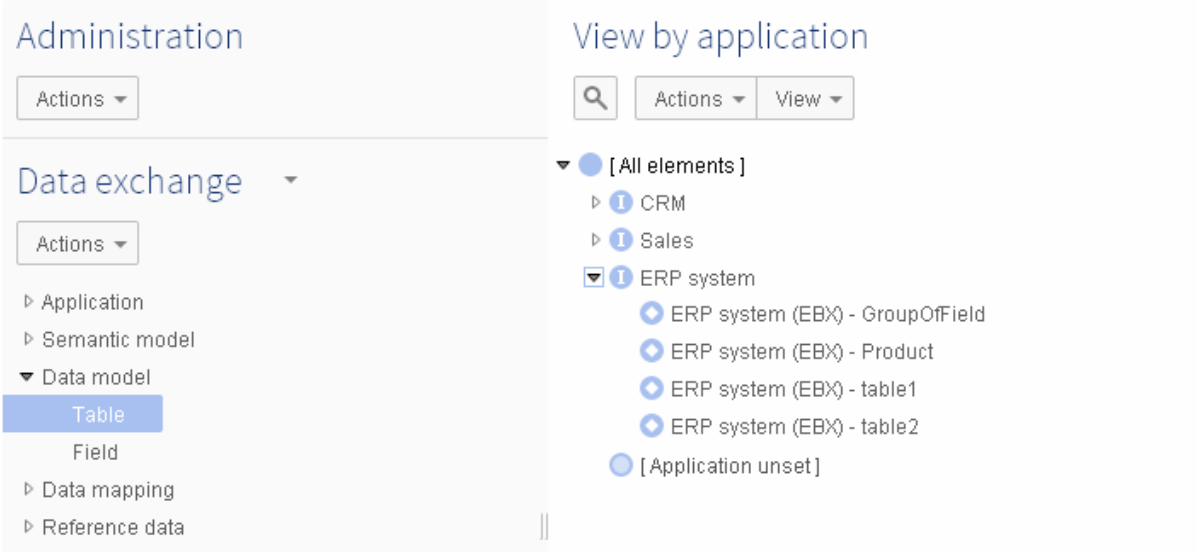
Generation report

Generated on: Master Data - Reference > ERP system
Done on: 09/08/2014 12:04:39

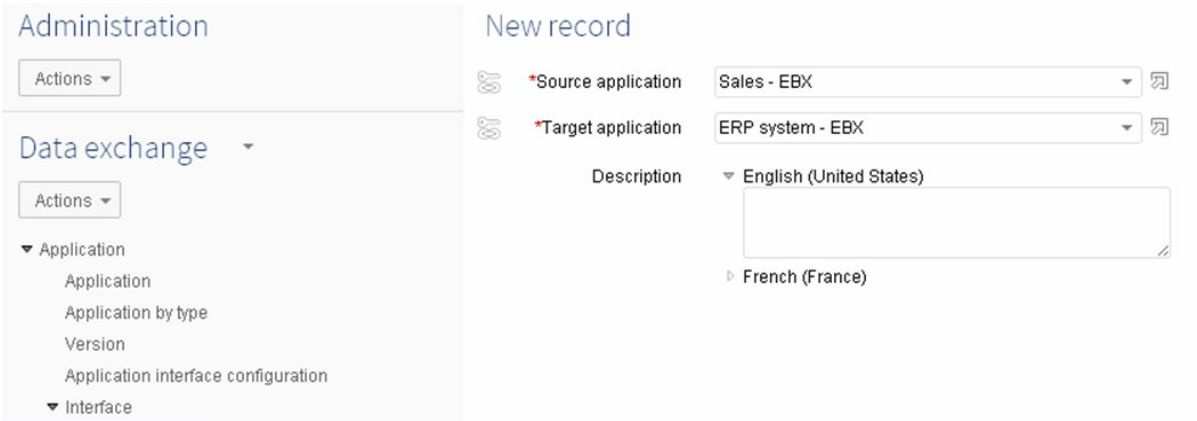
Total number of processed records: 34

Tables	Created record(s)	Updated record(s)	Invalid record(s)
Application			
Object Class by Application	0	0	0
Semantic model			
Object Class	0	0	0
Property by Object Class	0	0	0
Property	0	0	0
Data model			
Table	0	3	1
Field	0	13	0
Path			
Path	0	16	1

Then, the tables are declared automatically for the 'ERP system' EBX type application:

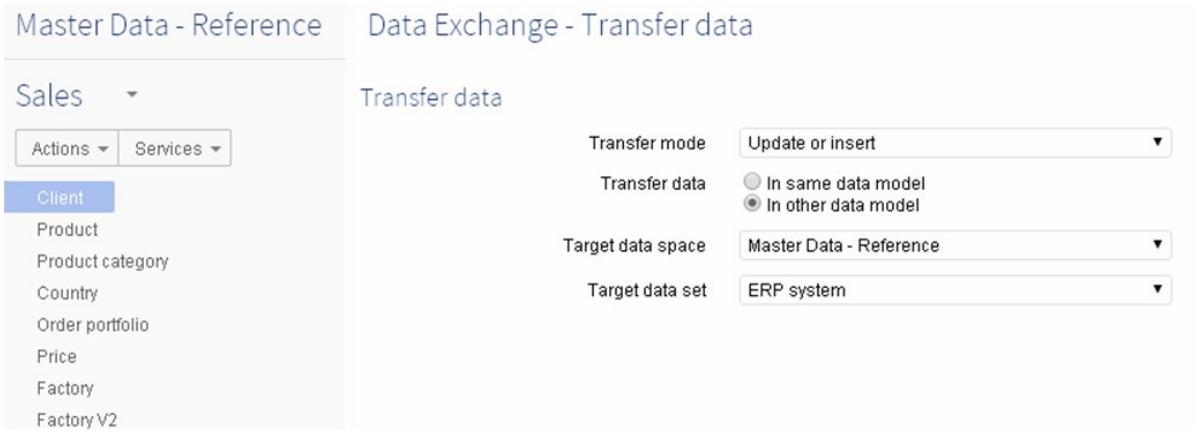


It is now possible to configure the data transfer between the 'Sales' source application and the 'ERP system' target application. To declare this transfer process, an 'Interface' is created as illustrated below:



The table mapping and field mapping use the same procedure already described for the XML data flow for configuration.

Based on this configuration, data transfer can be performed between the two applications relying on different data models as illustrated below.



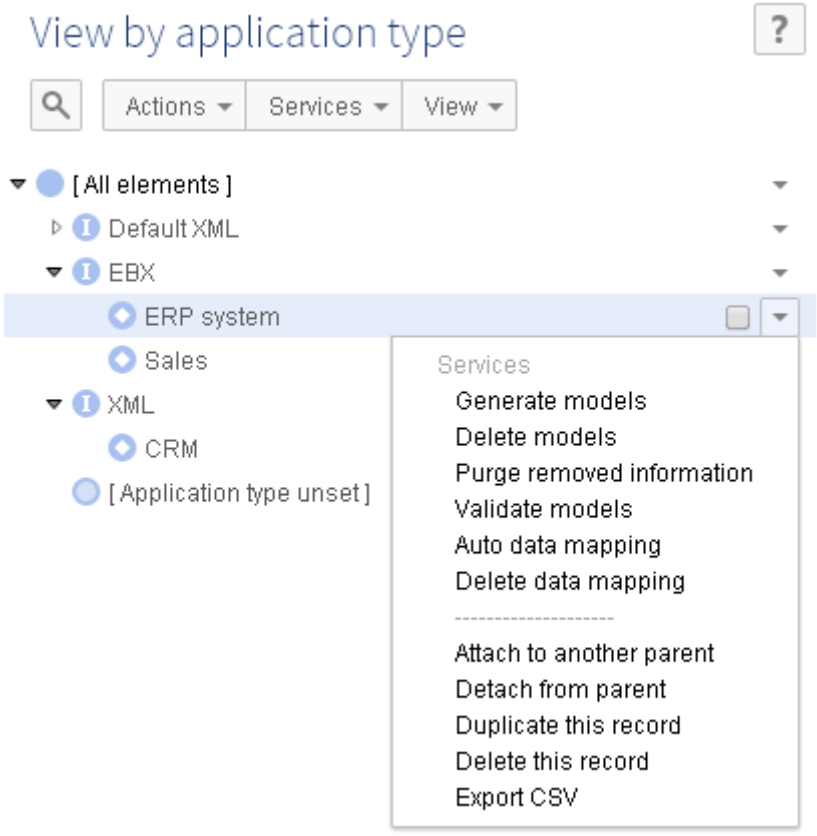
2.4. Excel and CSV data flows

The Excel and CSV export and import processes are based on the previous version of Data Exchange, called 'Add-on for Microsoft® Excel'. You can refer directly to the online help for this part of the add-on.

In the current version of Data Exchange Add-on, the user-defined configuration is not used for the Excel and CSV data flows.

3. Advanced operations

In order to manage user-defined data mapping configurations, a portfolio of services is available depending on the application type. These services are located on the 'Application by type' table in the 'Data exchange' data space under the EXB5 'Administration' tab.



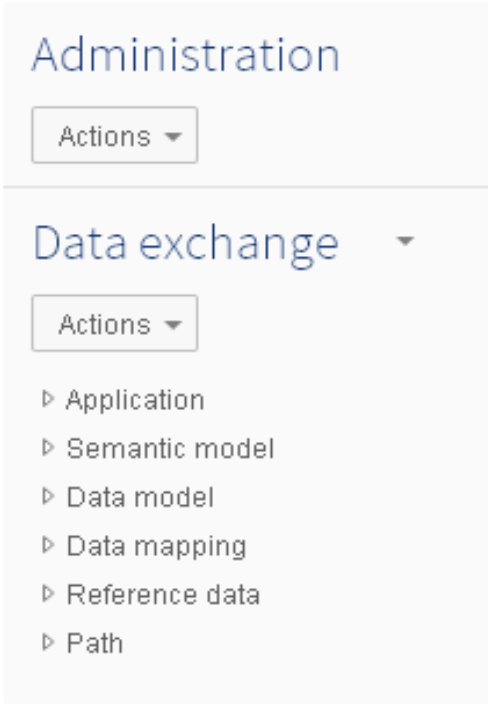
The creation of user-defined data mapping configurations requires IT skills. Based on these configurations, end-users can export, import and transfer the data easily, all the technical aspects of data mapping configuration are hidden from them.

The table below gives a short description of the services available to manage the user-defined data mapping configurations.

Services applied to an Application of type ->	Default XML	XML	EBX
Delete data mapping	Physical deletion of all Tables, Fields and related data mapping (Tables: Version, Interface Application, Table mapping, Field mapping, Field mapping transformation and Path) for the Application. For an EBX type application, the Object class, Property and related tables (Object Class by Application, Property by Object Class) are also removed.		
Generate models	N/A	Tables and Fields are declared as XML Tags. The generation is based on a sample XML file that must be provided as input data.	Tables and Fields are declared as an EBX path. Object classes and Properties can also be generated to get the semantic data model. The generation is based on a data space – data set corresponding to the Application.
Delete models	N/A	N/A	Logical deletion of Object class and Property type items–if not used by another application. Logical deletion of all application Tables and Fields. The logical deletion is registered by using the property 'Is removed'
Purge removed information	N/A	N/A	Physical deletion of the items that are tagged 'Is removed' = 'Yes' by the 'Delete models' service. All data mappings that refer to the deleted items are also physically removed.
Validate models	N/A	N/A	Checks if the configuration is still updated with the EBX' data model. All unaligned Tables and Fields are then modified by changing the 'Is removed' property into the value 'Yes'
Auto data mapping	N/A	N/A	Automatically configures the data mapping between two EBX applications sharing the same Object class and property items.

4. Configuring Data Exchange

4.1. Global view



The Data Exchange repository for data mapping configurations is located in the 'Data exchange' data set under the EBX5 'Administration' tab. The 'Application' domain contains the portfolio of applications with the interface configuration. An interface is used to configure the source and the target applications involved in a data flow.

The 'Semantic model' domain contains the Object Class and Property items used to link the Tables and Fields to the semantic layer. The Semantic model is not mandatory to configure data mapping. However, it allows you to improve data lineage and integration with Information governance. The 'Data model domain' contains the configuration for Tables and Fields. The 'Data mapping' domain contains the actual data mapping configuration applied at the table level, and then the field level.

4.2. Application

The Application domain contains the declaration of every application that is involved in the data flow (import, export, transfer) managed by Data Exchange.

4.2.1. Application

An application is used as a source and/or target in the execution of a data flow process (import, export or data transfer).

Property	Definition
Universal name	Any naming convention is valid.
Logical name	The logical name is automatically provided by the add-on to ensure a unique identification.
Code	Any naming convention is valid.
Last modification date	Date of the last modification applied to the application.

4.2.2 Application by type

An application can be associated with one or many of the following types: CSV, Excel, Default XML, XML and EBX.

Property	Definition
Application	The reference to an application.
Application type	The type of the application.
Application path	When this is an EBX type application, the path gives the data space and data set.

4.2.3. Version

An application can be associated to a version. This version is used as documentation included in the exported data file when the targeted format can integrate it (this is the case in XML as appinfo tag). It has no impact on the data mapping configuration management.

Property	Definition
Code	Any naming convention is valid.
Version	Any naming convention is valid.
Application by type	The reference to an application with its type.

4.2.4. Interface

An interface allows data to flow between two applications.

Property	Definition
Source application	Reference to the application with its type playing the role of the source.
Target application	Reference to the application with its type playing the role of the target.
Description	Description of the interface.

4.2.5. Object class by Application

An application can hold one to many Object Class items. You can use the Object Class to create the relationship between a Table in the Data model and its related item in the Semantic model. Several tables can be linked to the same Object Class.

Property	Definition
Application	Reference to an application.
Object class	Reference to an Object Class.
Description	Description of the relation between the application and the Object Class.

4.3. Semantic model

The Semantic model provides a business data architecture based on Object Class and Property items. An Object Class can be linked to one or many tables. A Property can be linked to one or many fields.

4.3.1. Object class

An Object class is a business concept that can be linked to one or many tables declared in the data models.

Property	Definition
Code	Any naming convention is valid.
Name	Any naming convention is valid.
Is removed	If set to True: The Object Class is no longer valid. It is logically removed. By using the purge service, the data will be physically deleted. If set to False: The Object Class is valid.
Description	Description of the Object Class.

4.3.2. Property by Object Class

A property is held by one or many Object Class items.

Property	Definition
Object Class	Reference to an Object Class.
Property	Reference to a Property held by the referenced Object Class.
Description	Description of the association between the Object Class and the Property.

4.3.3. Property

A Property is a business concept that can be linked to one to many fields declared in the data models.

Property	Definition
Code	Any naming convention is valid.
Name	Any naming convention is valid.
Is removed	If set to True: The Property is no longer valid. It is logically removed. By using the purge service, the data will be physically deleted. If set to False: The Property is valid.
Description	Description of the Property.

4.4. Data model

The Data model provides a logical data architecture based on Table and Field items. The data mapping configuration is also based on these items.

4.4.1. Table

A table is a container of data that depends on the application type. For EBX type applications, it is directly a table in EBX. For XML type applications, it is a node in the XML file.

Property	Definition
Code	Any naming convention is valid.
Application	Reference to the application that owns the table.
Application type	Type of the application.
Object Class	The table can be linked to an Object Class. This is the way to make an association between the logical data architecture and the business architecture.
Name	Logical name of the table.
Label	Label of the table.
Path	The path of the table.
Is removed	If set to True: The Table is no longer valid. It is logically removed. By using the purge service, data will be physically deleted If set to False: The Table is valid.
Description	Description of the table.

4.4.2. Field

A field is held by one Table only.

Property	Definition
Code	Any naming convention is valid.
Table	Reference to the table that owns the field.
Property	The field can be linked to a Property. This is the way to make an association between the logical data architecture and the business architecture.
Name	Logical name of the field.
Label	Label of the field.
Data type	Reference to a Data type for the field.
Order	Order of the field in table. The first position is '0'.
Path	The path of the field.

Is removed	If set to True: The Field is no longer valid. It is logically removed. By using the purge service, data will be physically deleted. If set to False: the Field is valid.
Description	Description of the field.

4.5. Data mapping

This domain contains the data mapping configuration for tables and fields.

4.5.1. Table mapping

The data mapping configuration between the tables.

Property	Definition
Source table	Reference to the table used as the source for the data flow.
Target table	Reference to the table used as the target for the data flow.
Source application	Used to sort the records based on the application name and type. This data is computed automatically from the source table value.

4.5.2. Field mapping

The data mapping configuration between the fields.

Property	Definition
Source field	Reference to the field used as the source for the data flow.
Target field	Reference to the field used as the target for the data flow.
Mapping type	The mapping can be performed in a 'direct' way (the target field is equal to the source field) or based on split, or aggregation policies. In the current version of the add-on, only the direct mapping type is available.
Source application	Used to sort the records based on the application's name and type. This data is computed automatically from the source field value.

4.5.2. Field mapping transformation

The data transformation applied during field data mapping.

Property	Definition
Field mapping	Reference to a Field mapping declaration.
Transformation function	Reference to the transformation function to be applied.
Order	Execution order of the transformation function in case several functions must be applied.

4.6. Reference data

4.6.1. Application type

The current version of the add-on manages the following types: XML, Default XML and EBX.

Property	Definition
Code	Code of the application type.
Name	Name of the application type.

4.6.1. Predefined Application type

The add-on provides these predefined Application types.

Application type	Description
EBX	An EBX type application is referenced through a data set. Its tables and fields are described through an EBX path.
Default XML	A 'Default XML' type application relies on default XML paths automatically created by Data Exchange when an export XML process executes.
XML	An XML type application relies on user-defined XML path declarations.

4.6.2. Data type

This table references all the possible data type of a field.

Property	Definition
Code	Code of the data type.
Name	Name of the data type.

4.6.3. Mapping type

In the current version of the add-on, only the 'Direct' mapping type is available. In future versions, other types will be provided to manage aggregation and split of fields.

Property	Definition
Code	Code of the mapping type.
Name	Name of the mapping type.
Description	Description of the mapping type.

4.6.4. Predefined Mapping type

The add-on provides this predefined Mapping type.

Mapping type	Description
Direct	The mapping between the source and target fields is executed in a direct way: the target value is equal to the source value. There is no aggregation and/or split of fields values.

✕	Aggregation and split of fields values

4.6.5. Transformation function

During data mapping, the data value can be transformed before moving to the target application. In the current version of the add-on, the 'No import' (and 'No export', 'No transfer') transformation function is delivered.

Property	Definition
Code	Code of the transformation function.
Name	Name of the transformation function.
Java class	Java class of the transformation function.
Description	Description of the transformation function.

4.6.6. Predefined Transformation function

The add-on provides these predefined Mapping types.

Transformation function	Description
No export	The source field value is not exported.
No import	The target field value is not imported.
No transfer	The field value is not transferred.

4.7. Path

The path domain contains the declaration of all paths used in the data mapping configurations.

4.7.1. Path

The Path table collects all the path declarations used in the data mapping configuration.

Property	Definition
Application type	A reference to the application type (EBX, Default XML, XML).
Path type	Either Table, Field or Application.
Path	The path value.
Comment	Description of the path.

5. Known limitations

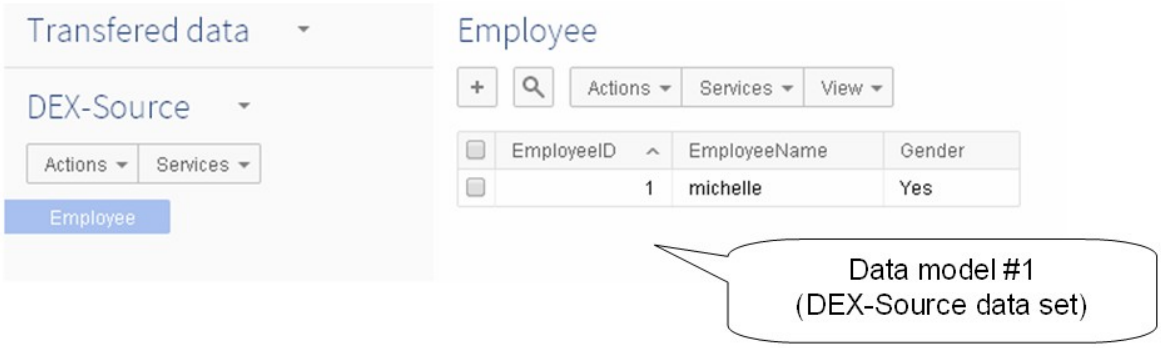
- Multi-occurs complex data type are not supported (will be fixed in further release).
- Association object and selection node are not supported.
- User defined attribute (UDA) are not supported for the export and import processes.
- Split and aggregation of fields (will be added in further release).
- Transformation of fields (will be added in further release).
- Data lineage (will be added in further release).
- Graphical view of the data mapping configurations (will be added in further release).

Appendix

Appendix 1 – Automatic data mapping for transferring data

This appendix presents a use case about the automatic data mapping feature. The user wants to transfer data between two tables in two different applications relying on different data models (DEX-Source and DEX-Target).

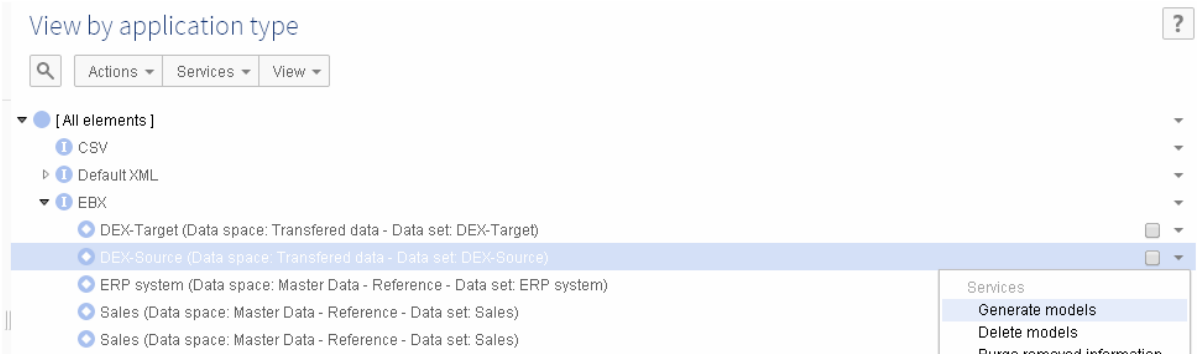
Since the source and target applications are not based on the same data model, a configuration must be declared in Data exchange.



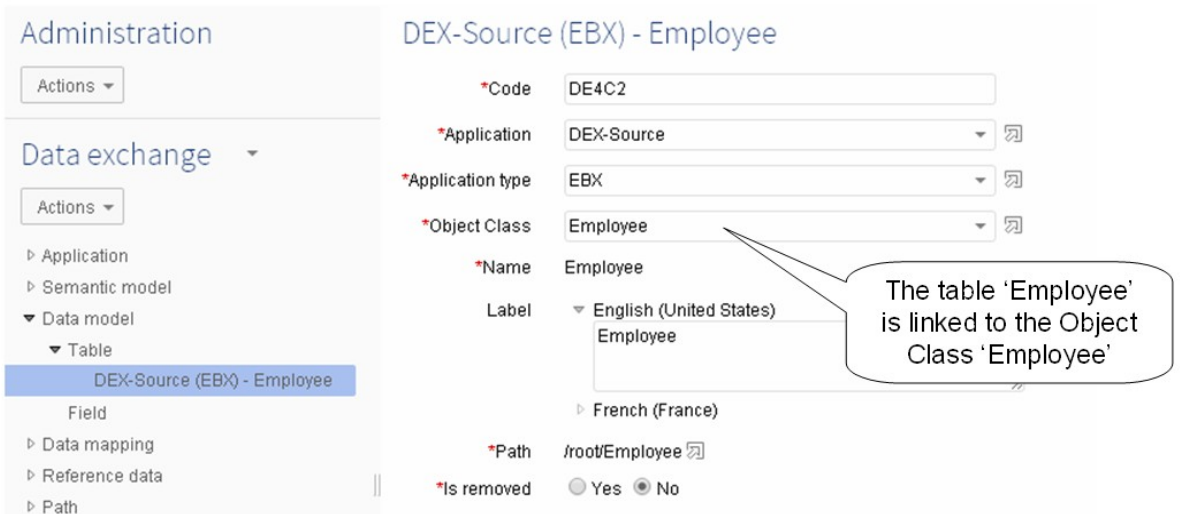
To facilitate the data mapping configuration, one of this application is used as the pivot (DEX-Source) to generate the semantic and data model.

The generated Tables and Fields are linked to Object Class and Properties respectively of the source application.

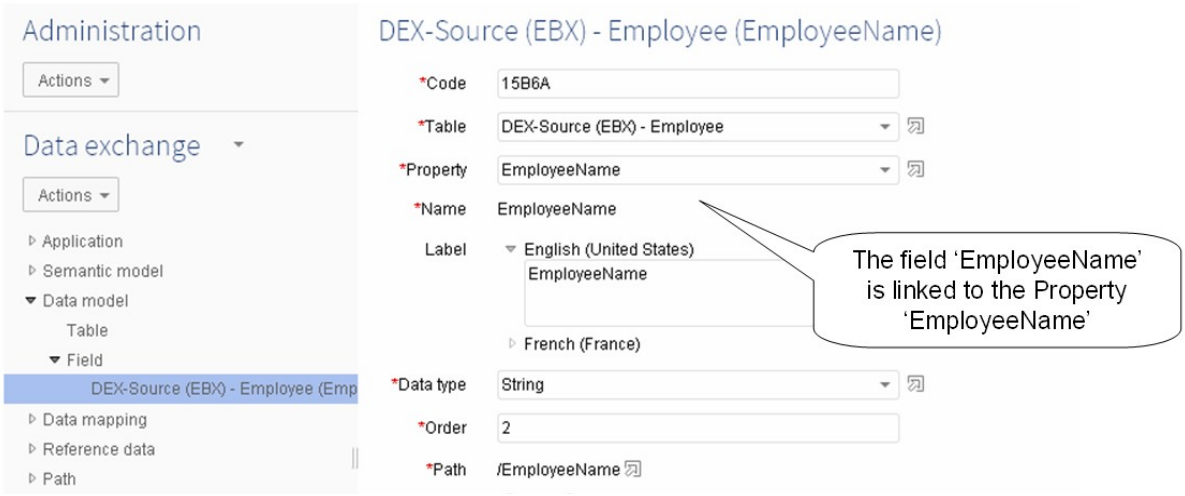
Execution of the service 'Generate models' on the application 'DEX Source'.



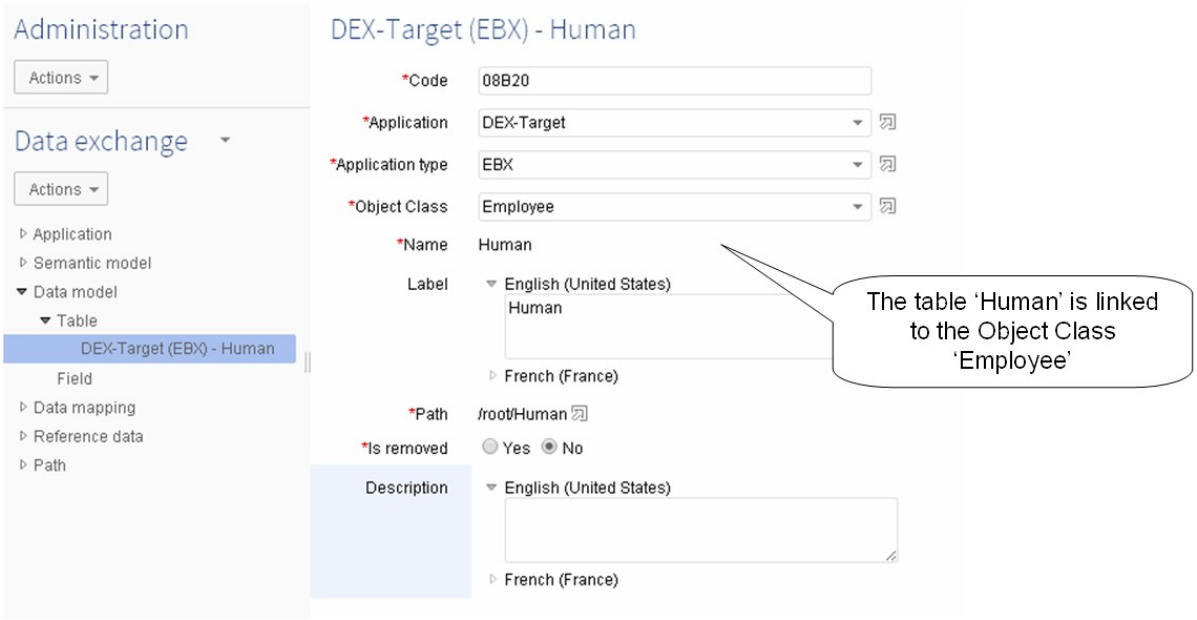
The Table 'Employee' is then linked to the Object Class 'Employee'.

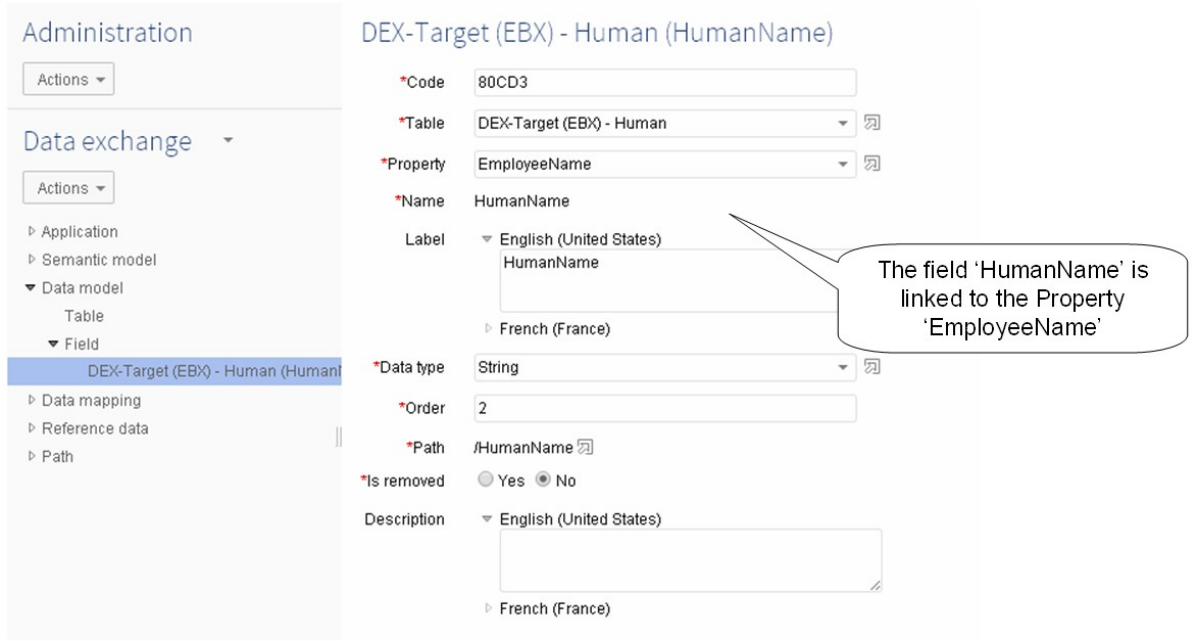


Each field of the table 'Employee' is linked to its related Property. Below the example of the field 'EmployeeName'.

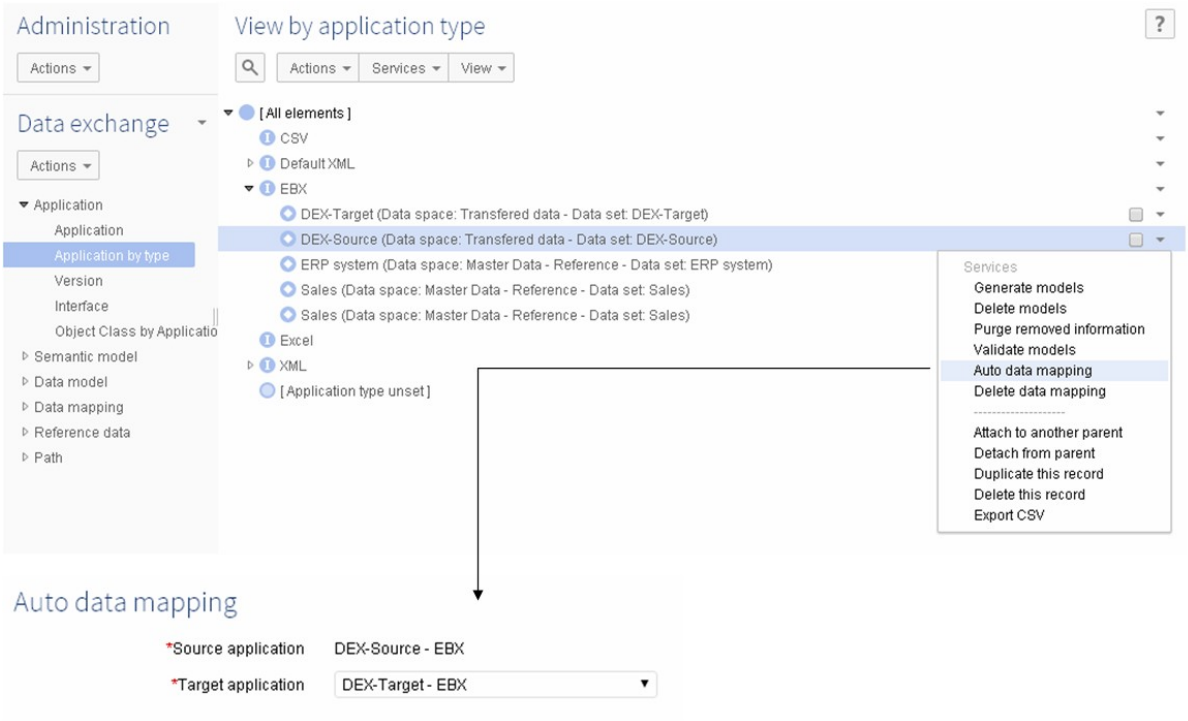


The table and fields of the target application are manually linked to the same Object Class and Properties of the source application.

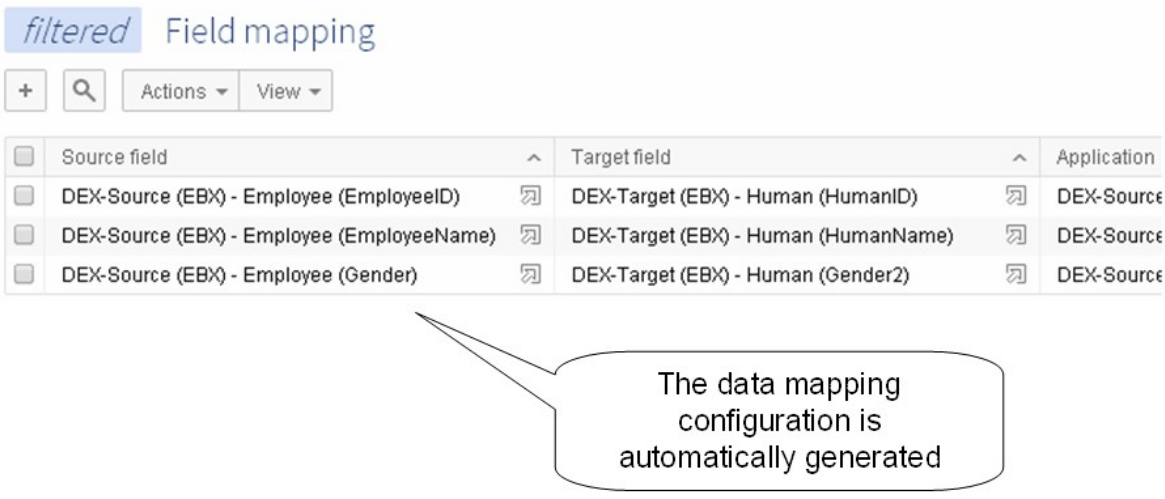




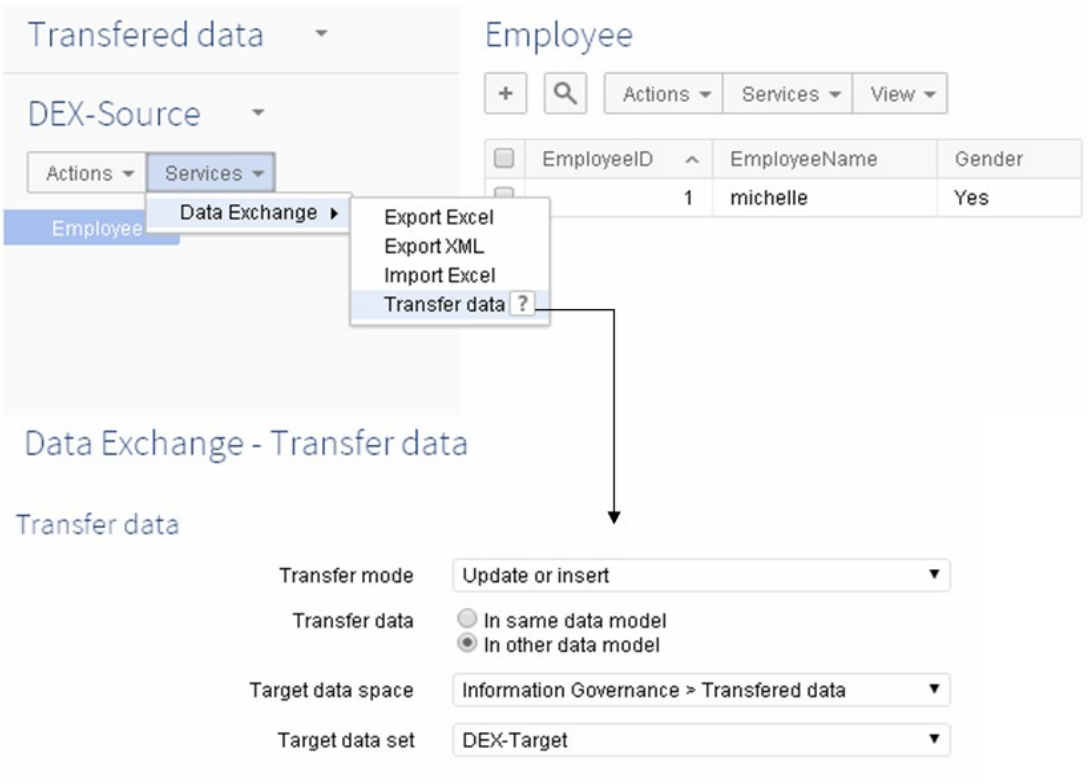
Once the target application has been configured to share the same Object Class and Properties of the source application, the automatic generation of the data mapping is executed and the data transfer becomes available.



The result of the 'Auto data mapping' is illustrated below. It is used to make possible the data transfer from the source application 'DEX-Source' to the target 'DEX-Target'.



On the data set of the source application, the 'Transfer data' service allows you to transfer the data from the DEX-Source to the DEX-Target directly.



Then the target table 'Human' is fed with the source table 'Employee'.