M3 Enterprise Collaborator ION Mapper User Guide

Version 10.4.2.0
Published February 6, 2014

Important Notices

The material contained in this publication (including any supplementary information) constitutes and contains confidential and proprietary information of Infor.

By gaining access to the attached, you acknowledge and agree that the material (including any modification, translation or adaptation of the material) and all copyright, trade secrets and all other right, title and interest therein, are the sole property of Infor and that you shall not gain right, title or interest in the material (including any modification, translation or adaptation of the material) by virtue of your review thereof other than the non-exclusive right to use the material solely in connection with and the furtherance of your license and use of software made available to your company from Infor pursuant to a separate agreement, the terms of which separate agreement shall govern your use of this material and all supplemental related materials ("Purpose").

In addition, by accessing the enclosed material, you acknowledge and agree that you are required to maintain such material in strict confidence and that your use of such material is limited to the Purpose described above. Although Infor has taken due care to ensure that the material included in this publication is accurate and complete, Infor cannot warrant that the information contained in this publication is complete, does not contain typographical or other errors, or will meet your specific requirements. As such, Infor does not assume and hereby disclaims all liability, consequential or otherwise, for any loss or damage to any person or entity which is caused by or relates to errors or omissions in this publication (including any supplementary information), whether such errors or omissions result from negligence, accident or any other cause.

Without limitation, U.S. export control laws and other applicable export and import laws govern your use of this material and you will neither export or re-export, directly or indirectly, this material nor any related materials or supplemental information in violation of such laws, or use such materials for any purpose prohibited by such laws.

Trademark Acknowledgements

Publication Information

Release: 10.4.2.0

Publication date: February 6, 2014

Document Number: MECMMUG_10.4.2.0_W_01

Version Log

The version log describes the changes between versions of this document.

Part Number	Release Date	Description
MECMMUG-10420W-01	201402	Updated for ION Mapper version 10.4.2.0 Updated topics and added the following: Deploy Multiple Mappings Deploy Mappings Searching in a Mapping XML Schema Basics for Wildcard Elements Mapper terminologies Wild Card Elements terminologies Palette Behavior
MECMMUG-10410W-01	201308	Updated for ION Mapper version 10.4.1.0 Added the following: Mapping Lifecycle M3 Company and Division Restructure Function
MECMMUG-104W-01	201305	Updated for ION Mapper version 10.4.0.0 Added the following: Schema Requirements, Editing Schemas,Setting the XML Document Editor Updated views, reorganized topics, and examples.
MECMMUG-103W-01	201208	Updated for version 10.3.0.0 Updated MEC node names. Message Validation Navigating Around Mappings Filtering options , Function Elements Overview Updated topics : Displaying the Mapping Console View Changing Documents and Root Element Added new procedures: Saving a Mapping

Part Number	Release Date	Description
		Deleting Mappings Creating and Adding Repository Function Deleting a Repository Function Exporting Mappings Broken Links
MECMMUG-92W-01	201201	Updated for version 9.2.0.0 Eclipse plugin
MECMMUG-91W-01	201105	Updated for version 9.1.4.0
MECMMUG-91W-01	201005	Updated for version 9.1.3.1

Contents

ION Mapper Overview

1

This chapter provides an overview of the functions, usage, and management of ION Mapper.

About this Guide

This user guide describes how to use and manage the ION Mapper client tool.

The ION Mapper client tool is a component within the MEC product. For more information about MEC, see the *M3 Enterprise Collaborator Server and Client Tools Installation Guide*.

Users of this Guide

This user guide is intended for use by ION Mapper client tool business consultants, application engineers, and Java developers to create mappings for transformation of XML documents on the run-time server. These transformations can be available for use through the ION Desk and MEC.

This user guide is not intended to be used as a complete self-study guide on how to learn to use the tool and all its concepts. It is expected of the reader to have sufficient back-ground knowledge acquired elsewhere.

Knowledge prerequisite

To use this product, you should be knowledgeable and experienced in the following areas:

- Strong understanding of ION Mapping and its prerequisites.
- Programming concepts, such as functions and input/output parameters, loops, and execution flow control.
- XML concepts such as elements, attributes, namespaces, and schemas.

- Java programming.
- Eclipse IDE Framework.

Documentation conventions

This document uses specific text conventions and terminologies. For a list of terms and definitions, see the Appendix section.

Conventions used

Naming

When prompted for names do not use white spaces or quotation characters (single, double, or curly) within names.

Naming Example:

- Acceptable: **`MyMapping`**, **`my_mapping`**, **`myMapping`**
- Not acceptable: **`My Mapping, Peter's Mapping`**

Selection

A selected mapping element is indicated by a target icon or a dashed border.

Search Filter

Some tasks in this tool use wizard pages or views with **Search** filter option.

When you search for document elements you have two options, either you type a name that will locate all element that matches that name, or you type a path on the **`/A/B`** format (as denoted by the "A/B = path" expression before the filter box).

The latter option, for example "**`/SyncCreditTranfer/DataArea`**", will locate all elements under that path, for example "**`/SyncCreditTranfer/DataArea/AnElement`**" and "**`/SyncCreditTranfer/DataArea/AnotherElement`**".

Search filter guideline:

- To display all data, leave the filter field blank and press Enter.
- Use an asterisk (*) character in place of a sequence of unspecified characters, for example **`MyMapp*`**.
- For filters used together with a multi-column list the filter is applied across columns. For example, to see all mappings containing the phrase **`AutoBank`** in the **`PBM`** category, type the expression **`AutoBank*PBM`**.

Name	Version	Category	Id
AutoBank_XYZ	1.0	PBM	137
AutoBank_QRT	1.0	PBM	149
MyMapping	1.0	TEST	2

Progress and validation

The status of actions writing output to the Mapping Console view is shown using a traffic light icon, listed here with their corresponding states:

- **Red light** - The last command failed.
- **Yellow light** - The command is running.
- **Green light** - The last command was successful.

For example, when publishing a mapping:

- While being published by the mapper server, the traffic light is temporarily **yellow**.
- If successfully published, the traffic light then turns **green**.
- If not successfully published, the traffic light turns **red**.

The state of the traffic light is maintained until you run the next command. The traffic light always reflects what is being logged in the Mapping Console view. For example, if you run the command to clear the console, the traffic light will always be green because the clear command is always successful.

There are **Validation** and **Broken Links** tabs that shows the result for a document or a mapping validations. If issues are detected, a table listing the category label and detailed descriptions are shown. Select an issue from the table list to display more specific details about that issue.

ION Mapper Overview

ION Mapper is used to create and deploy mappings. On a high-level, a mapping is a description of how to transform (translate) messages, or documents, sent between internal and external resources. On a detailed level, a mapping is an XML file expressing a transformation between input and output elements and attribute structures defined by XML schemas representing the input/output messages and documents. ION Mapper provides the capabilities to define mappings, for example, for OAGIS BODs, EDI, EBS, financial messages etc.

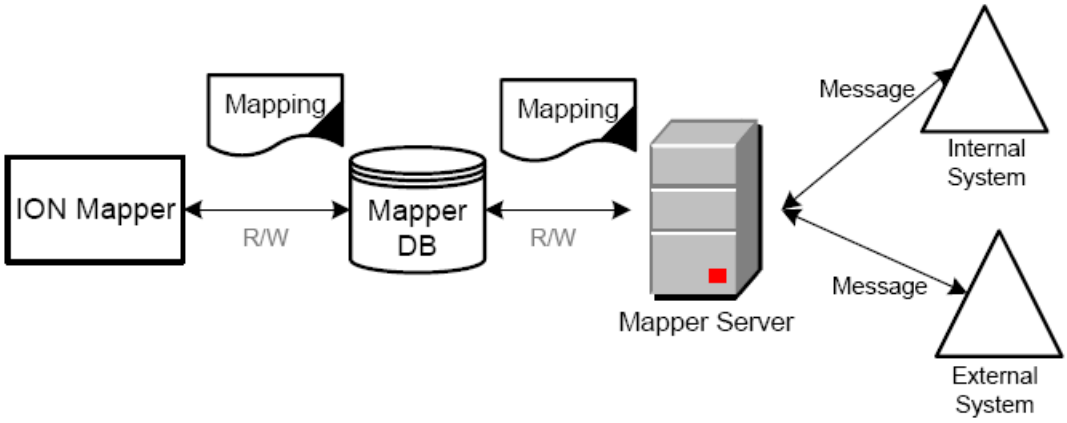
Mappings are organized in projects. These projects can contain new or imported mappings. A graphical editor, called Mapping Editor, is used to define the transformation.

In the Mapping Editor the left and right panes display the elements and attributes of the input (left) and output (right) schemas as tree structures. The middle pane shows the transformation expressed as a

sequence of connected mapping elements that are categorized as functions, variables, or constants. An empty mapping has a blank middle pane.

After a mapping is completed you deploy it to the Mapper Server run-time environment. Deployment is the method of storing a mapping in a Mapper Database accessed by a Mapper Server. The Mapper Server uses the mappings in the database for message translation.

The following diagram shows the interaction among ION Mapper, database, server, and messages.



In this scenario the ION Mapper can help you achieve two tasks:

1 Create mappings.

- Select predefined functions, or create new user-defined functions, to map elements and attributes in the input schema to a number of input parameters in the functions.

  The output from the functions is mapped to the elements and attributes according to the output schema.

- A completed mapping is saved as an XML.

  The file contains all the information needed by the Mapper Server run-time environment.

2 Deploy mappings to the Mapper Server.

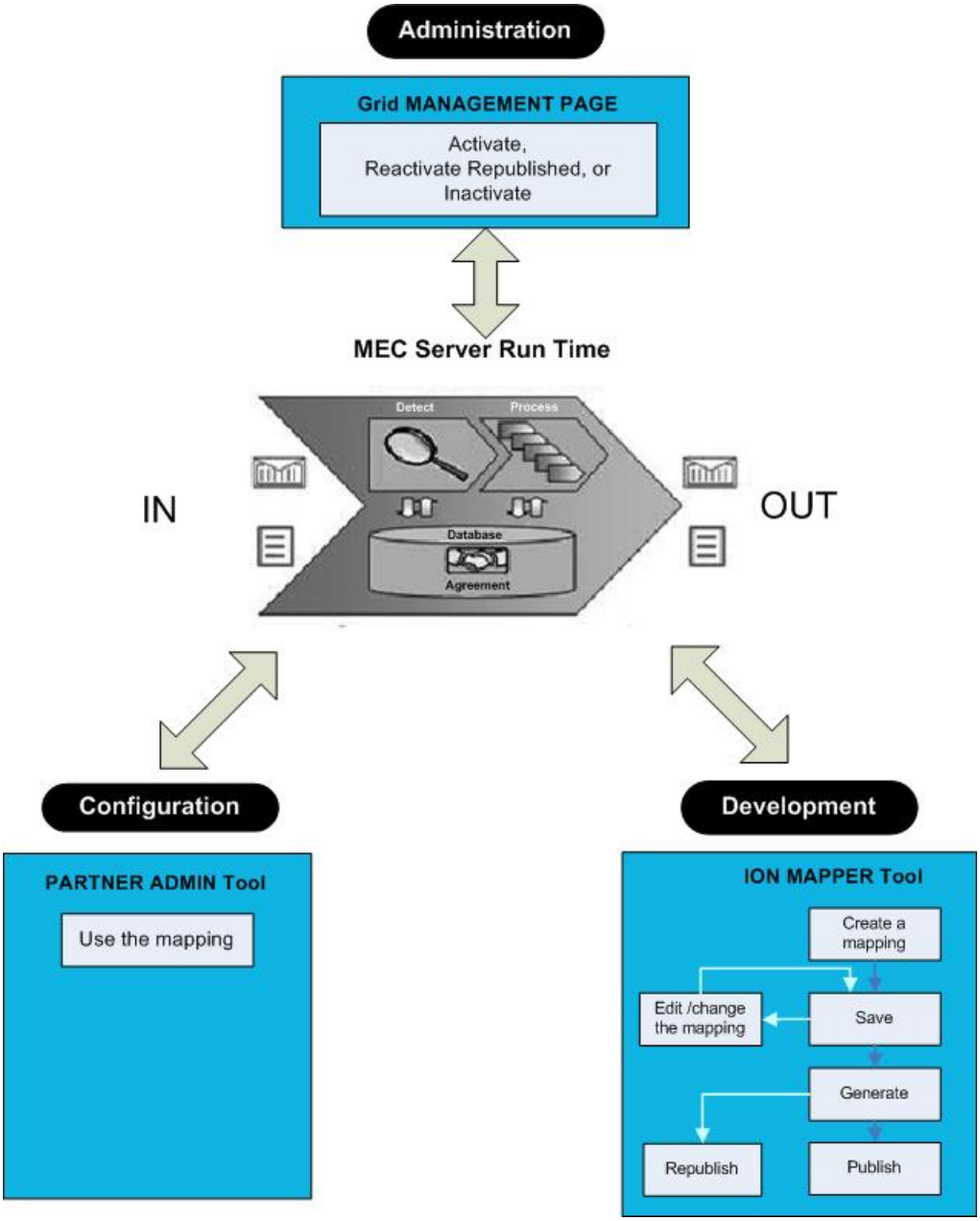
- Deployment is done by storing the mapping in a Mapper Database.

  Saved user-defined functions can be reused in multiple mappings through import.

Mapping Lifecycle

These concepts will help you understand the mapping generation, publish, and activation lifecycle of a mapper.

On an overview, the diagram here shows that you use ION Mapper to create a mapping, activate it in grid Management Page, and then use in Partner Admin. You should first activate the mapping and then reload the cache to activate the Partner Agreement using the mapping.



Process Step	Details
1. **Create** new mapping	• Create a new mapping in ION Mapper or import an existing mapping, either from a (.zap) file or from a mapper database. • The mapping now exists as files in your Eclipse workspace.
2. **Save** the mapping to the mapper database.	• The mapping now exists as metadata in the mapper database. • At this stage, if this is a new mapping, no runtime mapping class exists.

3. **Generate** the mapping.
This step is optional.
	• A runtime mapping class is generated and compiled, but the runtime mapping class is not inserted into the mapper database. A Java file is written to the file system, though.
	• **MEC data translations** are parsed, but **metadata** is not inserted or updated in the mapper database.
	Note: This step is for testing purposes only.
	  • That is, to verify that the mapping can be generated and compiled, and that MEC data translations are correctly defined.
	  • This step does not affect runtime, however note that the mapping metadata in the mapper database is updated.

4. **Publish** the mapping.	• A runtime mapping class is generated and compiled. • The runtime mapping class is inserted into the mapper database as **Current Source**. • A **Java file** is also written to the file system. • **MEC data translations** are parsed and **metadata** is inserted or updated in the mapper database. The mapping is now in the Inactive state and ready for activation

Process Step	Details
5. **Activate** the mapping	**1** Access Grid > (MEC) Application Management Page > Server > Mappings. **2** Click "**Activate**" for the mapping. • The **runtime mapping class** will now be compiled and put into the classpath. • The **mapping** is now ready to use in Partner Agreements. **Important:** A mapping cannot be used before it is activated. **Error messages** • **`NoClassFoundException`** This error is thrown if a Partner Agreement uses an inactive mapping in runtime. Activate the mapping and then reload the cache to activate the Partner Agreement using the mapping. • **`Unable to find source code for mapping id '<id>'`** This error is thrown if the mapping used by the XML Transform process in the agreement is published but is not activated. Activate the mapping and re-run the message.
6.**Update the mapping** in ION Mapper.	• Update the mapping in ION Mapper. • The updated mapping now exists as files in your Eclipse workspace. **Note:** If you change the mapping's version number the updated mapping will be handled in the same manner as a new mapping, see step one.
7. **Save the updated mapping** to the mapper database.	• The updated mapping metadata is now stored in the mapper database, but the runtime mapping class is still generated from the old **Current Source**.
8. **Generate the updated mapping**.	This step is optional, see step three.

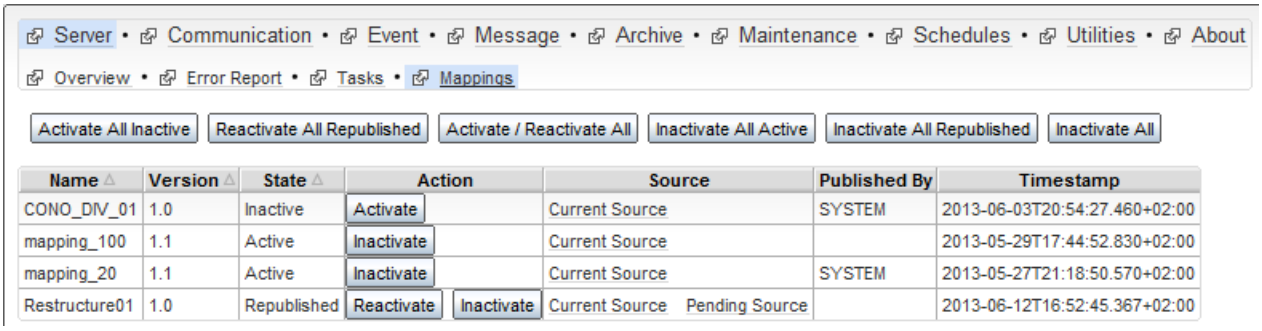
Process Step	Details
9. **Republish** the mapping.	• A new runtime class is generated and compiled. If the mapping is in **Active** or **Republish** state: • The **new runtime mapping class** is inserted into the mapper database as **Pending Source**. • The **Current Source** is not overwritten. **Note:** It is still the **Current Source** that is used in runtime. Even if you restart MEC, or if new MEC Server nodes are started, **Current Source** is used. To use the **Pending Source**, you need to reactivate the mapping. The following message will be given in the Mapping Console in ION Mapper when (re)publishing an active mapping: `"Mapping 'mapping_name' version mapping_version is currently active, your changes will not take effect until the mapping is reactivated"` If the current mapping is in **Inactive** state: • The **new runtime mapping class** is inserted into the mapper database as **Current Source.** • The previous **Current Source** is overwritten.
10. **Reactivate** the mapping.	**1** Access Grid > (MEC) Application Management Page > Server > Mappings. **2** Click "**Reactivate All Republished**" for the mapping. • The **Pending Source** will now be moved to **Current Source** and, • The **runtime mapping class** will be compiled and replace the **old class** in the classpath. This will happen on all MEC nodes that execute runtime mappings. After this operation, **Pending Source** does not exist as it is moved to **Current Source**. The **mapping** is now in its **Active** state again.

Process Step	Details
11. **Inactivate** the mapping	**Important:** To avoid **`ClassNotFoundException`** to be thrown at runtime, you should remove the mapping from all Partner Agreements and reload the cache before performing this operation. **1** Access Grid > (MEC) Application Management Page > Server > Mappings. **2** Click **Inactivate** for the mapping. • The **runtime mapping class** is now removed from the classpath. • The **mapping** is now in its **Inactive** state, and • The latest source code for the runtime mapping class is stored as **Current Source**.

When restarting MEC, the previously loaded runtime mapping classes will be automatically compiled and added to the classpath. The **Current Source** will always be used when starting MEC.

If a mapping was republished, so that **Pending Source** also exists, you still need to reactivate the mapping in the new MEC session to run the latest version of the mapping source code. To reactivate, a republished mapping always requires an explicit user action.

This screen shot shows the mapping state and available actions in grid **Management Page**.



Under the Action column, select a new state to invoke for the mapper in that row.

On the bar at the top, the options are executed for all, or certain mappings in the table.

Activate

- **Activate All Inactive** - to activate only the mappings in **Inactive** state.

- **Activate/Reactivate All** - to activate all mappings, both when the mapping is inactive and when the mapping is republished, but not yet reactivated.
- **Reactivate All Republished** - to reactivate all republished mappings.

Inactivate

- **Inactivate All Active** - to inactivate only the mappings in **Active** state.
- **Inactivate All** - to inactivate all mappings, both when the mapping is active and when the mapping is republished, but not yet reactivated.
- **Inactivate All Republished** - to inactivate all mappings that are republished and already reactivated.

ION Mapper Configuration and Settings

2

Configuration and Settings Overview

Post installation configurations

Note: For the following configurations, see the "*ION Mapper Configuration and Settings*" section in *M3 Enterprise Collaborator Server and Client Tools Installation Guide*.

- **Memory Settings** - The memory settings helps you o accomplish the important task of ensuring that ION Mapper has sufficient amounts of memory. The maximum memory setting controls the highest memory amount that can be allocated at the start.
- **Security Settings** - Set the user password in order to access a secured connectivity.
- **Server Connectivity** - Set the server connection to be able to generate and publish a mapping to Mapper Server.
- **Database Connectivity** - Set the database connection to be able to store and import mappings to/from a Mapper database.
- **SQL Connectivity** - Set the SQL database connectivity only if the SQL feature will be used, for example, the database, commit, or rollback functions.
- **M3 API Connectivity** - Set the M3 API connectivity only if the MI feature will be used, for example, the M3 API messages in a mapping.

Self diagnose

When starting ION Mapper it automatically performs self diagnosis. It checks for the Java, Eclipse, and GEF expected versions. If an unexpected version is found, a warning message is shown.

ION Mapper configurations

- **Type Definitions** - These are the data types you can use on your functions parameter, variables, and constants.
- **XML Document Editor** - Define an external XML schema editor to use.
- **Import/Export Settings** - Define default folders to use when importing or exporting mappings.

Setting the Type Definitions

Use this procedure to add new or edit the existing type definitions available for parameters, variables, and constants.

1. On Eclipse menu, select Windows > Preferences.
2. Expand the view of ION Mapper and select **Type Definitions**.

   Type definitions and description are displayed in the right pane.
3. Perform any of the following tasks:

Task	Steps
Add new type definitions	**a** Click **Add**. **b** In type Definitions table, find **NewType**. Update the Name and Description field values.
Remove a type definition	**a** Select a Type Definition to delete. **b** Click **Remove**.

4. Click OK to close the Preferences window.

Setting the XML Document Editor

Use this procedure to define an external XML schema editor and be able to open your mapping as an XML document. This is also the recommended way to edit a schema, if needed. For further details around schema editing see section "Schema Editing".

1. On Eclipse menu, select Window > Preferences.
2. Expand the view of ION Mapper and select XML Document Editor.

XML Document Editor details are displayed in the right pane.

3 On Editor field, type the complete path to your installed XML schema editor of choice.

Or, click Browse to navigate to its folder location and then select the program.

4 Click OK.

Getting Started with ION Mapper

3

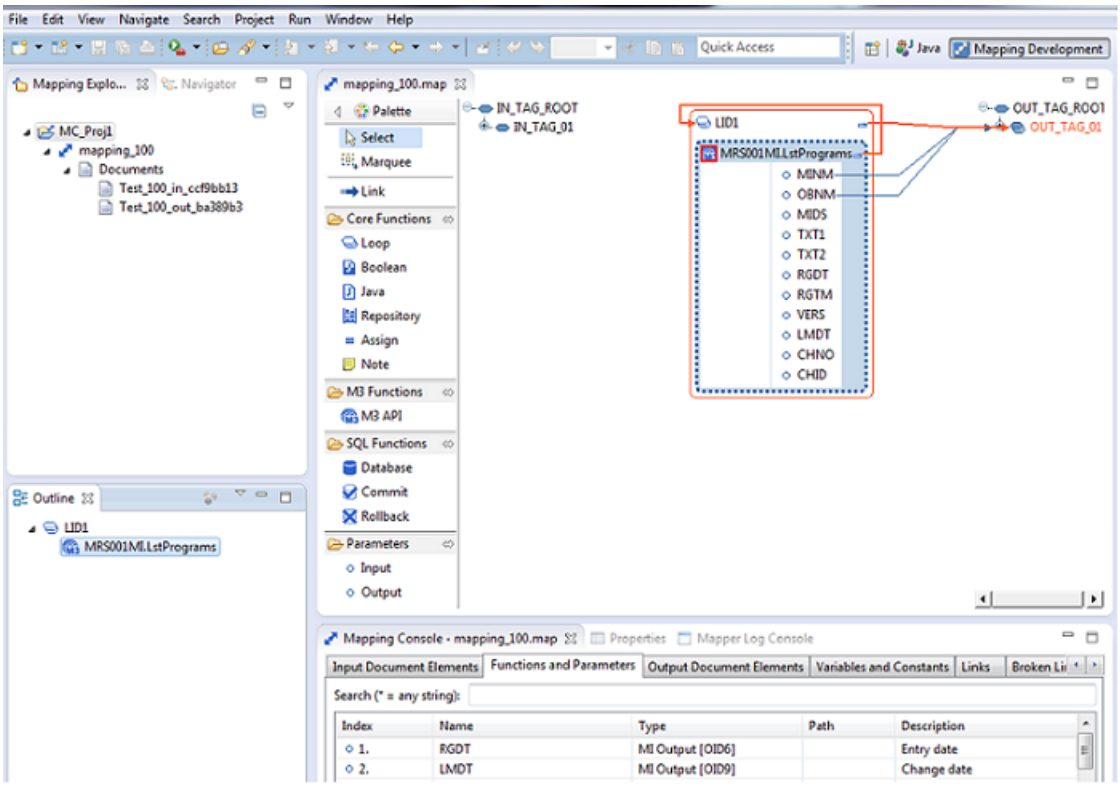
This chapter describes the functions and usage of the ION Mapper Eclipse graphical user-interface.

ION Mapper UI

ION Mapper Perspective

When ION Mapper Eclipse is started it opens to the Mapping Development Perspective with customized views for working with mappings.

The screen shot here shows Mapping Development Perspective with a mapping in full view:



Note: To optimize loading time, the Mapping Editor shows only the required and top-level schema elements connected to mapping elements.

- To see all connected elements, expand the tree view.
- To drill down and see non-connected elements, expand individual schema elements.
- To define a mapping, access the Palette, and use items made available in the drawers of the side pane.

Editor area layout

Allocated view space for the input-, mapping-, and output- editor areas can be changed through the **Mapping.Layout** property. Changing this view is useful especially for documents with long element names.

Access the Properties tab and double-click on **`Mapping.Layout`** to open the field for property. Modify the three sets of numeric values. These numbers represent the editor area split view in percentage for <INPUT_AREA_SIZE MAPPING_AREA_SIZE OUTPUT_AREA_SIZE>.

For example, type the values "20 60 20" to display: 20% for input, 60% for mapping, and 20% for output. The total equals 100% mapping editor view.

Note: You can also change the <INPUT_AREA_SIZE> and <OUTPUT_AREA_SIZE> by using the action **Set Width** in the Input and Output Document context menus, see Mapping Editor tasks.

Mapping Editor tasks

The following views are accessed through the Mapping Editor context menu, right-click on an area in the Mapping Editor and select the appropriate view to display.

Note: The following views can also be displayed through the Eclipse menu, click Window > Show View > Mapping Console and select the appropriate view to display.

Task	Steps
To display the **Mapping Console**	From the Mapping Editor context menu, select **Mapping Console...** **Note:** In Variables and Constants tab, use **Search** to filter the list. A variable and constant may be located in the mapping by double-clicking on the corresponding list entry, and if used multiple times select the desired occurrence in the presented alternatives.
To display the **Properties**	From the Mapping Editor context menu, select **Properties...**
To hide/show unconnected elements	From the input or output document context menu, select the appropriate action: **Hide Unconnected Elements** – unconnected elements and attributes for the document are not shown. The root element icon is overlaid with a funnel to indicate that all elements and attributes for the document may not be shown. **Show Unconnected Elements** - unconnected elements and attributes for the document are shown. **Note:** Only one of these actions is shown at a time
To display complete input/ output tree	From the input or output document context menu, select: **Expand Document** - to display all connected schema elements in the tree plus all schema elements that you have expanded while editing the mapping. **Collapse Document** - to display only top-level elements of the tree. The top two levels are shown. **Note:** Only one of these actions is shown at a time

Task	Steps
To set the width of the document areas	From the input or output document context menu, select the action **Set Width**. You can select Default and from 15% to 50% of the total width of the mapping editor view. When selecting the Default option the with of both the input document area and the output document are set to 20%, leaving 60% for the mapping area. This corresponds to the default value "20 60 20" for the mapping property **Mapping.Layout.** These are the rules for the width options: • Minimum document width is 15%. Less than that is not usable. • Maximum document width is 50%. More than that will make the two other areas too narrow. • Minimum width for the mapping area is 20%. Less than that is not usable. • Total width, input document area + mapping area + output document area, must always be 100%. Because of these rules all widths are not always available, depending of the with of the other document area. **Note:** These settings are stored with the mapping, using the mapping property **Mapping.Layout.**
To display mapping structure	• On Eclipse menu, click Window > Show View > **Outline**. Outline tab is displayed with the hierarchical tree view of the mapping.
To hide/reveal variables and constants	From the Mapping Editor context menu, select the appropriate action: **Hide Variables and Constants** **Show Variables and Constants**
To hide/reveal unconnected parameters	From the Mapping Editor context menu, select the appropriate action: **Hide Unconnected Parameters** **Show Unconnected Parameters**
To edit Java code	From Java context menu, select: **Edit Java Code...** A new tab for selected Java code opens.

Mapping Explorer View

In the left pane, ION Mapping Explorer tab displays an artifacts oriented view of mappings. It shows mapping artifacts together with associated context menus. The artifacts shown are mapping projects with mappings where each mapping has a "Documents" node listing the input/output schema documents. Each artifact, or node, has a context menu.

Select **Refresh** from the project context menu to refresh the mapping project.

Select **Refresh All** to refresh all mapping projects in the workspace.

ION Mapper Views

Use these procedures to display views to help you monitor or find your way around your mappings better. The tabs in each view contain features to help you search for elements in a mapping, or filter log expressions.

- "Displaying the Mapping Console View" on page 27
- "Searching in a Mapping" on page 28
- "Mapping Console Commands" on page 30
- "Displaying the Mapper Log View" on page 31

Displaying the Mapping Console View

Access this view to navigate a mapping or display the progress of mapping operations. This view opens automatically when you invoke a mapping feature with progress feedback, for example, when importing mappings from database.

- Right-click on an area in the Mapping Editor, select **Mapping Console...**

  The Mapping Console view is displayed as a new tab with tabs within. Each sub-tab has a search feature.

  Following is a list of available tabs and description:

Tab name	Description
Variables and Constants	This is the default opening tab for this view. Access this tab to search for variables and constants in a mapping. The Search feature in this tab allows you to filter the variables and constants in the mapping. Double-click on a row to reveal the location of a variable or constant.

Tab name	Description
Links	This tab contains all the links in the mapping. Double-click on a link to select it in the Mapping Editor. You can also delete a link using the context menu. This tab is intended to help you select a link in the editor when working with huge and complex mappings.
Moogle	This tab contains search functionality for the mapping. This tab replaces the tabs **Input Document Elements, Functions and Parameters**, and **Output Document Elements** in previous versions of ION Mapper. For more information, see "Searching in a Mapping" on page 28.
Broken Links	**Important:** Saving a mapping to database may fail if there are broken links. Access this tab to shows the broken links in a mapping. If a link cannot be made, due to a missing source or target element, it will be listed here. The missing party is indicated with a question mark (?), and detailed information is shown below the list. There you can find information about the internal ID's of the missing elements and a reason. If you are a very advanced user you might use the internal ID information, together with a previous version of the mapping file, to deduce why and what is missing. Double-click on a link to select the existing source or target in the Mapping Editor. The broken links will remain in the mapping until you discard them with the "Discard" option in the context menu.
Validation	Shows the result of document or mapping validations. If issues are detected, a list with detailed descriptions are displayed. Select an issue from the list to display more specific details about that issue.
Console Commands and Output	Access this tab to view mapping operation progress and to access additional commands, if needed.

Searching in a Mapping

Moogle tab contains search functionality for the mapping. Use this procedure to search in a mapping.

1 Select the Moogle tab in the Mapping Console view.

2 Select the category you want to search in by using the radio button Search in:

- Select Input Document Elements to search in elements and attributes for the input document.
- Select Mapping Elements to search in mapping elements, for example loop, function and parameter names.
- Select Java Code to search in the Java code for all UVJ and UVB functions.
- Select Output Document Elements to search in elements and attributes for the output document

Note: When selecting one of the categories a new search is done. It is possible to search for the same string in different categories by selecting a new category.

3 Type a string to search for in the field Search for.

This can be, for example, a document element name or a parameter name. The search string is case insensitive and will match part of element names. For example, if you search for "bc" there will be a hit on "ABCD".

You can use the wildcard character * (asterisk) in place of an unknown string. For example, if you search for "b*d" there will be hits for *"ABCD", "bed"* and *"abd-001".*

For advanced users, you can use a regular expression (regex) as search string. Start the string with the character # (hash mark) followed by a regex. Regex are case sensitive. For example, to search for all elements with the name "ID", type the search string "**#.*/ID**"

where:

- **"#"** is the regular expression indicator,
- **".*"** means any character zero or more times,
- **"/ID"** is the end of the XPath.

For more information on regular expressions, see **http://docs.oracle.com/javase/7/docs/api/java/util/regex/Pattern.html#sum, Wikipedia**, or other resources on the internet.

It is the paths for the elements that are used for searching, not the element names. If, for example, the name of a loop matches the search string, all functions and their parameters and also inner loops within the loop will be returned as result because the name of the loop is part of their paths.

To perform the search press Enter, click on the search button, or select a category.

The search result is displayed in a table list. The number of hits is shown in the status bar at the bottom of Eclipse.

4 Optional: To refine the search result, type a string in the field **Find in result**. This filtering is currently applied to all columns. The string is case sensitive and will match part of column values. You can use the wildcard character (*) in place of an unknown string.

Search result

The result table contains the following columns:

- **Index** - The icon of the element. For example, a repeating element, a UVJ function or a parameter, plus a sequence number.
- **Name** - The name of the element. This name does not need to match the search string. It can be the name of a parent element in the element's path that matches the search string.
- **Type** - The type of the element.
- **Path** - The path of the element. An element in this path matches the search string.
- **Description** - The description of the element. Multi-line descriptions are shown in tooltips.

Double-click on an item in the table list to find the element in the mapping editor. The mapping editor will automatically expand parent nodes, scroll to the element, and select (highlight) it.

Data for input and output document elements and attributes are read into memory only when they are "used" which means they are connected and/or expanded in the current editing session. This "lazy loading" is done because it would consume too much memory to, for example, store data for a complete BOD schema in memory.

When searching for input, or output document elements, the search is done both in memory and in the schema including all unexpanded elements.

- Hits on elements/attributes in memory are indicated by their icons as used in the document tree.
- Hits on elements/attributes in the schema are indicated by a document icon, the same icon as used for schemas in the Mapping Explorer view.
- Hits on elements/attributes in the schema are not shown if they are in use.

When resting on the field headers, tooltips are displayed with guidance on how to search.

When searching schemas, all schemas known for a mapping will be searched and this includes custom schemas, if any. If hits occurs in such schemas the returned paths will be relative to the root element(s) of that schema and not relative to a root element of the input- or output schema. In this case, the path shall be regarded only as information on which schema the element is defined. If you double-click on the result row, it will not locate the element in the mapping editor.

Mapping Console Commands

Important: These commands are intended for very advanced users and the tool developers as an aid in error analysis. This is not intended for regular end-users.

For command details, write <*cmd*> **`help`**

Command	Details
`clear`	Clears the console.
`genadm`	Lists stats for or purges generator connections.

Command	Details
gen	Saves the current mapping to file and database and then generates or publishes it. • Use **gen pub** to publish the mapping • Use **gen** to generate the mapping • Use **dbloc** and **genloc** to set the database and generator locations.
genloc	Lists and selects the known mapping generator location(s).
save	Saves a mapping to file using normal or classic format.
dbdel	Removes a mapping or a function from a Mapper database/repository.
dbls	Lists the mappings or functions in a mapping database/repository.
dbloc	Lists and selects the known Mapper database location(s).
dbinf	Lists info about a mapping or a schema in a Mapper database.
dbadm	Lists stats for or purges db connections, result sets, and statements. Shows last SQL error, if any.
version	Prints the version of the ION Mapper Editor Feature.
xdstat	Analyzes and prints stats about an xml schema (tool developer aid).
linkstat	Prints supplementary information about broken links.
history	Prints the history of the ION Mapper Editor Feature.
editor	Enables or disables visual guides. **border_on** displays a border around the sequence.
dump	Dumps the Epoxy model to file (tool developer aid).

Displaying the Mapper Log View

Important: These commands are intended for very advanced users and the tool developers for tracing and debugging purposes. This is not intended for regular end-users.

1. On Eclipse menu, select Windows > Show View > Other...

   The Show View window is displayed.

2. Select to expand the view of ION Mapper > Mapper Log Console.

3 Click OK.

The Mapper Log Console view is displayed as a new tab. Specific tasks are available on each tab within.

Tab name	Description
Log	In this tab you can do the following: • View the path to the log file, this appears as editable but do not change the default value. Click to load and view the current log. • Use **Search** to filter your logs. • Insert mark - Use this option to put in a marker in the log file before invoking actions that generates large amounts of log traces. It helps you to trace exactly what actions are causing an issue. • Clear log • Copy to clipboard • Save current filter as default
Loggers and Configuration	In this tab, here are the tasks that you can do: • View the path to the Configuration file. • View the path to the default log level file. • Use the search filter to short list your logs. • Set the logger level - FATAL, ERROR, WARN, INFO, DEBUG Then click **Apply** or **Apply to all** to execute your selected logger level. • Save the current levels as default.

Mapping Projects

Use the procedures in this section to start an ION Mapper project.

Before you start Ensure that you have configured the ION Mapper.

• "Creating a Mapping Project" on page 33

Creating a Mapping Project

Use this procedure to create a mapping project to contain the mappings that you will create later.

1 In the left pane, right-click on an area in **Mapping Explorer** tab. Select **New Project...**

2 On the New Mapping Project window, type a name for this new project.

3 Click Finish.

In the Mapping Console tab, the status of the new project is displayed. On this tab error messages are also displayed, if any.

In the Mapping Explorer tab, the new project is listed.

To view the default contents of your new project, click on Navigator tab and select to expand the view of your project.

Note: If you are creating the first mapping project, an additional project called **`MappingUtils`** will be automatically created.

This is a resource project that is referenced from your mapping projects. You can safely ignore this project. However, if you happen to delete **`MappingUtils`**, it will be restored the next time you create a new mapping project. This project is not shown in the Mapping Explorer view.

You can create several mapping projects and later select the project where to add mappings.

For more information, see "Creating a Mapping" on page 34.

You can also import existing mappings into your project.

For more information, see "Importing Mappings" on page 45.

Mappings

Use the procedures in this section to start mappings.

Before you start Ensure that you have configured the ION Mapper.

Opening Mappings

Use the instructions here to open a mapping in Mapper Editor.

❑ Open a mapping in Mapper Editor

___1 In the Mapping Explorer tab, right-click on a mapper and select **Open Mapping...**

A new tab for this mapping opens in the Mapper Editor area.

___2 Perform mapping edits and then save the mapping.

❑ Open a mapping as XML document

Important: If you have defined an external XML editor in the Window > Preference menu option, you can also open the mapping in the external XML editor. However this option is only for advanced users. You can easily destroy the mapping by directly editing the map file.

Creating a Mapping

Before you start

- You must have a project to contain your mappings.
- Read and understand the requirements for using complex schemas in a mapping.

Schemas can be very complex and may import or include other schemas. If you select to use complex schemas in your mapping, they must meet two requirements for the Mapper to be able to correctly import all the relevant schemas into the Mapping project.

- All referenced schemas must be present on disk. References to non-local schemas, for example **http://some/url/somewhere/aschema.xsd**, are not valid.
- All references must be relative to the folder with the root schema.

  For example, if root schema **`A.xsd`** import schema **`B.xsd`** and **`C.xsd`**, located in sub-folders **`FB`** and **`FC`** to the folder with **`A`**, the reference paths must be **`/FB/B.xsd`** and **`/FC/C.xsd`** respectively.

Relative paths against the root schema location are also valid, for example if schema A, B, and C are located in the following structure:

MY_FOLDER/A.xsd

MY_FOLDER/SUB_FOLDER/B.xsd

MY_FOLDER/SUB_FOLDER/C.xsd,

then, B may import C as "**../C**"

Important:

- If the schemas do not meet these criteria they must be edited before they can be used.
- The mapping name must be a valid Java class name.

❑ Create a mapping

___1 In the left pane, right-click on an area in **Mapping Explorer** view. Select **New Mapping...**

___2 On the New Mapping window, consider the following fields:

Project	Type the project name where to add this new mapping. Or, browse to the folder selection of projects and click OK. **Tip:** You can initially right-click on a project where to add this new mapping to auto populate this field.
Mapping name	Type a descriptive name for this mapping. Default name: **MyMapping**
Input Schema	This is a mandatory field with a default schema name: **DefaultInput** To change the default, click Browse to navigate to the folder location of another input schema to use and click Open.
Output Schema	This is an optional field. Click Browse to navigate to the folder location of your output schema and click Open.

___3 Click Next.

The New Mapping Root Elements window is displayed.

___4 Select a root element for each of the input and output schema.

The input and output schemas may contain multiple element definitions. Select a root element for the input, and a root element for the output data structure.

Default Element : **IN_TAG_ROOT**

Default Namespace: **`http://www.intentia.com/MBM`**

Note: To accept the default input and output schemas, click Finish without changing the field values.

___**5** Click Finish.

In the **Mapping Console** view, the status of the new mapper and error messages, if any, are displayed.

In the **Mapping Explorer** view, the new mapping is listed under the project where it belongs. To view the contents of your new mapping, expand the nodes under the project folder.

When completed successfully, the new mapping is added to your selected project. Mapping Editor area displays the input schema in the left pane and the output schema in the right pane. The space in the middle will contain a sequence of mapping functions, to be added later on.

❑ Set the mapping schema categories

When a mapping is created, its default **`SchemaIn.Category`** and **`SchemaOut.Category`** are set to **Other**. Use this procedure to set and change the default schema.

___**1** Double-click on a mapping to open it in the Mapper Editor.

___**2** Right-click on an area in the Mapper editor and select Properties.

___**3** On the Properties tab, do the following:

a Find the **`SchemaIn.Category`** property.

b Select a value type for this property.

c Repeat steps a and b for **`SchemaOut.Category`**.

Note: Schema categories are used in ION but not in MEC.

For more information, see "Schema Requirements" on page 51.

Copying Mappings

Use this procedure to copy a mapping under another, or the same, mapping project. The progress and result of this action is logged in the Mapping Console tab.

Important: The new mapping name must be a valid Java class name.

1 On the Mapping Explorer tab, right-click on a mapping to copy. Select **Copy**.

The Copy Mapping window is displayed.

2 Click Browse to select a Mapping Project for this copy.

3 Type a new name for this copy or accept the default name prefixed with "**CopyOf**<*name*>".

4 Click Finish.

Saving a Mapping

Created mappings, even when not yet completed, must be saved on your workspace. In the Mapping Editor view, or in the Java editor view, an asterisk after the mapping name, or function, indicates that the editor content is not yet saved to disk. To save your created mappings, click CTRL-S, or on Eclipse menu, select File > Save.

Before a mapping can be generated or published, it must be saved to the Mapper Database. If the current mapping does not exist in the Mapper Database, you cannot generate or publish it since the Mapper Server reads mapping metadata from the Mapper Database. If you have updated the mapping and generate or publish the mapping without first saving the mapping to the Mapper Database, you will generate or publish an old version of the mapping.

The progress and result of save action is logged in the Mapping Console view.

Note: If you replace a mapping that already exists in the Mapper Database, only the mapping is saved and **not** the schema files. If you need to change and save a schema file, refer to Schema section in this guide.

This behavior is not encouraged. If you change the XML schemas for a mapping you should give the mapping a new version number, or rename the mapping. Then, a new mapping will be created in the Mapper Database.

For more information, see "Editing Schemas" on page 51.

❑ Save to workspace

Use this procedure to save the changes made to your mapping in its current name and workspace location.

___**1** Open the mapping on Mapper Editor.

___**2** On Eclipse menu, select File > Save.

___**3** Click OK.

The mapping and all updates you have made on it is now saved to workspace.

❑ Save to database

Use this procedure to save a mapping to database. The save activity and information is logged on the Mapping Console view.

___**1** Open the mapping on Mapper Editor.

___**2** Right-click on a blank space in Mapper Editor view. Select **Save to Database...**.

The Save Mapping in database window is displayed.

___**3** Select a mapping database location where to save this mapping.

___**4** Optional:

Click **Test Connection...** to check the connectivity before you proceed.

___**5** Click Finish.

Note: If save to database action fails, the database will be restored and the mapping is not saved. Take note of the error and review your mapping or save process.

___**6** Depending on whether or not the mapping to save already exist in the database,

- If the mapping, with the same name and version, does not exist in the database:

  Click OK at the prompts to confirm to add the mapping to the database.

- If the mapping already exist in the database:

  At the confirmation prompts, select to overwrite the existing mapping in the database, or to update the version of the mapping and create a new version of the mapping in the database.

  a Click **Overwrite** to overwrite the mapping in the database.

  b Click **Update Version...** to update the version of the mapping before saving it to the database. A new window is opened where you can type a new version for the mapping. By default, 0.1 is added to the current version. For example, if the current version is 1.1, the proposed new version is 1.2. Change the new version, if necessary, and click OK to create a new version of the mapping in the database.

  The version of the mapping will be updated before it is saved to the database.

  Note: Use **Update Version...** only when you want to create a new version of the mapping.

  While developing a mapping you should always overwrite the existing mapping in the database because there is no need to keep intermediate versions of the mapping. The recommended procedure when creating a new version of a mapping is to first update the version of the mapping, and then overwrite the new version of the mapping, until the development is complete.

Deleting Mappings

Use the instructions here to remove a mapping from your workspace or a database.

Important:

- **Delete from workspace** removes the complete mapping folder and not only the mapping file.

  This action can be undone using the Eclipse standards feature "Restore from Local History" in the Navigator tab.

- **Delete from database** has no undo function. Verify the version and name of a mapping before you proceed.

Note: Input and output XML schemas are also saved in the database. If two or more mappings use the same XML schema the XML schema is only stored once in the database. To remove an XML schema from the database all mappings using the XML schema must be deleted. When the last mapping using an XML schema is deleted from the database the XML schema is also deleted from the database.

❑ Delete from database

Before you start Ensure that your selected mapping is saved on your selected database. The mapping, with the correct mapping version, must also exist in you workspace to be able to remove it from the database.

___**1** Open the mapping on Mapper Editor.

___**2** Right-click on a blank space in Mapper Editor. Select **Delete from Database...**.

The Delete Mapping from database window is displayed.

___**3** Select a mapping database location where this mapping is stored.

___**4** Optional: Click **Test Connection...** to check the connectivity before you proceed.

___**5** At the delete confirmation prompt, click OK.

The selected mapping is now removed from the database.

❑ Delete from workspace

___**1** On the Mapping Explorer tab, right-click on a mapping to delete. Select **Delete**.

___**2** At the delete confirmation prompt, click OK.

The selected mapping is now removed from workspace.

Validating Document

1 In the Mapping Explorer view, expand the mapping node if not already expanded.

Right-click on the Documents node and select Validate Documents.

All XML schemas for the mapping will then be validated. You can also right-click on a single Document node to validate only that schema.

In the Mapping Editor, right-click on a document and select **Validate Documents**.

Validation checks the encoding, references, and checksums on a document.

2 On the Mapping Console tab, a detailed report is logged.

If an error occurs, click on the details link. Take note of the error to correct.

The validation process will check the consistency of the mapping details:

Updating Document Checksums

A document may contain a checksum value as part of the document name. Checksums must be updated when changes are made to document contents.

1 In the Mapping Editor, right-click on a document and select **Update Document Checksums**.

2 At the Confirm Checksum Update prompt, click OK.

The document checksums are now updated.

Validating Mapping

Completed mappings may be validated to avoid compilation erorr when you run publish or generate.

1 In the Mapping Editor, right-click and select Validate.

2 In the Mapping Console tab a detailed report is logged.

If an error occurs, click on the details link. Take note of the error to correct.

The validation process will check the consistency of the mapping details.

Checks performed:	Check details:
loops	that loops are controlled by a function element. the controlling function element, if it is a UBJ function then it must be the first child of the loop.
data links	that the data links are made between source and target elements using the same or compatible data type.
input/output root elements	that the input/output tree root element is defined in the input/output schema.

Checks performed:	Check details:
variables and constants	for unused (unconnected) variables and constants.

Renaming Mappings

Important:

- The "Refactor > Rename" action from Eclipse is **not** applicable to mappings. Invoke **Rename** only through the ION Mapper context menu in the Mapping Explorer view.
- The mapping name must be a valid Java class name.

1 On the Mapping Explorer tab, right-click on a mapping. Select **Rename**.

Note: Your selected mapping must not be open in the Mapper Editor. Otherwise, click OK when prompted to close the Mapping Editor.

2 Type a new name for this mapping.

3 Click OK.

On the Mapping Explorer view, a new name is assigned to your selected mapping.

Changing Documents and Root Element

Use this procedure to update or change the input/output schema or document root elements of a mapping.

The progress and result of this action is logged in the Mapping Console view.

1 Select a mapping to use.

2 In the Mapping Explorer view, right-click on the mapping, and select **Change Documents...**

The Change Documents window is displayed.

Note: The fields will be automatically populated with default values.

3 Consider the following fields:

Project Defaults to the current project name. This field is not editable.

Mapping Name	Type a new name for the mapping. **Important:** • The mapping name must be a valid Java class name. • Because this is a potentially dangerous operation your current mapping will be preserved. This means that a new mapping folder will be created by copying your current mapping and applying the changes.
Input schema	Type or Browse to select a new input schema, or accept the default value.
Output schema	Type or Browse to select a new output schema, or accept the default value.

4 Click Next.

The Root Elements window is displayed.

5 Select root elements for the input or output tree.

6 Click Finish.

The new mapping opens on a new tab in the Mapping Editor.

Important: Document elements linked to mapping elements may have disappeared if the new schemas have different content compared to the old ones. This will result to an inconsistent mapping with broken links.

Navigating Around Mappings

You can display all your mappings and navigate around the ones in the Mapping Explorer view.

1 In Eclipse, go to Mapping Explorer view.

2 Select a project and expand it.

Each mapping icon indicates a mapping.

3 Select a mapping.

The Mapping Editor displays the mapping in a compressed state. Schemas are shown as hierarchical tree structure.

Solid blue bullets indicate required schema elements, hollow blue bullets indicate optional elements.

Layered bullets indicates repeating elements. Smaller round bullets indicates element attributes.

Note: To optimize loading time, the Mapping Editor shows only the required schema elements and top-level schema elements connected to mapping elements.

- To see all connected elements, you can expand the tree.
- To drill down and see non-connected elements, you can expand individual schema elements.

4 Perform any of the following tasks:

Note: In Variables and Constants tab in Mapping Console View, use **Search** to filter the list.

A variable and constant may be located in the mapping by double-clicking on the corresponding list entry, and if used multiple times select the desired occurrence in the presented alternatives.

Task	Steps
To display the complete input/ output tree	From the Mapping Editor context menu, select: **Expand Document** - to display all the connected schema elements in the tree. **Collapse Document** - to display only top-level elements of the tree.
To locate mapping elements	Perform any of the following: • Access the **Mapping Console View**. All mapping elements are listed in the various tabs of Mapping Console View. • Or, from the Mapping Editor context menu, select **Open Console**. Double-click a row in a list to reveal the corresponding element in the Mapping Editor.
To display the structure of a mapping	• On Eclipse menu, click Window > Show View > **Outline**. Outline tab is displayed showing a hierarchical tree view of the mapping.
To hide/reveal variables and constants	From the Mapping Editor context menu, select: **Hide Variables and Constants** - to hide connected variables and constants. **Show Variables and Constants** - to display connected variables and constants.

Mappings and Mapping Projects

This section begins with creating projects and mappings, setting the connectivity, and continues with navigating a mapping.

Import and Export Mappings

Exporting Mappings

Use the instructions here to add new or export existing mappings to your mapping project.

1. On the Mappings Explorer tab, right-click on a mapping. Select **Export...**.

   The Export Mapping window is displayed.
2. Select the mapping format to use.
   - Normal - exports a mapping using the .zap format.

     This is the preferred option.
   - Classic - exports a mapping using the .zap format but uses an older, classic, internal format.

     Use this option to import mappings in Mapper versions earlier than 9.2.0.
3. Click Next.

   The Folder and File window is displayed.
4. Type the path or click Browse to navigate to the folder location for the mappings.
5. Type a new name for this mapping to export or accept the default name prefixed with <*mapping_name*>.zap.

   Important: Do **NOT** rename a zap file after export. It is named as the mapping and must remain so in order to be importable in another ION Mapper.
6. Click Finish.

Importing Mappings

Use the instructions here to add new or import existing mappings to your mapping project.

Before you start Ensure that you have Mapper database connectivity.

❑ Import from database

___1 On the Mapping Explorer tab, select a project to contain your imports.

___2 Right-click and select **Import > From Database...**.

The Import Mapping window is displayed.

___3 Select a database location to use.

___4 Optional: Click **Test Connection...** to verify the machine connectivity.

___5 Click Next.

___6 Select the mappings to import, use either, CTRL-click or SHIFT-click.

___7 Click Finish.

Note: If a mapping with the same name and version already exists in the project you will be prompted to overwrite the existing mapping or not.

❑ Import from classic file

Use this procedure to import classic xml files located in a folder with the associated input/output schemas that is exported by the classic Mapping Manager.

Important: You must select a classic mapping to import and all referenced schemas must be present in the mapping folder and use valid relative paths, if any. For details refer back to the discussion about schemas in Schemas section in this document.

___1 On the Mapping Explorer tab, select a project to contain your imports.

___2 Right-click and select **Import > From Classic (.xml) File...**.

The Import Mapping window is displayed.

___3 Click Browse and select a classic .xml mapping file format to import.

___4 Click Finish.

Successfully imported mapping is shown as a new mapping folder in the Mapper Explorer tab with the input and output schema.

For more information, see "Schema Requirements" on page 51.

For more information, see "Editing Schemas" on page 51.

❑ Import from archive file

___1 On the Mapping Explorer tab, select a project to contain your imports.

___2 Right-click and select **Import > From Archive (.zap) File...**.

The Import Mapping window is displayed.

___3 Click Browse and select an archive .zap mapping file format to import.

___4 Click Finish.

Successfully imported mapping is shown as a new mapping in the Mapper Explorer tab.

Transferring Mappings

Mappings can be transferred between databases without having to import or export every mapping to Mapping projects first. This type of transfer is done through the import or export option from the Database connectivity wizard.

Transferred mappings, to or from database, will manifest as a "Mapping Unit" file with the "mpu" extension. An mpu unit is a self-contained mapping complete with schemas and stored as one xml file.

❑ Import to database

Use this procedure to send a batch of mappings and all its schemas into your selected database

___1 Access the Database Connectivity Preference window.

___2 Select a database connection to use. Click **Import.**

___3 Click Browse to navigate to the location of your source folder.

___4 Click Next.

___5 Select the mappings to import.

Use **Search** to filter your results.

___6 Click Finish.

If there is an error, take note of the error details to correct and click OK.

❑ Export from database

Use this procedure to get a batch of mappings and all its schemas from your selected database.

___1 Access the Database Connectivity Preference window.

___2 Select a database connection to use. Click **Export.**

This opens a wizard that lets you select a destination folder and the mappings to export.

___3 Click Browse to navigate to the location of your destination folder.

___**4** Select the mappings to export.

Use **Search** to filter your results.

___**5** Click Finish.

If there is an error, take note of the error details to correct later. Click OK.

Deploy Mappings

A mapping is saved in a Mapper Database associated with the Mapper Server before it can be used by ION or MEC. Saved mappings are then deployed through the Publish action.

- **Generate a mapping** - the mapping is tested on Java.

  For **ION**, when you generate a mapping it does not replace the current version of the mapping in the Mapper Server. Use Publish to update the sever copy.

  For **MEC**, when you generate a published mapping, it replaces the current version of the mapping in the Mapper Server.

- **Publish a mapping** - the Mapper database is updated to include the mapping.

  Successfully published mappings become available for use in ION and MEC.

Important: Before you deploy mappings:

- Define a connectivity to a Mapper generator that is associated with the Mapper Server.
- It must be saved to the Mapper Database.

If your current mapping does not exist in the Mapper Database you cannot generate or publish it since the Mapper Server reads mapping metadata from the Mapper Database.

If you have updated the mapping, then you generate or publish the mapping without first saving the mapping to the Mapper Database, you will generate or publish an old version of the mapping.

For more information on deploying multiple mappings, see "Deploy Multiple Mappings" on page 49

- "Saving a Mapping to Database" on page 47
- "Generating Mappings" on page 48
- "Publishing Mappings" on page 48

Saving a Mapping to Database

Before a mapping can be generated or published it must be saved to the Mapper Database.

If the current mapping does not exist in the Mapper Database, you cannot generate or publish it since the Mapper Server reads mapping metadata from the Mapper Database.

If you have updated the mapping and generate or publish the mapping without first saving the mapping to the Mapper Database, you will generate or publish an old version of the mapping.
For more information, see "Saving a Mapping" on page 37.

Generating Mappings

Use this procedure to verify that a mapping can be translated into Java code accepted by the **Mapper Server**.

1. Double-click on a mapping to open it in the Mapper Editor view.
2. Right click on the Mapper Editor view, select **Generate...**

   The Generate Mapping window is displayed.
3. Select a server location where to deploy the mappings.

   Optional. Click **Test connection** to check the connectivity to the server machine. Click OK at the connectivity prompts.
4. Click Next.
5. Select a mapping database location and click Finish.

   The generate activity and information is logged on the Mapping Console view.

Publishing Mappings

Saved mappings are published using a Mapper Generator that is associated with the **Mapper Server**.

Publish action updates the Mapper Database to make the mapping available for the Mapper Server. The mappings are loaded into the Mapper Sever and ready to run.

For ION, published mappings can be used through ION Desk to configure active document flows and/or business connection points based on the input/output schema types that are specified in the mapping.

You need to reactivate the mapping to use the republished mapping.

For MEC, published mappings can be used in Partner Agreements using the Partner Agreement Tool.

You need to reload the mapping to use the republished mapping.

1. Double-click on a mapping to open it in the Mapper Editor view.
2. Right click on the Mapper Editor view, select **Publish...**

The Publish Mapping window is displayed.

3 Select a server location where to publish the mappings.

Optional. Click **Test connection** to check the connectivity to the server machine. Click OK at the connectivity prompts.

4 Click Next.

5 Select a mapping database location and click Finish.

The publish activity and information is logged on the Mapping Console view.

Deploy Multiple Mappings

A mapping is saved in a Mapper Database associated with the Mapper Server before it can be used by ION or MEC. Saved mappings are then deployed through the Publish action. You can simultaneously publish multiple mappings or simultaneously generate multiple mappings.

Saving a mapping to database is part of the deployment process. For more information, see "Saving a Mapping to Database" on page 47.

For more information on deployment, see "Deploy Mappings" on page 47

- "Generating Multiple Mappings" on page 49
- "Publishing Multiple Mappings" on page 50

Generating Multiple Mappings

You can simultaneously generate multiple mappings from the Mapping Explorer view. The generate activity and information is logged on the Mapping Console view.

1 Select one, or several mapping projects in the Mapping Explorer view.

2 Right-click on one of the selected mapping projects, select **Generate...**

The Generate Mapping window is displayed.

3 Select a server location.

Optional. Click **Test Connection** to check the connectivity to the server machine. Click OK at the connectivity prompts.

4 Click Next.

5 Select a Mapping Database location and click Next.

All mappings in the selected projects are now displayed.

Note: A mapping must be saved to the selected Mapping Database in order to generate it. The column **Is in Database** shows whether the mappings are saved to the selected database or not.

It is not guaranteed that a mapping in the database is exactly the same mapping as the mapping in the mapping project, only that a mapping with the same name and version exists in the database.

6 Select one or several mappings to generate and click Finish.

Publishing Multiple Mappings

You can simultaneously publish multiple mappings from the Mapping Explorer view. The publish activity and information is logged on the Mapping Console view.

1 Select one, or several mapping projects in the Mapping Explorer view.

2 Right-click on the Mapper Editor view, select **Publish...**

The Publish Mapping window is displayed.

3 Select a server location where to publish the mappings.

Optional. Click **Test connection** to check the connectivity to the server machine. Click OK at the connectivity prompts.

4 Click Next.

5 Select a mapping database location and click Next.

All mappings in the selected projects are displayed.

Note: A mapping must be saved to the selected Mapping Database in order to publish it. The column **Is in Database**, shows whether the mappings are saved to the selected database or not.

It is not guaranteed that a mapping in the database is exactly the same as the mapping in the mapping project, only that a mapping with the same name and version exists in the database.

6 Select one or several mappings to publish and click Finish.

The publish activity and information is logged on the Mapping Console view.

Schemas

Schema Requirements

Schemas can be very complex and may import or include other schemas. If you select to use complex schemas in your mapping, they must meet two requirements for the Mapper to be able to correctly import all the relevant schemas into the Mapping project.

Important: If the schemas do not meet these criteria they must be edited before they can be used.

Use the following criteria when selecting to use complex schemas in your mapping.

- **All referenced schemas must be present on disk**.

  References to non-local schemas are not valid, for example, **`http://som/url/somewhere/aschema.xsd`**.

- **All references must be relative to the folder with the root schema.**

  For example:

  - Root schema **`A.xsd`**, Import schema **`B.xsd`** and **`C.xsd`**
  - Location: sub-folders **`FB`** and **`FC`** to the folder with A.

  The reference paths must be **`/FB/B.xsd`** and **`/FC/C.xsd`** respectively.

  Relative paths against the root schema location are also valid, for example if schema A, B and C are located in the following structure:

  `MY_FOLDER/A.xsd`

  `MY_FOLDER/SUB_FOLDER_B/B.xsd`

  `MY_FOLDER/SUB_FOLDER_C/C.xsd,`

  then B may import C as "**`../SUBFOLDER_C/C.xsd`**"

Editing Schemas

ION Mapper does not support direct editing of input or output document schemas. Additionally, saving a changed schema to a mapping database can contain complicated procedure.

Use the following guidelines in editing schemas.

1 Edit the schemas.

Important: If you have defined an external XML editor in the Window > Preference menu option you can also open the mapping in the external XML editor. However this option is only for advanced users. You can easily destroy the mapping by editing the map file directly.

2 Update the schema checksums.

ION Mapper keeps track of schema changes by appending a checksum, based on the content, to the end of the file name of all schemas in a mapping project. This means, that if you edit a schema, the checksum has to be updated and because other schemas might reference your changed schema, these references must be updated as well.

a On Mapping Explorer view, right-click on a Document node and select **Update Document Checksums...**

b At the confirmation prompt, click OK to continue and update document checksums.

3 Save the changed schemas.

Saving in a mapping database can be done in two ways:

- **Method 1 - recommended method**

  Remove the mapping from the database and then add it back again.

  Use this method specifically when you add mapping to a database for the first, and the existing mapping is saved to a database but the documents are not updated.

  If you want to keep your existing mapping, increase the version number for the mapping with the changed schema. Or, rename the mapping, and then save it to the database. The database will then contain both the old and the changed schemas.

- **Method 2 - optional method**

  Lets you update a specific schema in the database, provided that you know its database ID. Use this method only for emergencies.

  Important: In this method, many mappings may share the same schema in the database and updating in this way may corrupt other mappings that are unaware of the changes made.

  You must check if a schema is used by multiple mappings. Use the **`dbinf xsd`** *<schema_id>* command.

  a Set the target database location using the **`dbloc`** command.

  b Find the ID of the mapping of which schema(s) you are about to change using the **`dbls map`***<name or *>* command.

  c Find the ID's of the mappings used by this schema using the **`dbinf map`** *<mapping_id>* command.

  d Replace the content of a schema using the **`dbput xsd`** *<schema_id> <path_to_changed_schema>* command.

Wildcard Elements

- "Wildcard Elements Overview" on page 53

Wildcard Elements Overview

This chapter guides you on how to handle the XML schema element **any**. This XML schema construct is used, for example, for Infor BOD UserArea extensions. ION Mapper follows the W3C standards. The specification for ION BOD UserArea extensions is a subset of the W3C standards.

The XML schema element **any** enables the XML document to be extended with elements that are not specified by the schema. So, we do not know the data structure for the **xs:any** element. This is not good for data mapping, since data mapping requires known data structures to map data from and to. The solution is to define the elements that will replace the wildcard construct in instance documents. When these elements are defined for the mapping they are shown in the document trees and can be used just like any other element that is declared in the main schemas.

For terminologies used in wildcard elements, see "Wild Card Elements terminologies" on page 119

Wildcard Elements Limitations

There are two attributes for the **xs:any** element.

Namespace

This attribute specifies the namespaces containing the elements that can be used as replacement elements. ION Mapper assumes that elements from any namespace are allowed. This is the default behavior for this attribute.

processContents

This attribute specifies how the XML processor (receiving the output document) should handle validation against the replacement elements. ION Mapper assumes that the XML processor must obtain schemas for the used namespaces and validate the replacement elements. This is the default behavior for this attribute.

The described **xs:any** element looks like this:

```
<xs:any namespace="##any" processContents="strict"/>
```

The **xs:any** element's **minOccurs** and **maxOccurs** attributes are also reflected when showing wildcard elements in the input and output document trees.

Infor BOD UserArea Extensions

UserArea

UserArea is a construct provided by OAGIS that allows for data not previously defined for a noun to be associated with a noun. This is built with an "**xs:any**" type which says that any defined element can be used inside it. For more information on UserArea elements, see the Infor ION technical documentation.

The UserArea element has a wildcard element as its only child. This is the complex type used by all UserArea elements:

```
<xs:complexType name="UserAreaType" block="restriction">
 <xs:sequence>
  <xs:any namespace="##any" processContents="strict" minOccurs="0" maxOccurs="unbounded"/>
 </xs:sequence>
</xs:complexType>
```

A UserArea element may have one or more replacement elements as child elements. There are two types of replacement elements that can be used in Infor BODs:

User Defined Fields

User Defined Fields are always published using the Property element, which belongs to the default namespace.

BOD instance example:

```
<UserArea>
 <Property>
  <NameValue name="ln.Priority" type="NumericType">999</NameValue>
 </Property>
 <Property>
  <NameValue name="ln.Owner" type="StringType">SFC</NameValue>
```

```
 </Property>
</UserArea>
```

Custom Elements

Custom elements may not be added by standard development teams. These are new elements that are defined in a custom namespace.

BOD instance example:

```
<UserArea>
 <cust:AdditionalInformation>
  <cust:ReceivingWarehouse>WHAMS1</cust:ReceivingWarehouse>
  <cust:InterfaceUpdate>No</cust:InterfaceUpdate>
 </cust:AdditionalInformation>
</UserArea>
```

ION Connect restrictions

Here are some restrictions for ION Connect:

- Only one custom namespace is allowed per customer. Although, several custom schemas can be used for different replacement elements, for different UserArea elements.
- Each custom schema can only have one top-level element declaration, i.e. a replacement element. That element may be a complex element, but there should not be any includes or imports in the custom schema.
- Only one custom schema can be used per UserArea element. However one custom schema can be used by several UserArea elements.

ION Registry metadata provides information about all custom elements per noun for a customer. ION Mapper provides built-in functionality to read this metadata from the ION Registry, or from a copy thereof.

Icons

The element icons for wildcard elements and replacement elements are marked with an asterisk.

A wildcard element does not have a name, only the element icon with an asterisk is shown. Even though a wildcard element is shown as having in its own level in the document tree, this level does not represent a level in the data structure. Instead, it is used as a place holder for the wildcard element's replacement elements. For example, the relative XPath for a Property replacement element is "**`UserArea/Property`**", not "**`UserArea/ /Property`**".

Custom Namespace

A custom namespace is given by a custom schema's target namespace. Several custom schemas using the same target namespace can be used per custom namespace. The custom namespace must not be the same as the main schema's target namespace, any other namespace defined in the main schema, or another custom namespace.

Note: Custom namespaces, custom schemas, and replacement elements are handled separately for the input document and for the output document. Only the custom schema files are shared.

Managing a Custom Namespace

Custom namespaces are global per document, you can use any wildcard element to manage them.

To add a Custom Namespace

1. Right-click on a wildcard element and select Manage Custom Namespaces... in the context menu
2. Click Add to add a custom namespace.
3. Click Add to add a custom schema.
4. Select a custom schema (xsd file) to import.

   The target namespace must not already exist in the main schema or as another custom namespace. The namespace URI is displayed.
5. Type a unique custom prefix and click OK.

   The selected custom schema is added to the mapping.
6. Click OK to close the Manage Custom Namespaces dialog.

   Note: If the mapping is published you must first unpublish the mapping to be able to save the mapping to the database since the schemas are changed.

   The custom schemas are saved to the database, not only the mapping. You can increase the mapping's version number and publish a new version of the mapping instead.

To remove a Custom Namespace

Important: All associated replacement elements for the custom namespace must be removed before you can remove the custom namespace.

1. Right-click on a wildcard element and select **Manage Custom Namespaces...**
2. Select the custom namespace that you want to remove and click **Remove**.
3. At the confirmation prompt, click OK to remove the custom namespace.
4. Click OK to close the Manage Custom Namespaces dialog.

Note: If the mapping is published you must first unpublish the mapping to be able to save the mapping to the database since the schemas are changed.

When the mapping is saved, the custom schemas for the deleted custom namespace are removed from the mapping's file system. Although, a custom schema is not removed if it is also used for the other document (input or output).

Custom Schema

A custom namespace must always have at least one custom schema, since it is the target namespace in the custom schema that defines the custom namespace. Several custom schemas with the same target namespace can be used for a custom namespace. Top level element declarations from all these custom schemas for a custom namespace can then be used as replacement elements.

Custom schemas are handled exactly as the main schemas in the file system. They are copied from their original locations and stored together with the mapping metadata files.

Saving - When you save a mapping with custom schemas to the mapper database, the custom schemas are also saved.

Import and export - When you import and export a mapping the custom schemas are also imported and exported.

So, custom schemas follow the lifecycle of the mapping, just as the main schemas do.

Managing Custom Schema

You can add additional custom schemas to an existing custom namespace. The first custom schema was added when the custom namespace was created. And, you can remove a custom schema for a custom namespace.

To add a custom schema

1. Right-click on a wildcard element and select **Manage Custom Namespaces...**
2. Select the custom namespace that you want to add a new custom schema to and click **Edit**.
3. Click **Add** in the new dialog to add a custom schema.
4. Select a custom schema (xsd file) to import.

Note: The target namespace for the selected custom schema must be the same as the existing custom namespace URI.

The selected custom schema must not contain a top level element that is already declared in another custom schema for the custom namespace.

The added custom schema or schemas are shown. Repeat steps 3 and 4, if necessary.

5 At the prompts, click **OK**.

6 Click **OK** to close the **Manage Custom Namespaces** dialog.

Note: After the custom namespace is created, you cannot change the prefix.

To remove a custom schema

Note: If there is only one custom schema for a custom namespace you cannot remove it since a custom namespace must always have at least one custom schema.

1 Right-click on a wildcard element and select **Manage Custom Namespaces...**

2 Select the custom namespace that you want to remove a custom schema from and click **Edit**.

3 Select a custom schema to remove and click **Remove**.

Repeat this step to remove more custom schemas.

4 Click **OK** to remove the custom schema(s), or

Click **Cancel** to undo this operation.

5 Click **OK** to close the **Manage Custom Namespaces** dialog.

Managing Replacement Element

Replacement elements are handled per wildcard element. You must first find the wildcard element for the replacement element.

To add

1 Right-click on the wildcard element and select **Add Replacement Element...**

A list of all available replacement elements is shown. Namespace URI, prefix and name are shown for each replacement element. Replacement elements that are already added for the current wildcard element are omitted. There is a search field to help you find the correct replacement element. The search string can be part of the desired value and is not case sensitive. You can use an asterisk (*) in the search string as a wildcard character, replacing zero, one or many unknown characters. To

the right of the search field there is a drop-down box where you can select to search in all columns or any of the three columns.

2 Select the desired replacement element and click **OK**.

The replacement element is added as the last entry under the wildcard element.

3 To move the replacement element upwards, right-click on the replacement element and select **Move Up**.

The replacement element's cardinality is automatically set, but it can be updated.

To update

1 Expand the wildcard element to show its replacement elements.

2 To move a replacement:

- Upwards - right-click on the replacement element and select Move Up.

  This option is not available for the first replacement element.

- Downwards - right-click on the replacement element and select Move Down.

  This option is not available for the last replacement element.

3 To change the cardinality of a replacement element:

a Select the replacement element and open the Properties view.

b Click on the values for the properties Mandatory and Repetitive to change the cardinality.

The replacement element's icon will be updated in real-time.

Note: Child elements to a replacement element cannot be moved or updated because they are defined in the schema for the replacement element.

To remove

1 Expand the wildcard element to show all replacement elements.

2 Right-click on the replacement element and select **Remove...**

3 At the confirmation prompt, click **OK** to remove the replacement element.

All links to, or from, the removed replacement element are deleted.

ION Registry

The ION registry contains custom schemas and metadata for custom elements, that is replacement elements that are declared in the customer's custom namespace. ION Mapper can read this metadata and automatically add custom schemas and replacement elements using the ION registry metadata. The ION registry is a folder structure, so it does not matter if the original ION registry folder is used (maybe using a read-only share), or if you use a copy of the ION registry.

Note: ION Mapper only reads data from the ION registry, data is never altered.

Preferences

To connect ION Mapper to an ION registry, you have to enter the ION registry root folder in Window > Preferences > ION Mapper > ION Registry > Root folder. The ION registry root folder must contain the following sub-folders:

- BODMetadata
- BODMetadata\Standard
- BODMetadata\Standard\ BODs
- BODMetadata\Standard\Nouns
- BODMetadata\Standard\Resources
- BODMetadata\Standard\XML
- BODMetadata\Custom
- BODMetadata\Custom\BODs
- BODMetadata\Custom\Resources
- BODMetadata\Custom\XML

Note: From ION version 11.1 only flattened BOD schemas are delivered in the ION registry, no canonical BOD schemas are delivered.

ION Registry Tasks

❑ Setting the schema category

The **New Mapping** and **Change Document** wizards automatically detect if the input schema and/or the output schema is a BOD schema from the ION registry. The schema category, the mapping property **`SchemaIn.Category`** or **`SchemaOut.Category`**, is set to "BOD" if the following conditions are true:

- The ION Registry Root folder preference is not blank.

- The directory path for the selected xsd file is any one of these:

  for flattened BOD schemas - **`[rootFolder]\BODMetadata\Standard\BODs\`**

  for canonical BOD schemas - **`[rootFolder]\BODMetadata\Standard\BODs\Developer\`**

- The selected root element is a valid BOD root element, i.e. the root element name confirms to the pattern:

  `"[Acknowledge|Cancel|Change|Confirm|Get|Load|Notify|Post|Process|Respond|Show|Sync|Update][noun]"`

  Tip: You can also manually set the schema category in the Properties view.

❑ Add a Property Replacement Element

Infor BODs use the replacement element "Property", declared in the default namespace (http://schema.infor.com/InforOAGIS/2) for User Defined Fields. If the document's category, the mapping property **`SchemaIn.Category`** or **`SchemaOut.Category`**, is set to "BOD", the option Insert Property Element is shown in the context menu for all wildcard elements.

___ Right-click on the wildcard element and select **Insert Property Element**.

The Property element, declared in the default namespace, is automatically inserted first under the wildcard element. Its cardinality is set to non-mandatory and repetitive.

Note: This functionality does not require ION Mapper to be connected to an ION registry, because the ION registry does not contain any metadata about Property replacement elements.

❑ Add Custom Elements from ION Registry

If the ION registry root folder preference is not blank and the document's category, the mapping property **`SchemaIn.Category`** or **`SchemaOut.Category`**, is set to "BOD", ION Mapper can read the ION registry metadata and automatically create a custom namespace and replacement elements.

___**1** Right-click on the document's root element and select **Extend UserArea Elements...**.

All replacement elements as defined for the noun in the ION registry are now automatically added to the wildcard elements for the document's UserArea elements.

The custom namespace and its custom schemas are also added.

___**2** ION registry only contains metadata about replacement elements for the custom namespace. So, the Property replacement element used for User Defined Fields must be manually added. A simple instruction in the ending dialog shows to do this.

The custom prefix "cust" will be used for the custom namespace.

Note:

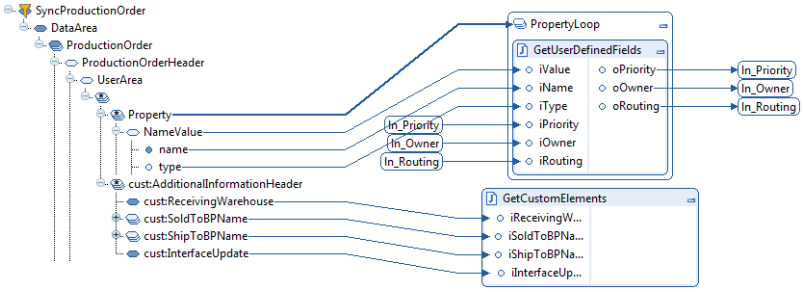
- If no custom elements are added, or if you miss a custom element, check the ION registry metadata.
- If one, or more, new replacement elements are added in the ION registry you can re-run **Extend UserArea Elements...** to add these replacement elements to the document.
- If a replacement element is removed in the ION registry you have to manually remove the corresponding replacement element for the document in ION Mapper.

Mapping

Replacement elements and their child elements and attributes are handled in exactly the same way as any other element and attribute in a mapping, they follow the same mapping rules.

Note: It is not possible to link from or to a wildcard element.

Here is an example of a mapping where data from an Infor BOD UserArea extension is retrieved:



Schema Location

In output instance documents all used namespaces will be declared in the document's root tag. If the output instance document contains replacement elements that are declared in custom namespaces these namespaces will also be declared.

BOD instance example:

```
<SyncProductionOrder xmlns="http://schema.infor.com/InforOAGIS/2"
xmlns:cust="http://schema.infor.com/InforOAGIS/Custom" releaseID="9.2" versionID="2.7.0">
```

A custom schema is referenced from the instance document, not from the schema for the default namespace. For an XML processor to be able to validate the document, schema locations must be given so that all namespaces can be found. For the example above the default namespace is defined in the main schema, that is the BOD schema.

The custom namespace is defined in one or more custom schemas. The locations for these schemas are given in the attribute "**xsi:schemaLocation**," as multiple value pairs, separated by a space.

- The first value is the namespace to use.
- The second value is the location of the XML schema to use for that namespace.

A schema (xsd file) can be located by either a local file path or an internet URL.

Note: Several schemas can be used for a namespace.

In Partner Admin (PA) tool, if you check the "Add Schema Location" checkbox for the XML Transformation process step, but you leave the field "Schema Location" blank, the original xsd file names are used, if possible. All custom schemas for the custom namespaces, if existing, are also given. The XML schema instance namespace is also given because the "schemaLocation" attribute belongs to that namespace.

For example:

```
<SyncProductionOrder xmlns="http://schema.infor.com/InforOAGIS/2"
xmlns:cust="http://schema.infor.com/InforOAGIS/Custom"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://schema.infor.com/InforOAGIS/2 SyncProductionOrder.xsd
http://schema.infor.com/InforOAGIS/Custom CustomHeader.xsd
http://schema.infor.com/InforOAGIS/Custom CustomDetail.xsd" releaseID="9.2" versionID="2.7.0">
```

Note: All custom schemas for a custom namespace are always listed, even if replacement elements from all custom schemas are not mapped. If no replacement element for the custom namespace is mapped, the custom namespace and its custom schemas are not listed.

For the XML processor that receives the XML document to be able to validate the XML document, all xsd files must reside in the XML processor's current directory, otherwise it will not be able to locate the xsd files. If you want to provide an URL and or a local directory path for the xsd files you have to give a specific schema location value. Note that you not only have to enter a location for the main schema (for the default namespace), you also have to enter locations for all custom schemas for the used custom namespaces.

If you enter the schema location in the field "Schema Location" for the XML Transformation process in the Partner Admin Tool, you can use any file paths or internet URLs for the schema files.

Here is an example schema location string and the resulting root tag:

```
http://schema.infor.com/2.7.0/InforOAGIS/BODs/Developer/SyncProductionOrder.xsd
http://schema.infor.com/Custom Custom\UserAreaExtensions\Customer\CustomHeader.xsd
http://schema.infor.com/Custom Custom\UserAreaExtensions\Customer\CustomDetail.xsd

<SyncProductionOrder xmlns="http://schema.infor.com/InforOAGIS/2"
xmlns:cust="http://schema.infor.com/InforOAGIS/Custom"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://schema.infor.com/InforOAGIS/2
http://schema.infor.com/2.7.0/InforOAGIS/BODs/Developer/SyncProductionOrder.xsd
http://schema.infor.com/Custom Custom\UserAreaExtensions\Customer\CustomHeader.xsd
http://schema.infor.com/Custom Custom\UserAreaExtensions\Customer\CustomDetail.xsd" releaseID="9.2"
 versionID="2.7.0">
```

Important: In PA tool, the maximum length of the field "Schema Location" is 256 characters.

You can also use some tokens that will be translated to schema file names in runtime. This schema location string will give the same result as above:

```
http://schema.infor.com/2.7.0/InforOAGIS/BODs/Developer/<s> cust:
Custom\UserAreaExtensions\Customer\<s>
```

The tokens in the given example above are **<s> cust:** and **<s>**

For more information about tokens, see "Namespaces in XML Transform Process" in *M3 Enterprise Collaborator Partner Administration Tool User Guide.*

Java Functionality

4

Data Translations

ION Mapper runtime provides built-in functionality for data translations. The Mapper database is used to store translations metadata and the ION Desk to administrate the translation data.

Data translation is normally required between an in-house system and an external system. Data translation allows string data going to (outbound), or received from (inbound) a system to be translated from one string form to another.

For example, between two partnering companies, the same things may have different representations like units of measure codes. One company uses a string data code "STK" while the partner company uses the string data code "PCS". Both codes refer to the same single unit of measure which means "styck" for Swedish and "pieces" for English.

Data Translation Configuration in ION Mapper

Data for transformation is defined and published in ION Mapper to allow the user to identify which elements need to be translated. Data translations are built into the mappings using Java code.

You feed in the data translation the application context for the message, the message identification, and the message direction (inbound or outbound). For every translation, you have to enter the rest of the data translation identification and the data to translate. This is the metadata used for translations.

The data translation metadata is automatically created in the Mapper database when a mapping is published, and when a published mapping is republished. Mappings are published in order to make them available for use from ION Desk or from the MEC Partner Administration Tool.

When a mapping is published, it becomes available in ION Desk, or from the MEC Partner Administration Tool as well as its Data Translations.

When a mapping is unpublished, the data translations records are automatically removed, or acquires a "deleted" state if you have entered any translation data for it, to prevent accidental deletion of business data.

When you update the data translation definitions in the mapping, the data translation records are also updated accordingly when re-publishing.

Translation Data

The values of the elements that need to be checked and replaced are defined with all published mappings. When Translations Data (values) for the elements are saved, the values are automatically saved in Mapper Database. Every time the translations data is executed, Mapper Database is not accessed, rather it only takes a copy of the database in the memory.

The first time a mapping containing data translations is executed the data translations and their values are loaded from the Mapper Database into memory. For new translation data to be used, you must reload them. For information on how to reload translation data, see documentation for ION Desk or for MEC Management Pages.

You must reload the translations data for the ION Mapper Runtime to load the translation values in the memory.

Translation Logic

For every data translation, the following logic is applied:

1. Try to find translation data for the given application context and the given partner.
2. Try to find translation data for the given application context and the generic (blank) partner.
3. Try to find translation data for the top application context (logical id) and the generic (blank) partner.
4. Use the input value.

Data Translations administration

The values of the elements that need to be checked and replaced are defined with all published mappings in **ION Desk** for ION and in **Partner Administration Tool** for MEC.

Translations Data (values) for the elements are saved in Mapper Database. Every time the translations data is executed, Mapper Database is not accessed, rather it uses a copy of the database in the memory.

MECDataTranslator

`MECDataTranslator` is a class that is used for translating data between the back end application and business messages. **`MECDataTranslator`** uses translation data from the Mapper database. Metadata for translations is automatically created in the Mapper database when a mapping is published and when a published mapping is (re-)published.

`MECDataTranslator` is used in Java functions in the mapping.

There is also an older class for data translation, **`DataTranslator`** . The behavior for this class is similar as for the class **`MECDataTranslator`**. However the translation data is not stored in the Mapper Database but in the M3 database. Hence, the class **`DataTranslator`** can only be used with M3. The functionality for this class is not described here. Please refer to Javadoc for MEC Utilities for more information on **`DataTranslator`**, and also for **`MECDataTranslator`**.

Using MECDataTranslator

In ION Mapper, use these procedures in a mapping to use **`MECDataTranslator`** class.

❑ Create an Object constant

___1 Open a mapping in the Mapping Editor.

___2 From the Mapping Console, go to the Variables and Constants tab.

___3 Right-click on the table and select **Create Constant**.

Important: It must be a constant and not a variable.

___4 Type a descriptive Name for this constant, for example "**`myMapping`**".

___5 Select an **`Object`** as data type.

Note: If Object type is not available, add it through Window > Preferences > ION Mapper > Type Definitions.

___6 Type the value **`this`** for this constant, .

___7 Type a brief description.

It is recommended that you use a global **`MECDataTranslator`** variable and instantiate the object in the first Java function where a data translation is performed or in a generic initialization function. This instantiation must be performed before the first data translation is done. Otherwise, the generation of data translation metadata in ION Mapper will not work.
There is no need to create several **`MECDataTranslator`** instances since you normally use the same message standard, message version, and message type (as given for the constructor) in the whole mapping.
However, it is possible to use several local **`MECDataTranslator`** instances and even mix them with a global **`MECDataTranslator`** instance, provided that the name of the object /variable for the local and global **`MECDataTranslator`** are not the same.

❑ Create a global MECDataTranslator variable

___1 Open a mapping in the Mapping Editor.

___2 From the Mapping Console, go to the Variables and Constants tab.

___3 Right-click on the table and select **Create Variable**.

___4 Type a descriptive Name for this constant, for example "**`dt`**".

___5 Select a **`MECDataTranslator`** as data type.

Note: If it is not available, add it through Window > Preferences > ION Mapper > Type Definitions.

___6 Type the value **`null`** for this variable.

___7 Type a brief description.

This variable is accessible within all Java functions in the mapping.

Note: Since the Java code is scanned when publishing or (re)publishing a mapping, some restrictions on how to use **`MECDataTranslator`** apply.

Here is an illustration on how to use **`MECDataTranslator`** in Java user functions:

Important: Java code for translations must use this pattern to avoid errors when generating and publishing the mapping. That is, hard coded strings must always be hard coded strings in the parameter list.

Note: For ION, use this example, where the application context is not used (logical id is always "Generic").

For MEC, you can use any application context, given the hard coded four levels logical id, tenant id, accounting entity id and location id. In MEC you can for example get tenant id and accounting entity id from an incoming BOD.

```
// Instantiate a MECDataTranslator object, used in the whole mapping
String logicalId = "Generic";
String tenantId = null;
String accountingEntityId = null;
String locationId = null;
dt = new MECDataTranslator(MyMapping, logicalId, tenantId, accountingEntityId, locationId,
"EDIFACT", "D01B", "ORDERS", 'O');

// Perform the actual translation (from the variable iCustPartyID to the variable e01_3039)
e01_3039 = dt.translate("g002/NAD", "3039", iCustPartyID, "3035", "BY", "Customer party ID");
```

Java Code Example and Concepts

Example for "dt"

Take a closer look at the following Java code example previously used here : "Using MECDataTranslator" on page 67

```
// Instantiate a MECDataTranslator object, used in the whole mapping
String logicalId = "Generic";
String tenantId = null;
```

```
String accountingEntityId = null;
String locationId = null;
dt = new MECDataTranslator(MyMapping, logicalId, tenantId, accountingEntityId, locationId, "EDIFACT",
 "D01B", "ORDERS", 'O');

// Perform the actual translation (from the variable iCustPartyID to the variable e01_3039)
e01_3039 = dt.translate("g002/NAD", "3039", iCustPartyID, "3035", "BY", "Customer party ID");
```

Instantiate

The first six lines instantiates the global MECDataTranslator variable "**dt**". This code can be placed in its own Java function within the mapping, if desired. However, it must be placed before any data translation is done. You must only instantiate a global MECDataTranslator variable once within a mapping.

Application context for data translations is not supported in ION. Hence,

- logical id is hard coded to "**`Generic`**".
- tenant id, accounting entity id, and location id are set to **`null`**.

If any other application context is used, the translation data given in ION Desk will not work. For MEC you can set the full application context in any way appropriate for the message. The following must be given as hard coded strings in the constructor's parameter list.

- message type - "EDIFACT"
- message version - "D01B"
- message type or document - "ORDERS"
- message direction - given by a hard coded character

  In this example, it is an outbound message since the character '**`O`**' is given.

  For inbound messages use the character '**`I`**' instead

Translate

The actual data translation is done after the MECDataTranslator variable **dt** is instantiated. The method "**translate**" is used to translate data, both for outbound and inbound messages. The following must be given as hard coded strings in the translate method's parameter list:

- parent path - "g002/NAD"
- value path - "3039"
- conditional path - "3035"
- conditional value - "BY"
- description - "Customer party ID"

In this example, the customer party id from the inbound BOD is given by the Java function's input parameter **`iCustPartyID`**. The translated value is assigned to the Java function's output parameter **`e01_3039`**, that should be mapped to the output EDI document.

In a normal mapping, there are numerous data translations made in several Java functions. Since the MECDataTranslator variable **dt** is global, and instantiated first in the mapping, you only need to call the translate method to perform the actual data translations.

If you want to check whether a data translation was actually performed or not, use the boolean MECDataTranslator method **`isTranslated()`** after the call to the translate method.

Data translations can be shared

Data translation can be shared by several mappings. For example, if you use the data translator example given above in two mappings, or maybe two mapping versions, the translation data given in ION Desk (or in the Partner Administration Tool for a MEC mapping)will be used by both mappings, or mapping versions.

In the example, the data translations will be unique for the **`EDIFACT D01B ORDERS`** message.

If you create another mapping for invoice, and follow the pattern above, there will be other data translations defined that are unique, for example, the **`EDIFACT D01B INVOIC`** message.

While this is a good practise, like in translating qualifiers, you may want to use generic data translations. For example, code lists such as language code, currency code, and others. Create another global MECDataTranslator variable for this purpose.

- **Variable** - give the variable name "**dtGen**", or any other suitable name.
- Instead of a specific message type, message version, message type (document) and/or parent path, give the hard coded strings "Generic" instead.
- **Value path** - must be given to identify the data to translate, for example, "**`LanguageCode`**" or "**`CurrencyCode`**".

  Note: Conditional path and value are normally not used for generic data translations and can be empty strings ("").

Since you now have the following two global MECDataTranslator variables, you can use either of them in the whole mapping when performing data translations:

- **dt** for message specific data translations, and
- **dtGen** for generic translations

Example of the Java function "initiate"

This is an example of the Java function "initiate" in an inbound EDI mapping for ION using both message specific data translation (**dt**) and generic data translation (**dtGen**):

```
String logicalId = "Generic";
String tenantId = null;
String accountingEntityId = null;
String locationId = null;
dt = new MECDataTranslator(MyMapping, logicalId, tenantId, accountingEntityId, locationId,
"EDIFACT", "D96A", "ORDERS", 'I');
dtGen = new MECDataTranslator(MyMapping, logicalId, tenantId, accountingEntityId, locationId,
"Generic", "Generic", "Generic", 'I');
```

Example of a generic data translation

This is an example of a generic data translation of a language code mapped from the input message to the input parameter **`iLanguageCode`**. The translated value is assigned to the output parameter

oLanguageCode that, for example, can be mapped to the root element attribute **languageCode** for BOD.

```
oLanguageCode = dtGen.translate("Generic", "LanguageCode", iLanguageCode, "", "", "Language
Code");
```

Data translations are bi-directional

For example, if you have defined data translations for an outbound mapping from **PurchaseOrder** to **INVOIC D01B ORDERS**, and use the same data translation definitions in an inbound mapping from **EDIFACT D01B ORDERS** to **SalesOrder**

The application values and partner values given in ION Desk will be used for both mappings, but in opposite directions.

Given this, each application value for a data translation must be unique per partner, or the generic "partner" since you can only have one "**to**" value defined per "**from**" value.

In the same manner, each partner value must be unique per partner, or the generic "partner".

MEC Functionality

5

M3 API

An M3 API function, or M3A function for short, is used to access M3 functionality through calls to the M3 API while controlling a loop. The main purpose is to let you create repeating elements in the output document, based on the result returned from the M3 API call.

An M3 API function call can fail and the action taken is controlled by the **ErrorHandling** attribute which can have following values:

- **Exit map on M3 NOK** - the current mapping operation is aborted.
- **Exit loop on M3 NOK** - the current loop operation in progress is aborted.
- **Ignore M3 NOK** - the error is ignored.

Database

Database object is used to access or modify data in an SQL Server database. There are two Database Object categories; Table and Stored Procedure.

Stored Procedures

Stored procedures have a singe operation, that is, **Execute**. This operation can return output parameters and/or a result set. A stored procedure that returns a result set can control a loop. Any mandatory parameters to the stored procedure must be mapped.

Output parameters that start with the '@' symbol are returned once per call, while other parameters return a result set.

Table Type Database Object

The table type of Database Objects can provide access to different types of standard SQL operations, listed below:

- **Create** - use this to create a record in a table. All fields with NOT NULL attribute are marked as mandatory and must be mapped.
- **Update** - use this to update a record in a table. Primary key fields are mandatory and must be mapped. Any number of non-key fields can be mapped.
- **Delete** - use this to delete a record from database. Primary key fields are mandatory and must be mapped.
- **Get** - use this to retrieve a single record from a table. Primary key fields are mandatory and must be mapped.
- **List** - use this to retrieve one or more records from database based on the mapped fields. This operation can control a loop.

  Use the Database widget wizard to view database tables with primary key, like when you add a database object to the mapping as follows:

  - Table with no primary keys (typically views) are listed but no transactions will be available.
  - Tables with no additional columns, apart from the PK, will not have the UPDATE transaction available.

    This means that you are now allowed to change the unique identity of this database record.

Database operations can fail and, in this case, will provide an exit function similar to **NOK** handling for M3 APIs. The action taken is controlled by Exit Type parameter which can have following values:

- **IGNORE** - the error is ignored.
- **LOOP** - the current loop operation in progress is aborted.
- **MAPPING** - the current mapping operation is aborted.

M3 Settings

6

M3 Company and Division

The M3 Settings function is used for setting Company (CONO) and Division (DIVI) for M3 API calls.

Sequence overview

- Place an M3 Settings function before the M3 API call
- Set the desired Company and Division by assigning variables to the Company and Division parameters
- Give the variables suitable values.

  Instead of using variables you can also, for example, use input elements containing the Company and Division.

All succeeding API calls will use the Company and Division as set by the M3 Settings function.

You can use another M3 Settings function later in the sequence to change Company and/or Division again, if needed.

Use inside a loop

If you use an M3 Settings function inside a loop that is controlled by a repetitive M3 API, the data from the M3 API controlling the loop will be fetched using the Company and Division set when entering the loop, not the Company and Division set by the M3 Settings function inside the loop. This is because the actual API call for the loop is only done once, when entering the loop.

Also note that if you execute an M3 Settings function inside a loop in the first iteration but not in the following iterations the Company and Division set by the M3 Settings function in the loop will be used for all following M3 API calls in the loop, that is even for iteration two, three, and more, and also for all following M3 API calls after the loop.

Use of default values

If you want to use default Company/Division values as defined by MEC control properties, API reference parameters, or the setting in MNS150 you do so by supplying null values for both Company and Division, for example by using the widget without any input data.

- This might be necessary if you, for example, dynamically change the Division, using the M3 Settings function, to get data from several M3 Divisions, and then you want to go back to the original settings as defined by MEC control properties, API reference parameters, or the setting in MNS150 for the last API call.
- If you want to use default Company/Division values as defined by MEC control properties, API reference parameters, or the setting in MNS150 for all M3 API calls in the mapping you do not need to use the M3 Settings function.

Variable values

The variables used for both Company and Division must have the data type **String**. Their values must therefore be enclosed within " " (double quotation marks), for example **"123"**. That is, even if Company is an integer value (see below), it should be enclosed. If **Division** is set, **Company** must also be set, or else a runtime error will occur. In the case where only Company is set, the default "blank" Division will be used.

- **Company** must be an integer value between 0 and 999.

  The parameter value can be max 3 characters long and is trimmed before the integer check. "1", " 1 ", " 1", " 01" and "001" are thus all valid companies and will be used as it is, including blanks, when establishing a new API connection to M3.

- **Division** must be an alphanumerical value with max length 3.

  The parameter value is NOT trimmed.

  Default value is an empty string (""). "A", "A ", "A ", " A " and " A" are thus different divisions in M3.

Mapping Management

7

Mapping Elements Overview

At the start, a mapping contains only input and output document elements. Your task is to describe how the input document is translated to the output document by defining the complete transformation process. This is done by adding a sequence of functions and linking them to the input and output document elements.

The function sequence is executed from top to bottom every time an XML document is transformed. In this process, data values will be fetched from linked input elements, processed, and then sent to linked output elements, constituting the output document.

Here is an overview of this process:

- Create a transformation by adding a sequence of function elements in the space between the input and output document trees.
- Create links between schema elements and function elements.
- Create links between function elements.
- Create links between variables or constants, and functions or documents elements.

Note: For MEC, output document is not mandatory. For example, in many inbound M3 messages no output document is used since the data is stored in M3 by calling M3 APIs in the sequence, and no response message is required.

Function Elements Overview

Functions define how data from an input document will be transformed and sent to the output document.

Using the input data, functions can be called and the information returned by the function can then be forwarded to the output document. Data from an input document can be, for example, an item number regarding a price request on a product. Other sources can also be used in functions.

Functions can be predefined or user-defined. For user defined specific transformations, use Java code.

To define the transformation sequence, drag the elements from the Functions palette and drop them in the space between the input- and -output schema trees. This will create a stack of elements executed from top to bottom.

Note:
- When connecting elements, the Mapping Editor will allow you to drop elements only where it is acceptable.
- To complement the "Move Up/Down" menu options you may use drag and drop to move function and loop elements up and down in the mapping.
- Output data from functions can be used as input data in functions later on in the function sequence.

For more information, see "Adding Input/Output Parameters" on page 80.

Core Functions

Loop

A loop is an iteration element controlled by a link. Loop functions may contain nested loops or other function elements. Basically, loops allow you to create repeating elements in the output document based on conditions that are determined by the type of source elements of the connected link.

A loop can be dropped on the mapping area or on another Loop function, where the latter yields a nested loop.

Boolean

Boolean functions should have input parameters only. These are also known as a "loop controlling functions", because**Boolean** function controls the loop iterations based on a user defined criteria.

Note: The **Boolean** and **Java** functions are almost the same, except that **Boolean** function outputs a Boolean value. However, you define the criteria by writing java code in the same way.

Java

Java functions can be dropped on the mapping or a Loop. A user-defined Java function (UV) processes data in the form of input parameters and outputs the result to output parameters. Java code is used to define the processing that the function will do.

Repository

Repository Functions are reusable Java and Boolean functions stored in Function Repository which is part of a Mapper database. Reusable functions can be dropped on the mapping area or a loop.

Note

You can use this to enter textual descriptions in relevant places in your mapping.

From the Core Functions pane, select and drag **Note...** in the Mapping Editor area and drop to insert notes in the mapping. Edit inserted notices through the Mapping Editor context menu. A text editor opens where you can enter and save texts. Move your mouse pointer over these notes to display its content.

Assign

This mapping element allows you to make a direct assignment from an input document element to an output document element.

You can drag and drop an input element directly to an output element. From the Core Functions pane you can also select and drag **Assign** in the Mapping Editor area and drop it in the mapping. Then attach the input- and output elements by dragging and dropping them to the **Assign** widget.

Variables and Constants

Variables are used as intermediate data holders to transfer data to or from functions. You thus use Variables to transfer values from functions on the upper part (earlier) to the lower part (later) of the execution sequence.

Constants are data holders for fixed data values to be assigned to function input parameters.

Palette Behavior

Tools

Use the **Select** tool to select functions, parameters, elements, and links in the mapping.

- When selecting a function, parameter, or element its icon will be highlighted and all its connected widgets will also be highlighted, including the links.
- When selecting a link its connected function, parameters and/or element will be highlighted recursively.

Aside from the **Select** tool you can also use the **Links** tab in the Mapping Console to select links.

To select multiple items, you can control-click or shift-click, for example to delete many functions or parameters at once.

The **Marquee** tool works like the select tool, but with Marquee you can mark areas to select multiple items simultaneously.

The **Link** tool is used to create links in the mapping.

1 Access the Palette tool and select **Link**.

2 Click on the element, function or parameter you want to link from.

3 Click on the element, function or parameter you want to link to. A new link is now created.

Repeat step 2 and 3 to create more links.

Note: It is more convenient to create links in the mapping using the **Select** tool. Select the element, function, or parameter you want to link from and drag it to the element, function or parameter you want to link to.

You use the **Link** tool only when creating a control link from a Boolean function to a Loop. Although, this is done automatically when inserting the Boolean function into the Loop.

Functions

When adding functions in the editor pane a grey widget will be shown where it is possible to create a function.

Here are the two ways of using the functions in the palette:

1 **Click and drag** a function in the palette to the editor pane.

While holding down the mouse button, point where you want to create the selected function.

Release the mouse button to create the function.

After this operation the Select tool will always be selected in the palette.

2 **Click to select** on a function in the palette.

Move the mouse in the editor pane to create a function and click to create a function.

After this, you can create more functions of the same type without accessing the palette for each new function.

The function will be selected in the palette until you select another function or tool.

Note: This behavior is also valid for parameters.

Adding Input/Output Parameters

Use this procedure to add Input/Output parameters in a mapping. These parameters contain input or output arguments and can be dropped on a Java function or on a Boolean function.

Note: Boolean functions should have input parameters only.

1. Double-click on a mapping to open it in the Mapping Editor.
2. From the Core Functions pane, click and drag a function to add in your mapping on the Mapping Editor tab.

   Note: When connecting elements, you can only drop elements only where it is acceptable.
3. In the Parameters pane, click and drag an input or output parameter to add in a function in your mapping.
4. Right-click on a parameter and select **Properties...**

   The Properties tab is displayed.
5. Type the appropriate values for your parameter properties.

Description	Type a brief description for this parameter.
Name	Default value: InParameter1
Type	Default value: String Select the data type to use. **`String, int, long, float,`** and **`java.math.BigDecimal`**

6. Select File > Save to save the parameter in to your mapping.

If you want to link an input Document element to a function, or if you want to link an output Document element to a function, a more convenient way to create parameters is just to drag and drop the document element to the function. A parameter will then automatically be created with the same name as the document element. A link to the parameter will also be created at the same time.

Adding a Loop Function

A loop can only be dropped on a mapping or on another Loop function where the latter yields a nested loop.

For more information, see "Function Elements Overview" on page 77.

1. Double-click on a mapping to open it in the Mapper Editor.
2. From the Core Functions pane in the Palette tab, select and drag **Loop** in the Mapper Editor tab to add it in your mapping.

Note: When connecting elements, you can drop elements only where it is acceptable.

3 Drag and drop the circular function element to the space between the input and output document trees.

Note: When connecting elements, the Mapper Editor will allow you to drop elements only where it is acceptable.

4 Connect a controlling link for your loops.

A loop can be controlled by a repeating input document element, a UBJ function, or a repeating external function (Database or M3 API), see the table below.

A loop also allows you to create repeating elements in the output document by linking the loop to a repeating output document element.

As an example, you can drag a repeating input document element to a loop and then drag the loop to a repeating output document element (you can also drag the output element to the loop). Now, the output repeating element will repeat as many times as the input repeating element does.

The type of source element of the link that controls the loop will define the loop condition:

Loop Condition	Details
If source is a repeating input document element then the loop may:	• Execute for each occurrence of the element (WHILE_TRUE, Execute loop FOR EACH element) • Execute once if the element exists (IF_TRUE, Execute loop ONCE if element exist) • Execute once if no element exists (IF_FALSE, Execute loop ONCE if NO element exist)
If source is a UBJ function then the loop may:	• Execute while the function returns true (WHILE_TRUE, Execute loop WHILE function is TRUE) • Execute while the function returns false (WHILE_FALSE, Execute loop WHILE function is FALSE) • Execute once if the function returns true (IF_TRUE, Execute loop ONCE if function is TRUE) • Execute once if the function returns false (IF_FALSE, Execute loop ONCE if function is FALSE)

Loop Condition	Details
If source is an external function (Database or M3 API) then the loop may:	• Execute for each record returned by the function (WHILE_TRUE, Execute loop for EACH record returned by a list program) • Execute once, if the function returns a record (IF_TRUE, Execute loop ONCE for the record returned by the program) **Note:** If the source is a Database Object, it will always be a Boolean condition, for example: **`DatabaseObjectName.Operation = true`**

5 Right-click on a loop and select **Properties...**

The Properties tab is displayed.

6 Select the appropriate condition for your loop.

Note: You need to expand the Condition property to be able to select the **`LoopExecution`** value.

7 Type the appropriate values for your parameter properties.

Condition

Description	Type a brief description for this loop.
Name	Default value: Loop

8 Select File > Save.

Adding Java Function

Java functions can be dropped on a mapping or a Loop.

1 Double-click on a mapping to open it in the Mapping Editor.

2 From the Core Functions page, click and drag **Java**.

Drop it in the space between the output document trees to add a **UVJ** function.

Note: When connecting elements, you can drop elements only where it is acceptable.

3 Define the processing to be done by the function.

For more information, see "Editing Java and Boolean Functions" on page 83.

4 Add links to input/output parameters to receive/output through **UVJ** function.

With the parameters, you can now receive and output data through the links.

5 Save the function in the Java Editor and in the Mapping Editor.

Important: Changes must be saved in both Java Editor and Mapping Editor.

Adding Boolean Function

Use this procedure to add boolean function in a mapping. Also known as a "loop controlling function", **UBJ** function controls the loop iterations. This function can be dropped on a loop. There can only be one boolean function within a loop and it must be the first in the loop's sequence.

Note: The methods for **UBJ** and **UVJ** functions are almost the same, except that the **UBJ** function outputs a Boolean value, as well as output parameter values. Although, you define the criteria by writing Java code in the same way.

For more information, see "Function Elements Overview" on page 77.

1 Double-click on a mapping to open it in the Mapping Editor view.

2 From the Core Functions pane, click and drag **Boolean**. Drop it in the space inside a loop to add a **UBJ** function.

Note: When connecting elements, you can drop elements only where it is acceptable.

3 Add input parameters to receive input data.

With the parameters you can now receive input data through links.

4 Define the processing to be done by the function.

For more information, see "Editing Java and Boolean Functions" on page 83.

5 Save the function in the Java Editor and in the Mapping Editor.

Important: Save in both Java Editor and Mapping Editor in order to not lose the changes made in your mapping.

Editing Java and Boolean Functions

Use the instructions here to edit Java and Boolean functions.

Important: The **Mapping Editor** and the **Java Editor** are two separate entities.

When you have edited and saved your code in the Java Editor and then switch back to the Mapping Editor it will become dirty because it detects that java code was edited. At this stage you MUST perform a save in the Mapping Editor so your code changes are included with the mapping. It is not enough to only save the code in the Java Editor, you must do both.

1. Double-click on a mapping to open it in the Mapping Editor.
2. Right-click on a Java or Boolean function and select **Edit Java Code...**

   A new tab opens with a name format: <*Function_name*>**.java**

   This is the standard Eclipse Java editor.
3. Write your Java code.

   Use the standard content assist and formatting features in the body of the method.

   Important: Do not type any code outside of the named method.

   All the surrounding code is generated each time you edit the java function so any changes here will be lost.

   This includes all automatic imports that the Java editor may do. If a class is not imported by the default imports you have to write the Java package name before the class name, for example **java.util.ArrayList**
4. Save the Java code in the Java editor, by **File > Save** or **CTRL-S**.

Getting help from Content Assist commands

To get help from the Content Assist command (Ctrl+Space) you must select Java Proposals.

1. On Eclipse menu, select Window > Preferences > Java > Editor > Content Assist > Advanced.
2. In Default Proposal Kinds pane, select the Java Proposals check box and click Apply, then OK.

By default, Java Proposals is not selected. If you like you can also check other proposal kinds and also how these are cycled through when repeatedly invoking content assist.

Important notes on saving a mapping and java code

- If the mapping was previously saved, you may notice that the mapping becomes "dirty" meaning unsaved, as indicated by an asterisk before the mapping name in the mapping tab. This is because when you save the Java code in the Java editor the code is inserted into the mapping, but the mapping is not saved.

  An invisible .java file is also created in your project but that is not used by the mapping. To save your Java code it is very important to also save the mapping.

- If you add, remove, or rename function parameters and/or global variables or constants, you have to close the Java editor tab (do not forget to save first) and edit the Java code again to reflect these changes.

  The Java skeleton is generated every time you select **Edit Java Code...** for a function.

Getting Javadoc for MEC Utilities

In the standard Eclipse Java editor you get Javadoc for standard Java classes, for example String. Javadoc is also included for the following MEC Utility classes:

- BundesbankCountryCode (for financial mappings)
- CATToolbox (for Caterpillar mappings)
- DataTranslator (for M3 BE mappings)
- EDIDateConverter (for M3 BE EDI mappings)
- FormatNumber
- IONApplicationArea (for ERP BOD mappings)
- IONToolbox (for ERP and ION BOD mappings)
- MECDataTranslator
- MessageCounter
- PersistentObject
- SortingStructure
- StringReplacer

When you pause on a class, method, or constant you get a tooltip with the Javadoc. You can press F2 or click in the tooltip to set focus on the Javadoc. When the Javadoc is in focus some buttons are shown:

- Back
- Forward
- @ Show in Javadoc view
- Open Declaration (opens the source code, if available)
- Open Attached Javadoc in a Browser

Click on the @ button to open the Javadoc view, where Javadoc for the currently selected class, method, or constant is shown.

Click on the **Browser** button to open the Javadoc in a separate Browser view. For example, to view Javadoc for the class MECDataTranslator, pause on a class reference (MECDataTranslator), the constructor, or a method, for example translate(). Press F2 and then click on the browser button. Javadoc for the MECDataTranslator class, including its constructors and methods, is displayed in a separate Browser view. Code examples for MECDataTranslator are included in the Javadoc for the class.

Creating and Adding Repository Function

Repository Functions are reusable Java functions (UVJ and UBJ) stored in Function Repository which is part of a Mapper database. Reusable functions can be dropped on a mapping or a loop.

For more information, see "Function Elements Overview" on page 77.

Note: When connecting elements, the Mapper Editor will allow you to drop elements only where it is acceptable.

Before you start Define a Mapper Database connection to be able to use repository functions.

❑ Create repository functions

___1 Double-click on a mapping to open it in the Mapper Editor.

___2 From the Core Functions pane, click and drag the **Boolean** or **Java** function element. Drop it in the space between the input and output document trees to add a **UBJ** or **UVJ**function.

___3 Add parameters to your function.

___4 Name the function and its parameters, then edit the Java code.

___5 Save the mapping with the new functions.

___6 Right-click on the function, select **Save to Repository...**

The Mapper Database window is displayed.

___7 Select a database location to contain this Repository Function and click Finish.

___8 Confirm to insert a new repository function into the repository, or to overwrite an existing repository function.

At the prompts, review the function version to save to the database and click OK.

Note:

- The version for a new repository function will always be 1.0.
- The category for a new repository function will always be Default.

❑ Add repository functions

___1 Select a mapping to use and open the Palette tab.

___2 Drag the **Repository** function element, and drop it to the space between the input and output document trees.

___3 The New Reusable Function window is displayed.

___4 Select the Mapper Database location to use.

___**5** Click Next.

___**6** Select a reusable function from the list.

___**7** Click Finish.

You can also click Next to view the input and output parameters before pressing Finish.

When completed, the selected function is inserted in the mapping complete with input output parameters that you can now link to other elements.

❑ Change a Repository Function Category

If you have a repository function in your mapping you can change its category.

___**1** Right-click on the repository function and select **Properties...**

___**2** Change the Category value in the Properties tab.

___**3** Save the mapping.

___**4** Right-click on the repository function and select **Save to Repository...**

The Mapper Database window is displayed.

___**5** Select a database location to contain this Repository Function and click Finish.

___**6** Confirm to overwrite an existing repository function.

❑ Insert a new version of a Repository Function

If you have a repository function in your mapping you can create a new version of it.

___**1** Right-click on the repository function and select **Properties...**

___**2** Change the Version in the Properties tab.

___**3** Save the mapping.

___**4** Right-click on the repository function and select **Save to Repository...**

The Mapper Database window is displayed.

___**5** Select a database location to contain this Repository Function and click Finish.

___**6** Confirm to insert a new repository function into the repository.

Deleting a Repository Function

Important: Delete from database has no undo function. Verify the category and version of a repository function before you proceed with delete.

❑ Delete from database

___**1** In the Mapping Editor, right-click on a repository function and select **Delete from Repository...**

The Delete Function from Repository window is displayed.

___**2** Select a database location containing the repository function to delete.

___**3** Optional: Test the database connection.

___**4** At the successful Connectivity Test prompt, click OK.

Note: This action will only test for the connectivity to the server machine.

___**5** Click Finish.

___**6** At the confirmation prompt, click OK.

Note: If you want to delete another version of the repository function, you have to change the version of repository function first. The repository function is identified by its name and version.

❑ Delete from workspace

___ In the Mapping Editor view, right-click on a repository and select **Delete**.

The selected function is now removed from workspace.

Links Overview

When the sequence of function elements are executed, data is transformed and transferred from the input document to the output document. The source and target of this data transfer is defined by connecting document elements and function elements with Data Links. In addition to Data Links, you can also control the iteration of a loop function using Control Link.

Here are element connections details when adding links:

Element	Connection Details
Document elements	Elements with multiple multiplicities (repeating elements) can be connected to (input) or from (output) a Loop. This means that the element multiplicity will control the iteration of the loop (input element) or the population of an output document (output element) Elements with single multiplicity can be connected to (input element) or from (output element) input/output parameters.
Boolean function	Can be connected to a loop.

Element	Connection Details
Variables	Can be connected to input parameters or to output parameters.
Constants	Can be connected to input parameters.

Adding links between function elements

Data Links and Control Links can be made between function elements as follows:

- Data Links can be made between UVJ- and Repository functions output parameters to input parameters of UBJ-, UVJ-, and Repository functions.
- A Control Link can be made between a function (M3 API or Database) and a loop. This means that the occurrence of output records controls the number of times that the loop will iterate.
- A Control Link can be made between an UBJ function and a loop. This means that the Boolean result controls if the loop shall iterate or not.

Assigning variables or constants to function elements

Variables or constants can be assigned to function elements as follows:

- Variables can be assigned to input/output parameters of UVJ-, UBJ- and Repository functions.
- Constants can be assigned to input parameters of UVJ-, UBJ- and Repository functions.

Adding and Removing Links

❑ Add links

___**1** Open a mapping on Mapping Editor.

___**2** Click and hold on the source document element and drag to connect to the target function element.

Important: The editor is context sensitive to connections and will only allow you to create meaningful connections. Refer to the following limitations:

- Crossing links to the output document is not allowed. Crossing output links will generate erroneous output documents. Crossing links from the input document is allowed.
- Control Links can be made from a repeating document element to a loop function and the other way around. This means that the occurrence of a repeating document element controls the number of times the loop will iterate.

- Control Links can also be made from a loop to a repeating output document element. This means that the number of times the loop iterates controls the occurrence of the repeating element.
- Data Links can be made from document elements to input/output parameters of UVJ- and Repository functions or the other way around.

❑ Remove links

___ Select a connection, right-click > Delete.

Or, select a connection and click Delete.

Note: When you delete a link between a variable/constant and a mapping element, only the connection is removed.

To completely remove the variable or constant, go to the Variables and Constants tab in the Mapping Console View.

Adding and Deleting Variables and Constants

Use these procedures to manage variables and constants.

❑ Add variables and constants

___**1** Open a mapping on Mapping Editor.

___**2** Right-click on a blank space in the Mapping Editor, select Edit Variables and Constants.

The Variables and Constants tab in the Mapping Console is opened.

___**3** Right-click on a blank space in Variables and Constants tab, select any of the following:

- **Create variable** - select this to create a new variable with a default name **`NewVariable1`**.

  A V-icon indicates a variable. Assigning values to Variable are optional.
- **Create constant** - select this to create a new constant with a default name **`NewConstant1`**.

  A C-icon indicates a constant. Assigning values to Constant are mandatory.

___**4** Fill out the property details: Value Type, Value, and Description.

___**5** Drag the variables or constants from the tab to connect to a parameter element in the Mapping Editor.

Note: A unique index is added to the default variable/constant name.

❑ Delete variables and constants

Use this procedure to delete variables and constants.

Note: If you delete a link between a variable/constant and a mapping element, only the connection is removed and not the variable/constant.

___**1** Access the Variables and Constants tab in the Mapping Console View.

___**2** Select a variable or constant, right-click > Delete.

The selected variable/constant is removed from the current mapping.

You can also select more than one variables and/or constants and delete these simultaneously. Right-click on any of the selected items > Delete. The selected variables and/or constants are removed from current mapping.

___**3** Save your mapping.

Note: Click in the editor pane to undo delete action on variables/constants.

Editing Mapping Element Properties

Use this procedure to edit mapping elements in Eclipse Properties view.

The information in the contextual Properties tab change depending on your currently active selection in Mapping Editor View. It can be properties on functions, elements, or mapping. In this procedure we will use the Mapping Loop as an example.

1 Select a mapping to use.

2 In the Eclipse Mapping Editor, right-click on a loop to edit and select **Properties...**

The Properties tab is displayed.

3 Select and expand the Condition property.

4 On **LoopExecution** property, select the corresponding Value to use.

- Execute loop FOR EACH element
- Execute loop ONCE if element exist
- Execute loop ONCE if NO element exist

5 Select File > Save.

Restructure Function

8

Restructure Overview

This chapter guides you on how to use the Restructure function using three examples. Interleaved with the examples are references to in-depth technical details. For more information, see "Restructure Concepts" on page 103.

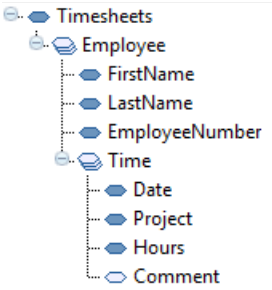
When mapping data between two different documents, the document structures are not always the same. While the input document contains all data needed for the output document, it is structured in a way different from what is required for the output document. Here are three example scenarios that we will use to describe this function.

- Example 1
- Example 2
- Example 3

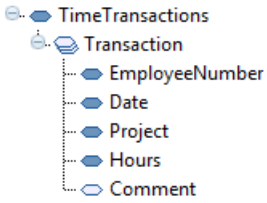
Example 1

We will use time reporting as an example where we need to restructure data in different ways.

This is the structure of the input data:



In this example, we want the time transactions in a flat structure:



This can be done using manually created standard Java code in Java functions. However, the task can be complicated and technically advanced. This is especially true if the output document requires a lot of nested loops with grouping and maybe summing, as in our following examples. The mapping will be cluttered with "technical" functions like "**`AddToRecord`**", "**`AddToArray`**", "**`SortArray`**", "**`CheckBreak`**", "**`GetFromArray`**", and more.

The Restructure function does all these without the need to create a single line of Java code. At runtime, all the processing is done in a similar but more complex way. It is like you had solved the problem with a manually created Java code in the mapping.

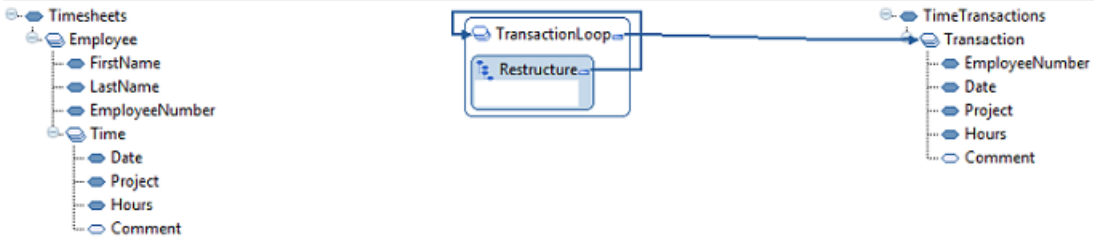
For more information, see "**The Restructure Function**" topic in Restructure Concepts.

1 Create loop

a We need to loop through all *time* transactions to create the repetitive *Transaction* element in the output document. Start by creating a loop that controls the output element *Transaction*.

For more information, see "**Defining Output data**" topic in Restructure Concepts.

b Since we want to restructure the input data we also insert a Restructure function in the loop. It will automatically control the loop.



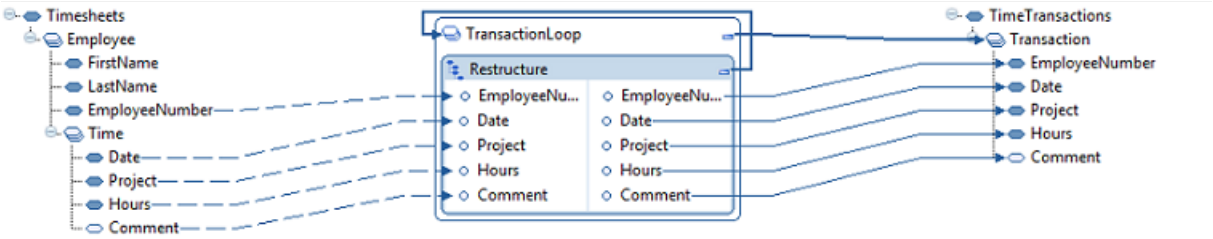
2 Select Data

Drag and drop all data elements we want to use from the input document to the Restructure function.

For more information, see "Defining Input Data" topic in Restructure Concepts.

At this stage, we do not need to think about the repeating elements in the input document.

Link the corresponding output parameters to the output document.

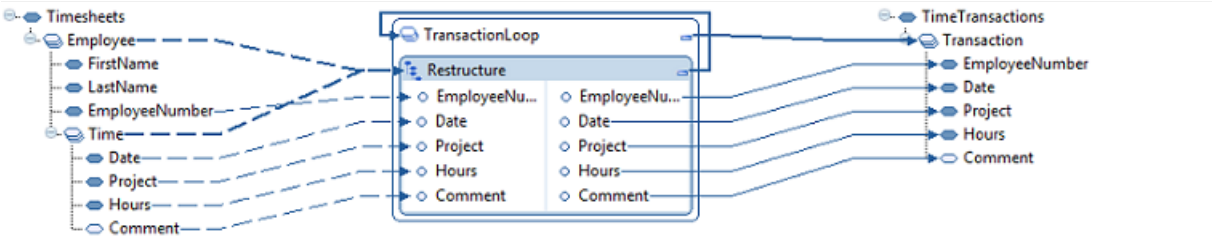


3 Select input loop levels

Now, we need to "tell" the Restructure function which repeating elements in the input document we want to loop on when reading the input data. In this case, we want to do the following:

- Loop on *Employee*, since we want data for all employees.
- Loop on *Time*, since we want data for all time transactions per employee.

Element	Movement
both *Employee* and *Time*	• from the input document • to the Restructure function's header.



4 Sort the data

To restructure the data, we need to sort it and decide the sorting sequence.

For more information on sorting, see the "**Sorting**" topic in Restructure Concepts.

Sort by setting the property **Sorting Sequence** for the input parameters.

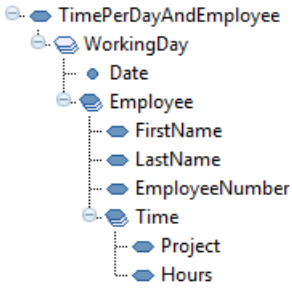
Input Parameter	Sorting Sequence
EmployeeNumber	1
Date	2
Project	3

Now, we can save the mapping to the mapper database, publish the mapping, set up an agreement, and more. Then, test the mapping.

Example 2

Our second example is a little bit more complicated. Using the same input structure, we now want to group the time transactions per day (date) and employee.

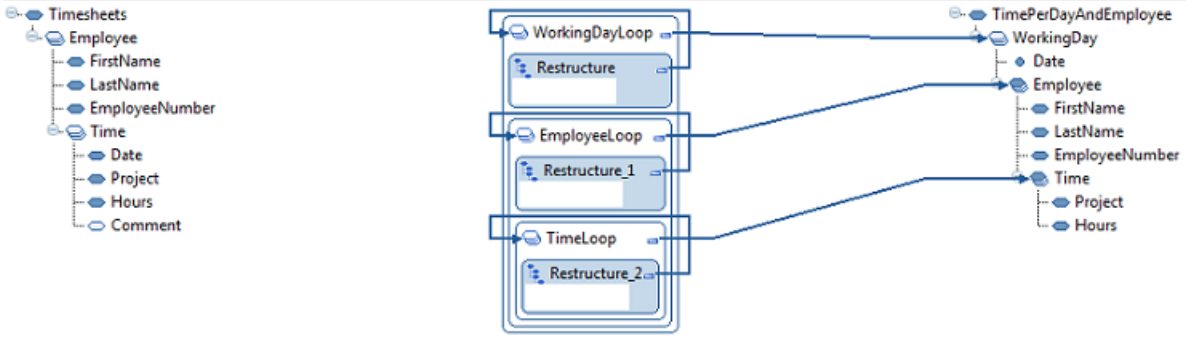
For more information on grouping see, **Grouping** topic in Restructure Concepts.



1 Create loops

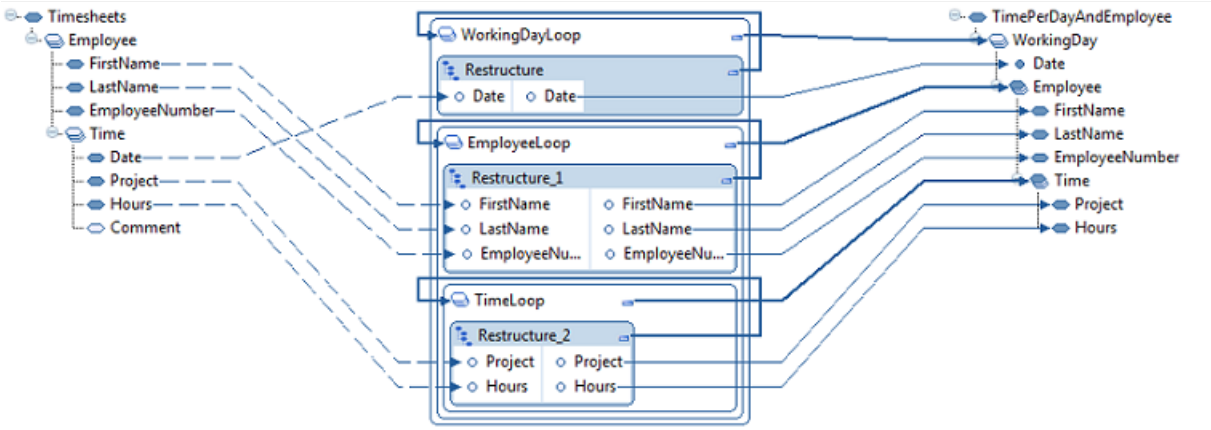
In this example, the loops will be for controlling the three repetitive elements in the output document.

The nested loops are controlled by Restructure functions.



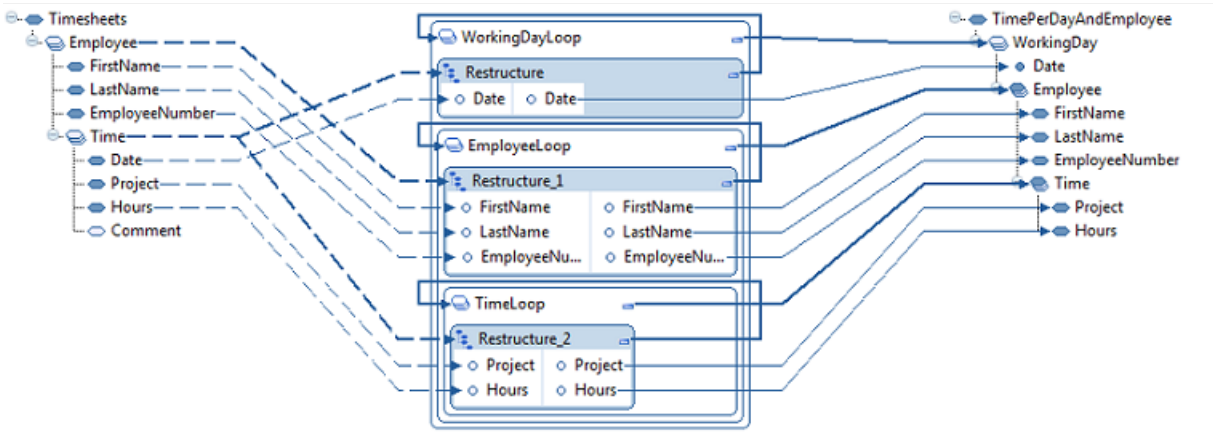
2 Select data

Drag and drop all data elements we want to use from the input document to the Restructure functions and from the Restructure functions to the elements and attribute in the output document.



3 Select input loop levels

Element	Movement
Time	Objective: **To get all the dates** Drag and drop the *Time* element, to loop on this element • from the input document • to the first (outmost) Restructure function's header
Employee	Objective: **To get all the employees** Drag and drop the *Employee* element to loop on this element • from the input document • to the second (middle) Restructure function's header
Time	Objective: **To get all the time transactions per employee and date** Drag and drop the *Time* element to loop on this element • from the input document • to the last (inner) Restructure function's header



4 Sort and group the data

Set properties for the input parameters to get the correct sorting and grouping.

- **In the first Restructure function**:

  This will make the outmost restructure loop create one *WorkingDay* output element per unique date for all time transactions.

Property	Setting
Sorting Sequence	**1** for *Date*
Is Grouped	**true** for *Date*

- **In the second Restructure function**:

  This will create one Employee element per unique combination of first name and last name.

Property	Setting
Sorting Sequence	1 for *FirstName* 2 for *LastName*
Is Grouped	**true** for *LastName*

- **In the last Restructure function**:

  This will sort the time transactions on project per employee and date.

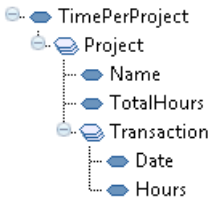
Property	Setting
Sorting Sequence	1 for *Project*

If we want to avoid getting duplicate transactions for a project per employee and date, we can set the following in the last (inner) Restructure function:

Property	Setting
Is Grouped	**true** for *Project*
Is Summable	**true**

Example 3

Here, in our last example, we want to create a time summary per project with the option to also show the transactions per date.



Time summary per project

1 Create loop

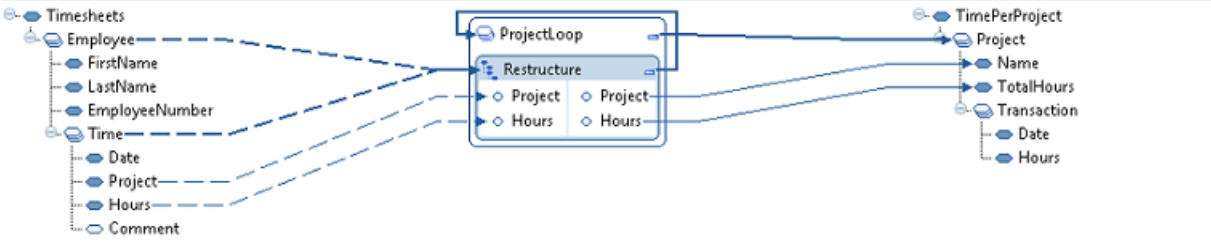
Create a loop controlling the repetitive *Project* element in the output document.

The loop is controlled by a Restructure function.

2 Select data and input loop levels

Element	Movement
Project	• Drag and drop in the input document to the Restructure function, and • Link the corresponding output parameter to the *Name* element in the output document.
Hours	• Drag and drop in the input document to the Restructure function, and • Link the corresponding output parameter to the *TotalHours* element in the output document.
both the *Time* and *Employee* repeating elements	• Drag and drop from the input document to the Restructure function's header. This will get all time transactions for all employees.

Note: If we would forget the link from the *Employee* element, we would only get time transactions for the first employee.



3 Set and group the data

Property	Setting
Sorting Sequence	1
Is Grouped	**true**
Is Summable	**true** for *Hours*.

Now, you have created a mapping for a simple time summary per project.

Add the time transaction details

Now, we want to add the time transaction details.

1 Create a nested loop

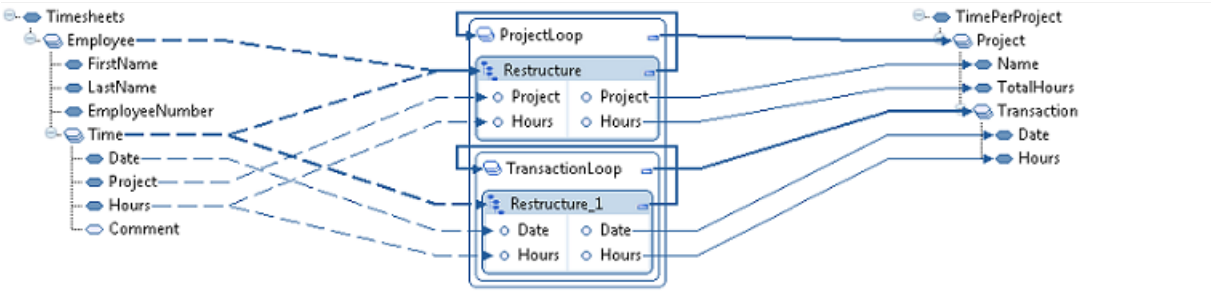
Create a nested loop controlling the repetitive Transaction element in the output document.

The loop is controlled by a Restructure function.

2 Select transaction data and input loop

Element	Movement
Date	• Drag and drop from the input document to the Restructure function, and • Link the corresponding output parameter to the *Date* element in the output document.
Hours	• Drag and drop from the input document to the Restructure function, and • Link the corresponding output parameter to the *Hours* element in the output document.
Time	• Drag and drop the *Time* repetitive element from the input document to the second (inner) Restructure function's header. This will get all time transactions per project, as handled by the outer Restructure function.

Note: We do not need to link from the Employee repetitive element again. It is already linked to the first (outer) Restructure function. So the restructure logic already knows that we want to loop on Employee in the input document. It will not do any harm to link the *Employee* element once again, though.



3 Sort the data

Property	Setting
Sorting Sequence	1 for *Date* This will sort the transactions per date, and per project.

Adding a Restructure Function in a Mapping

A Restructure function behaves as a **UBJ** function. That is:

- It controls the loop iterations
- It can only be created in a loop, and
- There can only be one Restructure function within a loop and it must be the first in the loop's sequence

Use this procedure to add a Restructure function in a mapping. Repeat this procedure for nested restructure loops, if necessary.

1 Double-click on a mapping to open it in the Mapper Editor.

2 In the Core Functions pane, select **Restructure**.

Drag and drop it in the space inside a loop where to add a Restructure function.

A control link from the Restructure function to the loop is automatically created.

Note: When connecting input elements, you can drop elements only where it is acceptable.

3 Add input parameters to receive input data.

Click and drag a data element, or an attribute from the input document to the parameter area, to create an input parameter.

A corresponding output parameter is automatically created.

The input links to Restructure functions are dotted to indicate that data is not read when executing the restructure function. Instead the data has already been read.

4 Add input control links to the function header.

Add this to indicate repeating elements in the input document that should control looping when reading the data.

Click and drag a repeating element from the input document to the function header.

A bold dotted input link will be created. You can create several control links to the Restructure function, if needed.

5 Link the output parameters containing the restructured data.

Click and drag an output parameter to a data element or an attribute in the output document or vice versa. You can also assign an output parameter to a global variable and/or click and drag the output parameter to an input parameter for a following function.

6 Define the restructure processing.

Set these properties for the input parameters:

Property	Details
Sorting Sequence	Set 0 for no sorting (default) or a positive integer to sort the data. A given sorting sequence > 0 must be unique within the function.
Sort Order	If sorting sequence **> 0**, set this property to decide the sort order, either Ascending (default) or Descending.
Is Grouped	If sorting sequence **> 0,** you can set this property to: **false** (default) for no grouping, or **true** for grouping of the data You can only set this property to true for the input parameter with the highest sorting sequence. If a sorting sequence is set for two or more input parameters, data will be grouped on the unique combination of these parameter values.
Is Summable	• If another input parameter is grouped and the data for the current input parameter is numerical, you can set this property to true to automatically sum the data. • If another input parameter is grouped and this property is set to false (default), data from the first record in the current group according to the total sorting is used. • If no input parameter is grouped this property has no effect. **Note:** • If you set an invalid value for a property, the value will automatically change back to its previous value. • If you press Tab after setting an invalid property value, an error message will be shown in the status bar.

7 Save the function in the Mapping Editor.

Restructure Concepts

This section contains an in-depth discussions of the details behind the Restructure function. References to the following input document will be used.

The Restructure Function

Normally, a function (Java, Boolean, Database, API, and others) is executed as defined in the sequence. If the function has input parameters that are linked to elements and attributes in the input document, here is what happens:

- First, the corresponding data is read from the input document.
- Then, the function is executed. For example, the manually created Java code.
- Lastly, if the function has output parameters, data created when executing the function is written to elements and attributes in the output document and/or assigned to variables.

These three steps are executed in real time for every function in the sequence: read data, run function, and write data.

The Restructure function is a little bit different. The last two parts are the same as described above; the function is executed and the output parameters get data that is linked to the output document and/or assigned to variables. However, this data is not fetched from the input document, even though the input parameters have to be linked to elements and attributes in the input document. Instead, the data is fetched from memory where it has been restructured. So, when is the data read from the input document?

You can compare the Restructure function to a Boolean function. It can only exist first inside a loop and it has to control the loop. This type of loop is now called a "restructure loop". You can have nested restructure loops, each controlled by a separate Restructure function.

Just before the outmost restructure loop begins to execute, the data is read from the input document. This is an invisible "automatic mapping" that, if visible, would look more or less as the mapping you would manually create to read data from the input document. If the input document contains two nested repeating elements, for example *CreditTransferPayment* and *PaymentTransaction*, the "automatic

mapping" will have two nested loops controlled by these two input elements. Within each loop, data is read from input data elements and attributes. Then, only the references to the data is stored in memory. The input data is never duplicated. This will keep the memory consumption for the restructuring process low.

After this "automatic mapping", the data (or data references) are restructured. Then, the outmost restructure loop is executed controlled by the Restructure function. Restructured data is fetched from memory and assigned to the output parameters. If there are nested restructure loops, these are also executed just as any other nested loop. The nested Restructure functions automatically keep track of the restructure context. For example, only the payment transactions for supplier "*A*" are given if there is an outer restructure loop grouping the suppliers, and supplier "*A*" is the current supplier.

Defining output data to write

The structure for the output document dictates the structure of the loops and functions in the sequence. This is because these functions create the output document. For example, if you want to loop through all suppliers:

- create a loop that controls the repeating element for supplier in the output document
- create a Restructure function in the loop

The Restructure function will automatically control the loop.

If the output document has a nested structure, for example payment transactions per supplier, create the corresponding nested loop controlled by another Restructure function. After doing this, you need to define the input data to use.

Defining input data to read

The "automatic mapping", that is data to read from the input document, is defined by the links from the input document to the Restructure functions' input parameters and function headers. You can only create input parameters for a Restructure function by linking elements, or attributes, in the input document to the Restructure function where input parameters are automatically created, just as for Java and Boolean functions. When doing so, notice that these links are dotted and not solid like the other links. This is to indicate that data reading does not happen at that exact place in the sequence, but earlier. Think of these links as "*I want to use this data*", or references.

When an input parameter is created, an output parameter with the same name is automatically created.

- If you rename the input parameter, the corresponding output parameter is also renamed.
- If you delete the input parameter, the corresponding output parameter is also deleted.
- If you move the output parameter, the corresponding input parameter is also moved.

The restructure process does not alter the data in any way, except when numerical data is summed, so we get this parameter pairing. The input and output parameters always have the data type **`String`**.

For the Restructure function, you must create links by linking repeating elements in the input document to the function's header. Drag and drop (or use the Link tool) a repeating input element to the blue area of the Restructure function, for example to the icon. Since this is a control link, it is bold. The link is also dotted like the data links. When doing the "automatic mapping", you need to know which input elements should control the loops. That is, on which input elements you want to loop (the "loop level").

For the credit transfer example, it would be the repeating elements **`CreditTransferPayment`** and **`PaymentTransaction`**. Think of these links as "*I want to loop on these elements*".

Note:
- Every Restructure function needs to have at least one loop link and one data link (input/output parameter).
- You can link the same repeating input element to several Restructure functions.
- You can also link an input data element to several Restructure functions.

The input loop, or control links, do not need to be nested in the same way as the restructure loop. Otherwise, this would not be restructuring.

Note that, if you have a loop level, that is a repeating element, in the input document that only contain an inner loop (repeating child element), and you do not need any data from that (outer) loop level, you still need to link that repeating element to a Restructure function. Otherwise, the "automatic mapping" will not loop on that level and will only read the first instance.

So, why can you link input elements to Restructure functions in this "anarchical" way? To create the "automatic mapping" that reads data from the input document, you need all XPaths for the repeating elements controlling the loops, and all XPaths for the data elements to read. The order in which the XPaths are defined does not matter since the actual XPaths can be sorted to create correct metadata for the "automatic mapping".

In this way, the input document's data structure is decoupled from the desired (output document) data structure.

Defining data to be sorted

To restructure the data, you need to sort it. There are two input parameter properties for sorting: **Sorting Sequence** and **Sort Order**.

Sorting Sequence

Sorting Sequence is set to 0 as default, which means no sorting. By setting Sorting Sequence to a positive integer value, the data is sorted on that element. You can set Sorting Sequence for several input parameters for a Restructure function.

For example:

- sort on supplier name (*Sorting Sequence* = *1*)
- sort on IBANID (*Sorting Sequence* = *2*).

Note: A Sorting Sequence value > 0, must be unique within a Restructure function.

Sort Order

If Sorting Sequence is > 0, you can set the property **Sort Order** to either Ascending, or Descending sort order.

Note: All data is handled as strings and sorted lexicographically according to the Unicode Collation Algorithm (UCA). For more information on UCA, see **http://unicode.org/reports/tr10/**

For example, "11" is less than "2" (due to the different lengths) for ascending sort order.

The null value for a non-existing input element is less than the lowest value for an existing input element. This means, null is less than an empty value (and any other value) for ascending sort order.

These input parameter properties decide the sorting for the data at the current loop level. The total sorting is constructed from the sorting properties for Restructure functions for all nested restructure loops. See this example:

If data for the outmost restructure loop is sorted on:

- 1 *supplier name*
- 2 *IBAN ID*

And, if data for the inner restructure loop is sorted on:

- *date*

The total sorting will be:

1 supplier name,

2 IBAN ID, and

3 date

Defining data to be grouped

In this example, since you have an outer restructure loop for supplier, you also want to group the data on supplier. Otherwise, this loop is not needed, nor even valid. If the input parameter property Sorting Sequence is > 0, you can set the property **`Is Grouped`** to true. If you set Sorting Sequence for *supplier name* to 1, and **`Is Grouped`** to true, you will get one restructure loop iteration per unique supplier name. You can also group several elements.

To get one iteration per unique combination of *supplier name* and *IBAN ID*, follow this setting:

- **`Sorting Sequence`** to 1, and
- **`Is Grouped`** to false for *supplier name*,
- **`Sorting Sequence`** to 2, and
- **`Is Grouped`** to true for *IBAN ID*

Note: You can only set the property **`Is Grouped`** to true for the input parameter with the highest sorting sequence number.

Defining data to be summed

For example, if you sort payment transaction on *date*, you will get one iteration per payment transaction in date order. If setting the property **`Is Grouped`** to true for the *date* parameter, you will instead get one iteration per unique date. If you also have an *amount* parameter in that Restructure function, as default, you will get the amount for one of the payment transactions per unique date. However, this is probably not what you want.

By setting the property **`Is Summable`** to true for the *amount* parameter, you will get a summed amount for all payment transactions per unique date. In order to make this work, you must ensure that the linked input amount element always exists, that is not null, and contains a valid integer or decimal number. Otherwise, the mapping will crash at runtime. There is no decimal rounding, or other processing of the number, done.

Note: The summing is made using the Java class **`BigDecimal`**.

Limitations

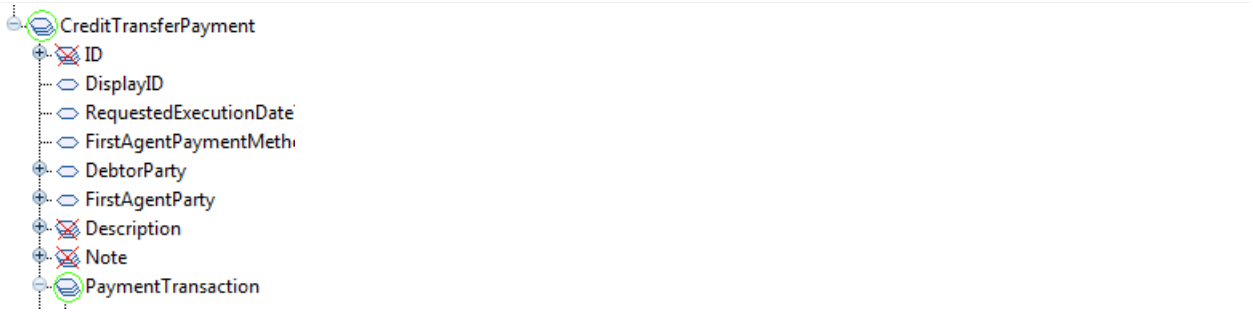
In the "automatic mapping" for Restructure functions, data is read from the input document to memory as it is. It means that, you cannot modify the input data before it is restructured, neither apply logic on it in any other way. After the restructuring, you can modify the data in the mapping as usual.

Some examples:

- It is not possible to substring a value, which may complicate sorting and grouping if all values need to have the same length for correct handling, or if you want to group on only a part of the values.
- It is not possible to use the instance with a specific attribute value for a repeating data element that is not looped in the restructuring process, for example a repeating ID data element. The first instance will always be used.
- It is not possible to sort the value for either element A, or element B, depending on the value of another element. You have to sort element A and element B as separate elements.
- It is not possible to sort the values from two input elements that are included within a choice (that is, only one of the elements can exist at a time) as one value. You have to sort these elements separately.
- It is not possible to convert a negative amount to a positive amount for correct summing.
- It is not possible to filter out data, for example, to skip all transactions with a negative amount.

The only workaround for this is to create another mapping that is executed before the mapping that restructures the data. The purpose of the first mapping is to "fix" the input document so that the data can be restructured as it is, in the second mapping. You could possibly use XSLT instead of the first mapping.

The current version does not support restructuring of parallel repeating elements, or loops, in the input document. It is not possible to restructure, for example, a repeating element for notes plus a repeating element for payment transactions, both per credit transfer payment, even if you just want to move the notes loop to the output document as it is. Support for this will be added in a future version.



All loop levels must be given in the input document instance for the restructure process to work. For example, if the complex element *PaymentTransaction* is not given for a *CreditTransferPayment* instance, the *CreditTransferPayment* instance will not be included in the restructured data.

Troubleshooting

A

Known Issues and Workarounds

Failure to remove a mapping folder

"resources could not be deleted" error message

When removing a mapping folder and prompted with this error message, use the following procedure to resolve the issue:

1. Set the Heap memory.

   Select Windows > Preferences > General.
2. Select **Show Heap Status**.
3. Click Apply, OK.

   The heap memory status indicator and a garbage bin icon is displayed in the lower hand corner of your Eclipse.
4. Click on the garbage bin icon to perform a garbage collection.
5. Retry the delete operation.

If this does not work, restart Eclipse and then retry the delete operation.

Failed to remove

Deleted mappings which no longer display in Mapping Explorer View (MEV) may not have been completely removed. To verify, go to the standard Navigator View, select the same mapping and invoke delete from the Navigator View context menu.

Failure to access a database

ION Mapper will hang when a connection to a database you are trying to access is corrupted.

Use the following Mapping Console commands:

- **`dbadm error`** - use this to see if an error occurred.

  Or
- **`dbadm purge`** - use this to reset the connection.

Schema files are not updated in the repository

For more information, see "Schema Requirements" on page 51.

Eclipse hangs when opening the Mapping Editor

When this happens, restart Eclipse and reopen the Mapping Editor.

Removal of a repository function

A repository function can only be removed from a repository using the Repository function > Delete option in the ION Mapper context menu. This means that, a function must be in a mapping before it can be removed.

Mapping Console is not connected with the ION Mapper Editor

When you reopen Eclipse it will display the last active ION Mapper when it was previously closed. However, the Mapping Console will not be associated with the editor. When this happens, select the ION Mapper to reactivate its connection to the console.

Editing function parameters are not instantly reflected in Java Editor

If you are simultaneously editing a Java function in the Java editor and editing the function parameters in the ION Mapper, these are not instantly reflected in the Java editor. To update the Java editor you must close and then reopen the Java editor.

Missing input or output document elements

Using XML schemas could be very complex; for example, the use of **`OAGIS BOD's`**. ION Mapper may lack the support for complex schema constructions and it could result to missing document elements and, or attributes.

When this happens, use the **`xdstat`** Mapping Console command to see how ION Mapper deals about the schema. Support for missing constructs will be added, as needed.

New type definitions are not instantly available

When you add new type definition for input/output parameters, or variables/constants, those will not be instantly available in the **Type** property options. You must refresh the options for the new types to be present.

1. Close the **Properties** tab.
2. Reselect the **Type** property.

Tracing

The **Mapper Log Console** in ION Mapper is an additional tool only for developers. They use this tool to trace or debug the internal components of a mapping.

"Unable to find source code for mapping id '<id>'."

If you get this error message it means that the mapping used by the XML Transform process in the agreement is published, but is not activated.

Activate the mapping and re-run the message.

Error Messages

B

Broken Links

Missing document error

When ION Mapper cannot find the input/output document referenced by a mapping, it results to broken links. In this case the mapping will still open, however, with an additional error for broken links.

If this happens check the **`SchemaIn/Out.File`** and **`SchemaIn/Out.Name`** properties of the mapping and verify that they refer to a schema file present in the mapping folder.

If there are difference in the file, use the Eclipse Refactor option to rename it to the expected name and try the mapping once again. For more information, see "Renaming Mappings" on page 41.

Broken link error

When the ION Mapper cannot find the source and/or target for one or more links in a mapping, it results to broken links.

The broken links are listed in a table and the details are displayed in the lower pane on the Broken Links tab of the Mapping Console view. Check the original internal ID for the target and source elements, and a statement about which of them is missing. If you double-click on the link, the remaining element will be selected in the editor.

Note: The broken links will continue to persist in the mapping until these are removed. Mappings cannot be saved to a Mapper database until all broken links are resolved.

For a more detailed information on broken or missing links, use the **`linkstat`** command in the **Console Commands and Output** tab of the Mapping Console view.

This command output details and the number of broken links categorized as:

- **Scalar link**- broken links between variables/constants, function parameters, and input /output document elements.
- **Non-scalar link** - broken links between loops, function parameters and input /output document elements.
- **Control link** - broken links controlling loops which are missing a condition. This is an abnormal case and does not normally happen.

The link details are shown following the format:

<SRC_NAME or "Missing!"><SRC_ID> -- <LINK_TYPE> <SRC_NAME or "Missing!"> <TGT_ID>

Where:

- **SRC_NAME** - is the name of the source mapping element.
- **TGT_NAME** - is the name of the target mapping element.

  Note: For missing mapping element, the NAME will be "Missing!"

- **SRC_ID** - is the Id of the source mapping element.
- **LINK_TYPE** - encodes the link type.
- **TGT_ID** - is the ID of the target mapping element.

Here is an example:

```
Element link: foreach_orderregels [ LID2 ] -- LM --> Missing! [ MOD3 ]
Element link: oId [ OID81 ] -- PM --> Missing! [ MOD4 ]
Element link: oFilePath [ OID90 ] -- PM --> Missing! [ MOD5 ]
Element link: oSpecification [ OID92 ] -- PM --> Missing! [ MOD8 ]
Element link: oId [ OID91 ] -- PM --> Missing! [ MOD9 ]
Element link: oDescription [ OID93 ] -- PM --> Missing! [ MOD10 ]
Element link: oPercentage [ OID94 ] -- PM --> Missing! [ MOD11 ]
Element link: oLanguage_Description [ OID83 ] -- PM --> Missing! [ MOD12 ]
```

Where:

Link type	Description
ML	Input Document Element > Loop
LM	Loop > Output Document Element
FL	Function > Loop
PL	Output Parameter > Loop
MP	Document Element > Input Parameter
PM	Output Parameter > Document Element
VP	Variable > Input Parameter
PV	Output Parameter > Variable
KP	Constant > Input Parameter
PP	Output Parameter > Input Parameter
master_control	Control Link
straight	Data Link

Link type	Description
variable	Link to/from variable
constant	Link from constant

Note: When a mapping is saved all broken links are discarded. If you want to keep the **`linkstat`** information to analyze later, you can copy and store the details.

Input/output missing root element error

When a mapping uses an input/output root document element that can no longer be found in the input/output schema it results to input/output missing root element error.

This can happen if you edit or change a schema, or when importing a mapping with inconsistent metadata. In the case of the latter, you can check the **`mapping.input/output.element`** from the mapping.properties file and verify that the values are correct. Or, you can manually edit the values and try again.

XML Schema Basics

C

XML Schema Basics for Wildcard Elements

The xs:any element enables the author to extend the XML document with elements not specified by the schema. Parent elements for the wildcard element can be **`xs:sequence`** or **`xs:choice`**.

For Infor BODs, the wildcard element is used in the complex type UserAreaType, which is used for all UserArea elements. The complex type UserAreaType is defined in the generic schema **`"..\..\Resources\Components\Common\Fields.xsd"`**.

The attributes "**`namespace`**" and "**`processContents`**" are important for which elements that can be used in the instance document.

Attribute	Description
namespace	Optional. Specifies the namespaces containing the elements that can be used. Can be set to one of the following: • ##any - elements from any namespace is allowed (this is default) • ##other - elements from any namespace that is not the namespace of the parent element can be present • ##local - elements must come from no namespace • ##targetNamespace - elements from the namespace of the parent element can be present • List of {URI references of namespaces, ##targetNamespace, ##local} - elements from a space-delimited list of the namespaces can be present

Attribute	Description
ProcessContents	Optional. Specifies how the XML processor should handle validation against the elements specified by this any element. Can be set to one of the following: • strict - the XML processor must obtain the schema for the required namespaces and validate the elements (this is default) • lax - same as strict but; if the schema cannot be obtained, no errors will occur • skip - The XML processor does not attempt to validate any elements from the specified namespaces

ION Mapper and mapper runtime will only support namespace="##any" and processContents="strict". This means that any element defined in any namespace can be used as replacement element. However the replacement element must be declared in the schema for the default namespace, or in any other custom namespace with a custom schema that is referenced from the instance document.

In order to validate the instance document it must have a reference to its schema. The attribute xmlns, for example xmlns="http://schema.infor.com/InforOAGIS/2", specifies the default namespace declaration. This declaration tells the schema-validator that all non-prefixed elements used in this XML document are declared in the namespace **"http://schema.infor.com/InforOAGIS/2"**.

Once you have the XML Schema Instance namespace available (**xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"**) you can use the attribute **xsi:schemaLocation**. This attribute has two values, separated by a space. The first value is the namespace to use. The second value is the location of the XML schema to use for that namespace, for example:

```
xsi:schemaLocation="http://schema.infor.com/InforOAGIS/2
http://schema.infor.com/2.6.4/InforOAGIS/BODs/Developer/SyncProductionOrder.xsd"
```

Now the schema-validator can validate all elements that belong to the default namespace, that is non-prefixed elements, as well as elements that belong to other namespaces defined in the schema. If you want to use a replacement element that is not declared in the referenced schema, you have to declare the replacement element in a custom namespace. And, give a reference to the custom schema for that namespace. You can use several custom namespaces in one instance document.

- First, you need to specify the custom namespace declaration. You connect the custom namespace with a prefix, for example **xmlns:cust="http://schema.infor.com/InforOAGIS/Custom"**, where the prefix "**cust**" is used for all elements that are declared in the custom namespace **"http://schema.infor.com/InforOAGIS/Custom"**.
- Then, you have to add the location of the custom schema for the custom namespace in the attribute **"xsi:schemaLocation"**:

```
 xsi:schemaLocation="http://schema.infor.com/InforOAGIS/2
http://schema.infor.com/2.6.4/InforOAGIS/BODs/Developer/SyncProductionOrder.xsd
http://schema.infor.com/InforOAGIS/Custom CustomHeader.xsd"
```

Now the schema-validator can find schemas for both the default namespace and for the custom namespace(s), provided that the URLs for the schema files are correct.

You can have several custom namespaces in an Instance document. You can also have several custom schemas for one custom namespace, using replacement elements declared in the same custom namespace but from different custom schemas. The custom namespaces and custom schemas are listed as pairs in the **xsi:schemaLocation** attribute value.

This is an example of a root tag for an Infor BOD where you have one custom namespace and two custom schemas:

```
<SyncProductionOrder xmlns="http://schema.infor.com/InforOAGIS/2"
xmlns:cust="http://schema.infor.com/InforOAGIS/Custom"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://schema.infor.com/InforOAGIS/2
http://schema.infor.com/2.6.4/InforOAGIS/BODs/Developer/SyncProductionOrder.xsd
http://schema.infor.com/InforOAGIS/Custom
custom/UserAreaExtensions/Ferrari/CustomHeader.xsd
http://schema.infor.com/InforOAGIS/Custom
custom/UserAreaExtensions/Ferrari/CustomDetail.xsd" releaseID="9.2" versionID="2.6.4">
```

This is the corresponding CustomHeader schema:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns="http://schema.infor.com/InforOAGIS/Custom"
targetNamespace="http://schema.infor.com/InforOAGIS/Custom" elementFormDefault="qualified"
attributeFormDefault="unqualified">
 <xs:element name="AdditionalInformation">
  <xs:complexType>
   <xs:sequence>
    <xs:element name="ReceivingWarehouse" type="xs:string"/>
    <xs:element name="SoldToBPName" minOccurs="0" maxOccurs="unbounded">
     <xs:complexType>
      <xs:simpleContent>
       <xs:extension base="xs:string">
        <xs:attribute name="languageID"/>
       </xs:extension>
      </xs:simpleContent>
     </xs:complexType>
    </xs:element>
    <xs:element name="ShipToBPName" minOccurs="0" maxOccurs="unbounded">
     <xs:complexType>
      <xs:simpleContent>
       <xs:extension base="xs:string">
        <xs:attribute name="languageID"/>
       </xs:extension>
      </xs:simpleContent>
     </xs:complexType>
    </xs:element>
    <xs:element name="InterfaceUpdate" type="xs:string"/>
   </xs:sequence>
  </xs:complexType>
 </xs:element>
</xs:schema>
```

Note: The custom namespace must be declared as target namespace in all custom schemas.

In the following BOD instance example the replacement element **"Property"** is declared in the default namespace, while the replacement element **"AdditionalInformation"** is declared in the custom namespace **"http://schema.infor.com/InforOAGIS/Custom"**, and uses the prefix **"cust"**. The replacement element **"Property"** is defined in the standard BOD schema and the replacement element **"cust:AdditionalInformation"** is defined in the CustomHeader schema.

```
<UserArea>
 <Property>
  <NameValue name="ln.Priority" type="NumericType">999</NameValue>
 </Property>
```

```
 <Property>
  <NameValue name="ln.Owner" type="StringType">SFC</NameValue>
 </Property>
 <Property>
  <NameValue name="ln.Routing" type="StringType">001</NameValue>
 </Property>
 <cust:AdditionalInformation>
  <cust:ReceivingWarehouse>WHAMS1</cust:ReceivingWarehouse>
  <cust:SoldToBPName/>
  <cust:SoldToBPName languageID=""/>
  <cust:ShipToBPName/>
  <cust:ShipToBPName languageID=""/>
  <cust:InterfaceUpdate>No</cust:InterfaceUpdate>
 </cust:AdditionalInformation>
</UserArea>
```

So, instead of defining the structure of an XML document only in one schema for the default namespace, the instance document can also use additional custom namespaces and provide references to custom schemas.

Definition of Terms

D

Mapper terminologies

- ION Mapper - The Mapping Manager client tool. Also known as "Mapping Manager"
- MEC - M3 Enterprise Collaborator (MEC). The product containing the MEC Server and MEC Client Tools. ION Mapper is one of the MEC client tools.
- Mapper Server - The server where you generate and publish mappings. This is part of ION Runtime Services and MEC.
- Mapper Database - The database repository for mappings.

  For ION this is one of the repositories. For MEC it is the MEC database.
- UVJ function - User defined Void Java function
- UBJ function - User defined Boolean Java function
- REP function - Repository function

Wild Card Elements terminologies

- Wildcard element - the wildcard schema component, that is the XML schema element "any".
- Replacement element - an element that will be used in instance documents as wildcard replacement contents, that is where the schema specifies a wildcard element.
- Custom namespace - a namespace that is not defined in the main schema. The default namespace is defined in the main schema.
- Custom prefix - a prefix for a custom namespace
- Custom schema - a schema where a custom namespace is defined. It is referenced from the instance document, not from the schema for the default namespace.