XLATE Evolution User Guide

A Comprehensive Guide to XLATE Evolution

Document Information

Document Date	13 July 2012
Product Version	1.6.0

Table of Contents

1 Introducing XLATE Evolution

1.1 Introduction

The purpose of this guide is to give you a full understanding of what XLATE Evolution (Xe) can do, and how to use it.

XLATE is Data Interchange PLC's established mapping engine for converting between EDI and in-house document types. XLATE Evolution incorporates an updated release of the XLATE engine, but is built around a new and more flexible mapping engine providing translation between any document types (EDI, in-house, IDOCs, XML) and includes a comprehensive set of tools for developing, testing and deploying maps and reports.

This release combines and extends the functionality of XLATE/Windows and EDI Studio in a single integrated Windows application, incorporating an updated Data Dictionary, which has been renamed as the Standard Reference Library and is implemented in a Microsoft Access database. A new component, the File Investigator, has been included to allow you to examine the content of EDI files in a user-friendly way. Finally, another new component, the Xe Mapper, has been included to provide a more powerful and flexible mapping tool.

1.1.1 Xlate Evolution overview

Xlate Evolution is a powerful EDI development tool, designed for the easy creation and use of file mapping tables, and the manipulation and examination of EDI files. The application comprises several main tools: Xlate, the Standard Reference Library, the File Investigator, the Definition Editor, the Report Designer, the Mapper and the Index Explorer.

1.1.1.1 Xlate

The Xlate component of Xe provides for the translation of commercial data in and out of standard format. It is for use by trading partners within an EDI (Electronic Data Interchange) environment.

The classic Xlate mapping functions are made available from a new GUI. The new environment includes editing facilities for Xlate tables that allow combined development and testing of mapping configurations.

There have been some minor functionality improvements to Xlate, notably the addition of pre- and post-processing commands which may be invoked by Xlate during construction or translation.

The Xlate/Windows GUI functionality has been added to the Xe GUI framework. Xlate functions (translate, construct, validate, reformat and options) can be selected either via a tree control or via the Xlate menu option.

The following functionality and features are provided by the Xlate component:

- Supports EDIFACT, UNGTDI and ANSI X12 syntaxes
- Written in 'C' for stability, portability, speed and efficiency
- Table driven
- Automatic syntax detection
- Ease of table definition – high level language
- Automatic message formatting by destination

- Default starter table definitions
- Table maintenance service
- Efficient compression of non-significant characters
- Table validation
- Flexible – can be used with a variety of communications packages
- Comprehensive reporting capability
- Parameter option
- Highly flexible structure for in-house file
- Table print / file print and reformat
- Comprehensive audit trail
- Full screen browse facility
- Integration with ODEX (Data Interchange Plc's communications product)
- Can be used stand alone
- Full support of EDIFACT qualified segments
- Full support of EDIFACT qualified data elements
- Single data element to multiple in-house record fields
- Support of mixed documents in a single transmission
- Codes and table support
- API / Batch Interface

1.1.1.2 Standard Reference Library

The EDI Data Dictionary is now known as the Standard Reference Library and has been extended to include all of the most recent EDIFACT standards releases, as well as a range of additional ANSI X12 dictionaries.

The Standard Reference Library offers automatic generation of Xlate tables (EDF and HDF definition files) , and an augmented reporting capability. The GUI also provides facilities for creating or editing an existing index file (IDX) required to configure a map.

1.1.1.3 File Investigator

The File Investigator is a new feature that allows the user to examine the contents of EDI files in a variety of user-friendly ways. EDI data can be displayed in raw or hierarchical views, highlighting errors in syntax or structure.

1.1.1.4 Definition Editor

Definition files are used to specify the structure and format of business documents to Xe. The definition editor is a new component that provides a simple, visual interface for the rapid development of Xe definition files.

1.1.1.5 Report Designer

The report designer is a new component for developing printed report and transport label layouts in a drag-and-drop environment. Complex report layouts

can be specified quickly and easily using the report layout viewer, built-in report element toolbox and property editor.

1.1.1.6 Mapper

More powerful and flexible than Xlate, the Mapper is another new feature which allows you to map data from an input file to one or more output files with the aid of a GUI. You simply have to provide a definition for the input file and the output file, then perform the mapping using a combination of drag-and-drop techniques and some simple programming. The GUI enables you to see exactly how each field in an input file maps to each element in a target file. The mapper can also be used in conjunction with the report designer to develop maps that will populate reports you create in it with data from any supported document type.

1.1.1.7 Index Explorer

The index explorer represents the core of the new version of Xe. It is used to display and manage the document definitions, maps and other configuration information required to use the Xe mapping engine. It includes tools for importing and exporting indexes and wizards for performing common tasks.

1.1.1.8 Platforms supported

Xe requires the Microsoft .Net Framework v2.0. Therefore the following platforms are supported:

- Windows 2000 Professional, Server & Advanced Server
- Windows 2003 Server
- Windows XP Home Edition & Professional
- Windows Vista (all versions)

On all these systems, Microsoft Internet Explorer 5.01 or later and Microsoft MDAC 2.6 or later are also required.

1.1.2 Xlate Evolution Help

Online help

Online help is available throughout the Xe product, simply by clicking **F1** or the **Help** button on the page for which you require help.

The on-line Help toolbar looks like the example shown below.

Each of the arrows icons will turn blue as your cursor passes over them.

Previous and **Next** allow you to navigate sequentially through the entire Help document via the headings shown in the Contents tree view in the left-hand Help panel.

Back and **Forward** allow you to navigate through the headings that you have already visited.

The **Print** option allows you either to print the selected topic or to print the selected heading and all subtopics.

Context-sensitive help

Context-sensitive help is available within the Mapper application. Simply click on an item or on the Mapping area to see the context-sensitive help appear on the Help page tab in the bottom right-hand corner of the Mapper page.

2 Installing the software

2.1 Introduction

This chapter provides a step-by-step description of how to install the Xe software. First, you need to ensure that you meet the requirements for installing and running the program.

2.1.1 Requirements

To install and run Xe you will require one of the following operating systems:

- Windows 2000 Professional, Server & Advanced Server
- Windows 2003 Server
- Windows XP Home Edition & Professional
- Windows Vista (any version)

In addition to the Windows operating system, you will also need:

- Windows Installer v2
- Internet Explorer 5.05
- MDAC 2.6 or above
- .Net framework v2.0

If you do not have these installed, they will be installed for you automatically when you install Xe.

To run Xe you will need a PC with at least 300Mb of free disk space.

2.2 Installation

2.2.1 Introduction

First of all, please read the License Agreement included in the package containing the distribution media. It is very important that you understand and agree to the terms and conditions relating to the software before opening the CD and installing it.

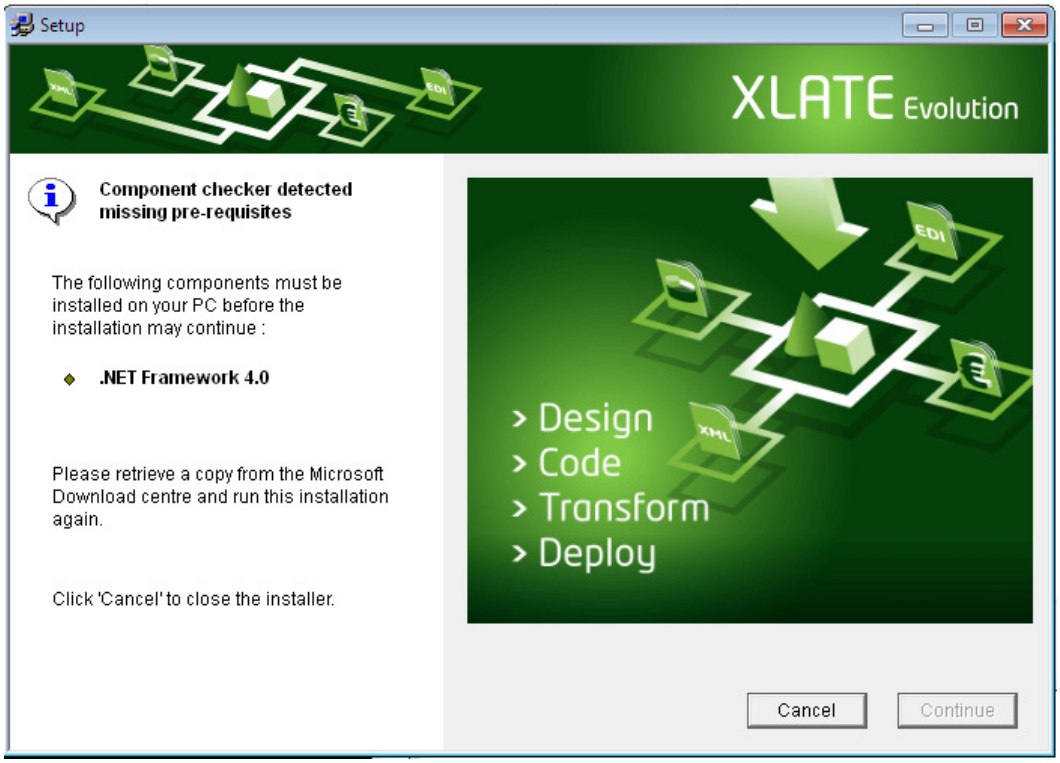
2.2.2 Installation and Setup

Before installing the Xe software, please ensure that you close all other applications.

Now insert the Xe CD into the CD-ROM drive of your computer. The installation program should begin automatically after a few moments but, if it does not, select **Start >> Run** from your Windows toolbar, and type in D:\setup.exe (where D is your CD-ROM drive) then click the **OK** button. This will begin the installation process.

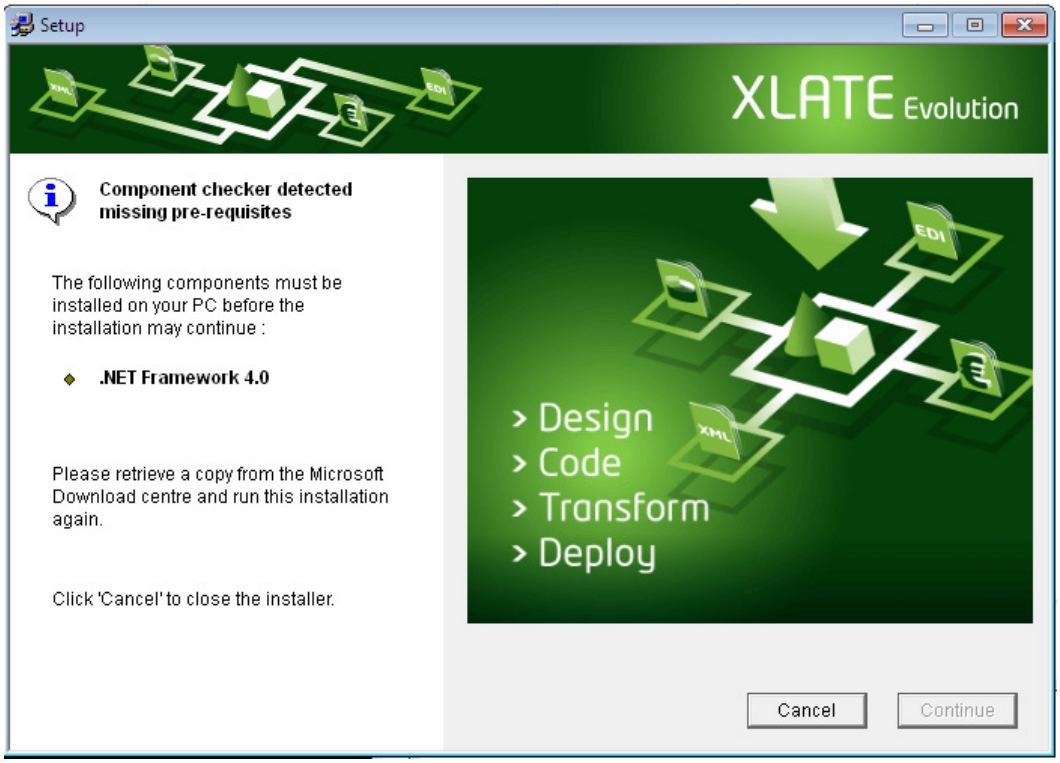
2.2.2.1 Required components installation

When you begin the installation, the component checker will check to see whether any required components need to be installed first. If any required components are missing, you will see the following screen. Otherwise, please continue reading from the section entitled “Xe installation”.



A list of missing pre-requisites will be displayed on the left-hand panel of the screen. Click **Continue** to install these components and proceed to the Xe setup. Click **Cancel** if you want to quit the installation process.

If you choose to continue, you will see the following screen.



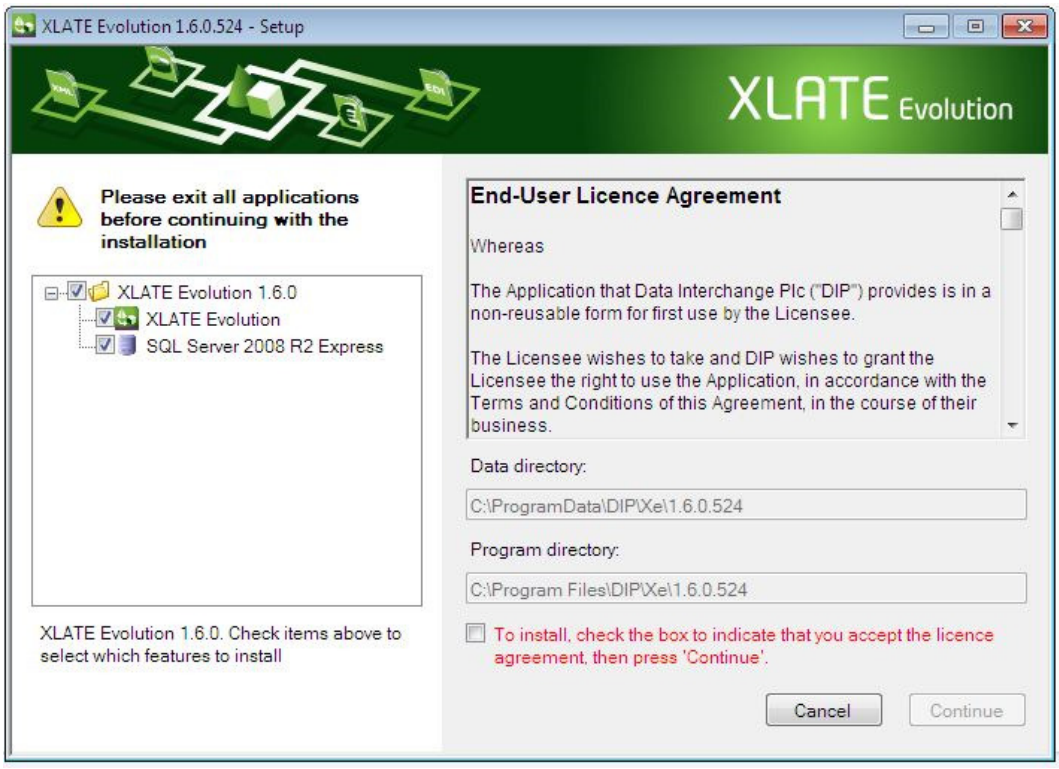
This screen shows you which components are being installed. The installation of each component may take several minutes.

As the installation of each component is completed, the component will be ticked and installation of the next component will begin.

Once all the pre-requisite components have been installed, the installation program will guide you through the process of installing Xe on your computer.

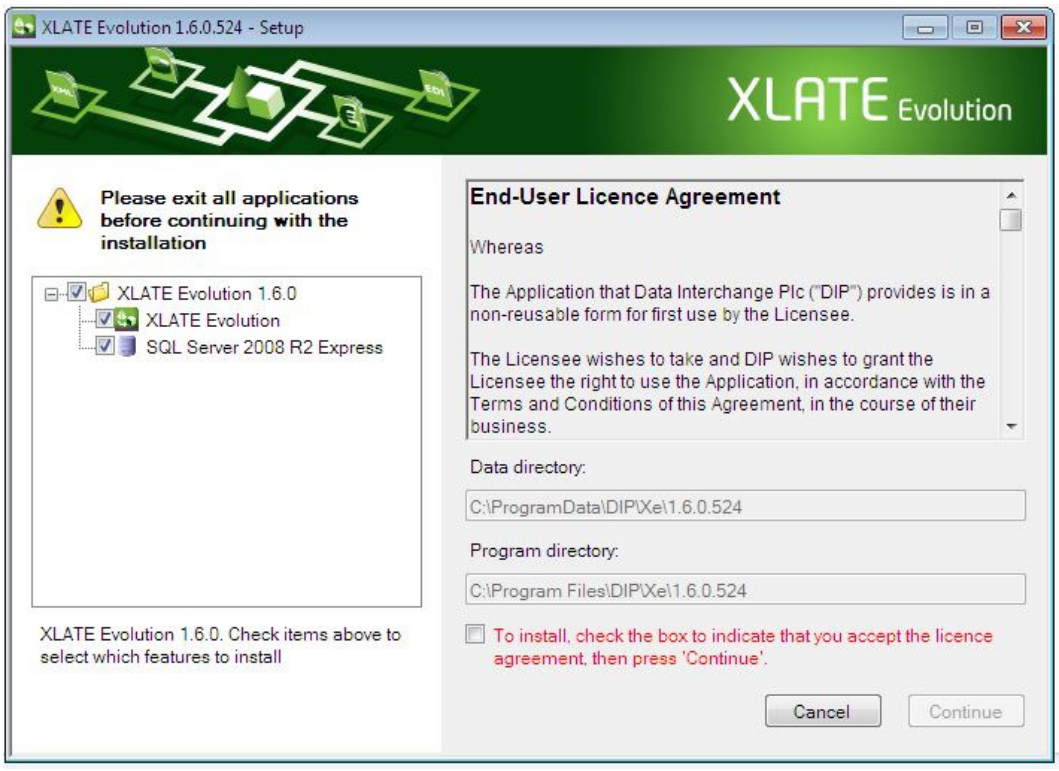
2.2.2.2 Xe installation

You will now see the following dialog, which contains the License Agreement. This is an exact copy of the License Agreement contained in the distribution media package. You should already have read this carefully to ensure that you understand and accept the terms and conditions of the software.



If you have read and agree to the terms and conditions, check the box below the Licence Agreement text. This will enable the **Continue** button. Click **Continue** to continue with the installation. If you click **Cancel**, the installation will close.

Once installation has completed, you will see that all four components have been installed.

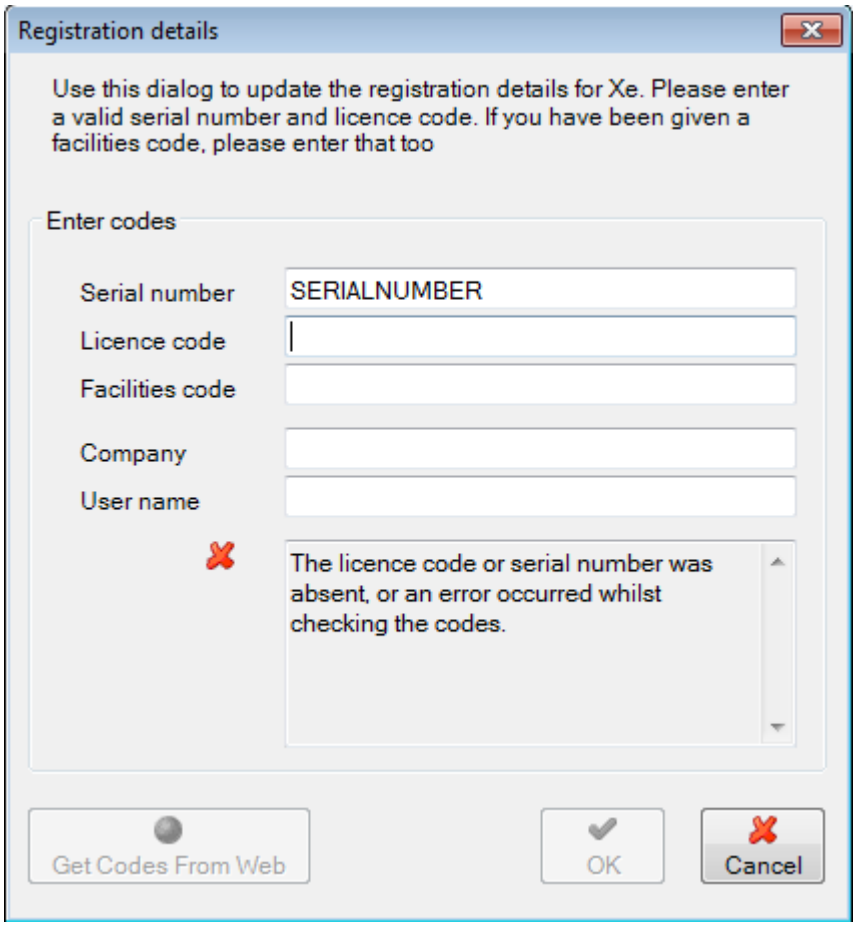


Click the Finish button to close the installer.

You will find that a new shortcut icon has appeared on your desktop. Xe is now ready to use.

2.2.3 Licence Code

The first time you try to use Xe you will see a dialog which requires you to provide your product serial number, licence code and, if applicable, facilities code.



You should have received the relevant information together with the software.

Simply type in the serial number, licence code and, if applicable, facilities code into the spaces provided.

If the details you provide are valid, you will be able to click **OK** to save the registration details and continue with Xe.

3 Using the software

3.1 Configuration

No pre-configuration is required for Xe. The only configuration required for Xlate is in the Xlate section of the main Options dialog, where memory allocations and file locations can be specified. However, these have been configured for you, with optimum work area memory allocations and default master index and table files. You will only need to increase memory allocations in exceptional circumstances. You may override the default index and table files when performing translations and constructions, by specifying your own locations in the Translate and Construct dialogs.

3.2 The GUI

The Xe GUI has been kept as simple and user-friendly as possible. All features and functions are available from the main menu, the toolbar and context menus.

The toolbar stays the same for each component, but the menu items available and context menus change depending on which component or window you are currently working with.

3.2.1 Main menu

The main menu items are:

- File
- Edit
- View
- Dictionary
- Investigator
- Mapper
- Definition
- Xlate
- Help

Each of the main menu items has its own subset of menu items. However, the full subset will only be available if you are currently using a component or window to which it is applicable. The Dictionary, Investigator, Mapper and Xlate menu options are described in the appropriate component section. All other menu options are described below.

3.2.1.1 File

Close – This option closes the window you currently have open in the right-hand panel.

Save – This option is only available if you are working with an existing file in the right-hand panel. It will save any changes you have made to the file, in its original location.

Save As – This option is only available if you are working with a file in the right-hand panel. It requires you to specify the full path and filename to save the file to.

Print – This option is only available if you are working with a file in the right-hand panel. It will print the file to your default printer.

Print Preview – This option is only available if you are working with a file in the right-hand panel. It allows you to preview the file before printing it.

Print Setup – This option is only available if you are working with a file in the right-hand panel. It allows you to specify print layout settings before printing the file.

Select Master Index – Displays a dialog that allows you to select the master index to be displayed in the index explorer. Also includes options for creating and deleting indexes. For more information see “Selecting An Index”.

Options – Displays an options dialog that allows you to set preferences for Xe. For more information see “Options”.

Exit – This option closes the Xe application.

3.2.1.2 Edit

The Edit subset options will only be available if you are working with a file in the right-hand panel.

Undo – This option undoes the last change you made to the file.

Cut – This option will only be available if you have highlighted part of the file. It allows you to remove the highlighted section and, optionally, paste it elsewhere.

Copy – This option will only be available if you have highlighted part of the file. It allows you to copy the highlighted section for use elsewhere.

Paste – This option will only be available if you have something on your clipboard ready to paste. Position your cursor where you want to paste the text and select this option.

Find – This option allows you to search for a specified character or string of characters within the file you are working with.

Replace – This option allows you to replace a specified character or string of characters with another specified character or string of characters within the file you are working with.

3.2.1.3 View

Xlate Index File – This option allows you to view the Xlate index file. It opens the file in a new window in the right-hand panel. If you are using the Xlate default Master Index file, the window will be entitled Xlatepc.idx. Otherwise it will bear the name of the index file you have specified in the Xlate Options dialog.

Xlate Log File – This option allows you to view the Xlate log file. It opens the file in a new window in the right-hand panel, entitled "XlatePc.log".

Clear Xlate Log – This option clears the contents of the Xlate log file.

Xlate Log Trace On – This option indicates whether tracing is turned on in Xlate and allows you to toggle tracing on and off.

File Investigator Log – This option opens the file investigator log, if there is one. The file investigator log is recreated automatically each time the investigator is run.

Xe Mapper Log – This option is available only if the Xe Mapper is visible. It opens the log associated with the project being edited in the mapper. If there is no log associated with the project then a message box is shown telling you this.

Xe Map Script – This option is available only if the Xe Mapper is visible. It opens the script associated with the project being edited in the mapper.

Zoom in – This option is only available if the view in the right-hand panel can be resized. Its effect is to increase the size of the text or image in the file.

Zoom out – This option is only available if the view in the right-hand panel can be resized. Its effect is to reduce the size of the text or image in the file.

3.2.1.4 Help

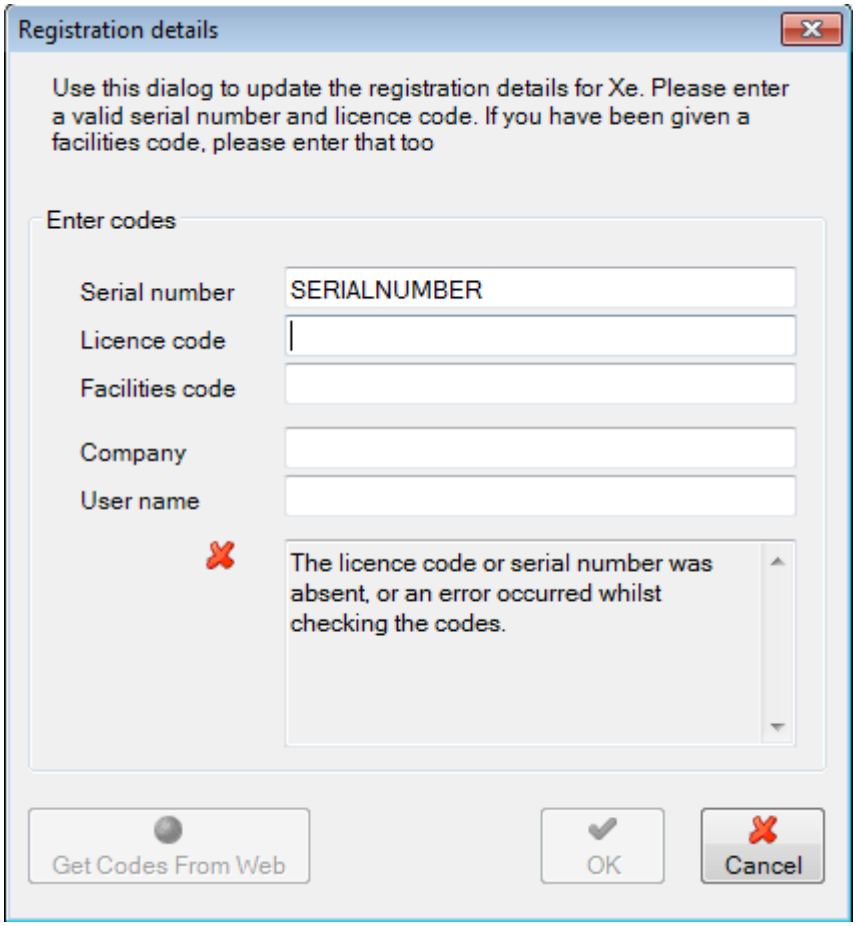
Contents – This option brings up the Contents page of the Xe on-line Help file.

Index – This option brings up the Index page of the Xe on-line Help file.

About Xe – This option brings up a dialog containing the serial number and build of the Xe application. You will need this information if you need to contact our Support department.

Licence Agreement – This option brings up a dialog containing a copy of the Xe Licence Agreement.

Registration Details – This option brings up a dialog containing your serial number and licence code details (and facilities code if you have one), as shown below.



You will need to open this dialog when you receive your new licence code each year, in order to update your details and continue to use the software. If you do not receive your new licence code by email, you can use the Get Codes From Web button on this dialog to obtain the new code.

3.2.2 Toolbar

The toolbar is shown below.

The first five icons on the toolbar are used to open one of the Xe components. You can open as many different components at a time as you wish. You can only open one instance of each component, however.

 This is the Standard Reference Library icon. Click this icon to show the Standard Reference Library tree view in the left-hand pane and the Standard Library window in the right-hand pane. The Standard Library window cannot be closed.

 This is the Xlate icon. Click this icon to show the Xlate tree view in the left-hand pane and open the Xlate Translate window in the right-hand pane.

 This is the File Investigator icon. If you click this icon, you will be asked to provide an existing file to be investigated. Having selected one, you will see a tree view representation of the file in the left-hand pane and the contents of the file in the right-hand pane.

 This is the Hex Viewer icon. If you click this icon, you will be asked to provide an existing file to be displayed in hex format. The Hex Viewer requires no tree view, so the right-hand pane will expand to fill the whole screen. The actual display format of the file can be changed by selecting different options on the screen.

 This is the Mapper icon. This option allows you to choose a map project to open for editing. A list of map projects configured in Xe will be displayed from which you pick one to load into the mapper component.

 This is the Zoom In icon. Click this icon to enlarge the text in the right-hand pane. This option can be used whenever the icon is enabled.

 This is the Zoom Out icon. Click this icon to reduce the size of the text in the right-hand pane. This option can be used whenever the icon is enabled.

 This is the Toggle Left Pane icon. Click this icon to show or hide the left-hand pane.

 This is the View icon. Click on this icon for the option to view the Xlate index file, Xlate log file, File Investigator log file or Xe Mapper log file.

 This is the Mapper Display icon. Click the icon to toggle the mapper display mode (which map connections are displayed).

 This is the Exit icon. Click this icon to quit the Xe application.

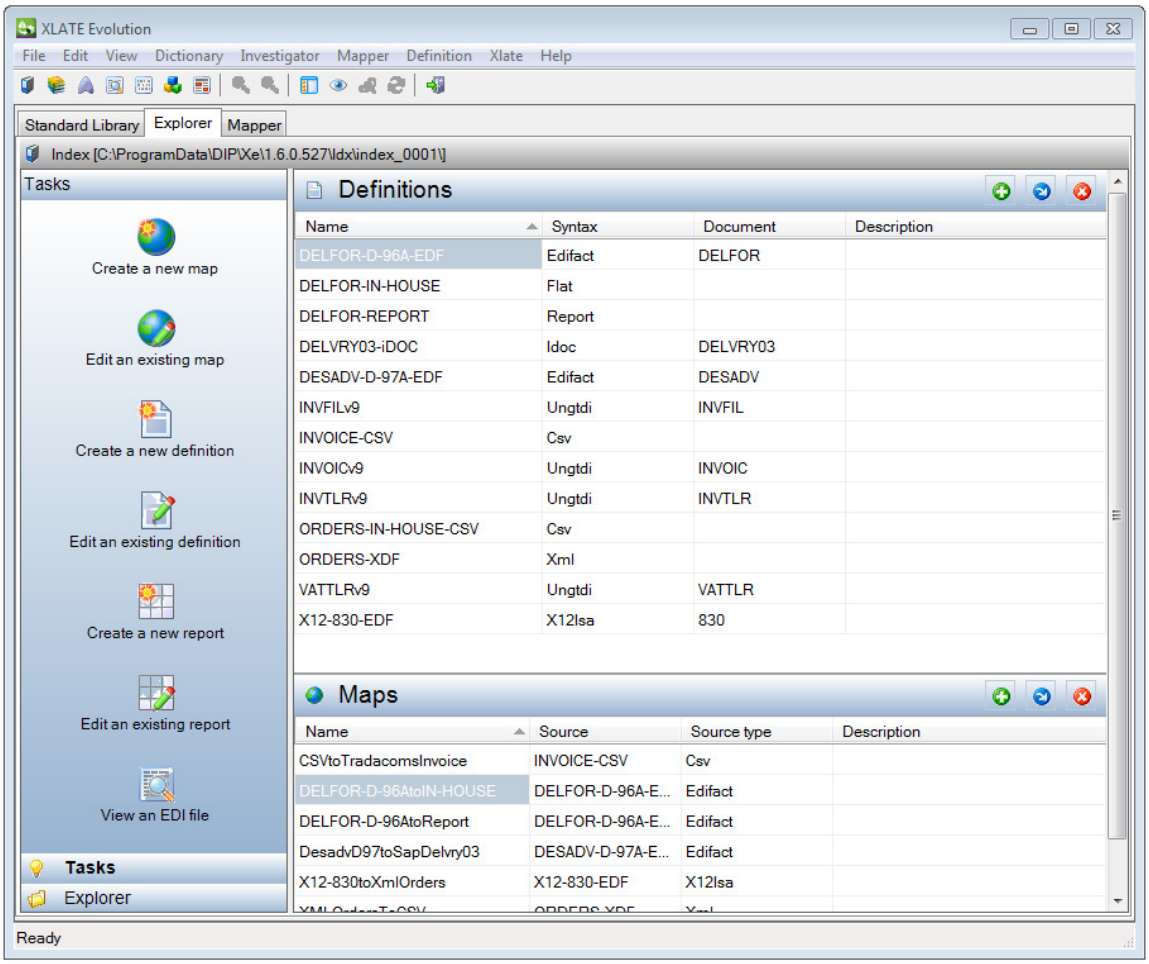
Let's have a look now at the different components and how to use them.

3.3 Index Explorer

The Index Explorer is at the core of the newest versions of Xe. It is used to display and manage the document definitions, maps and other configuration information required to use the Xe mapping engine. It includes tools for importing and exporting indexes and wizards for performing common tasks.

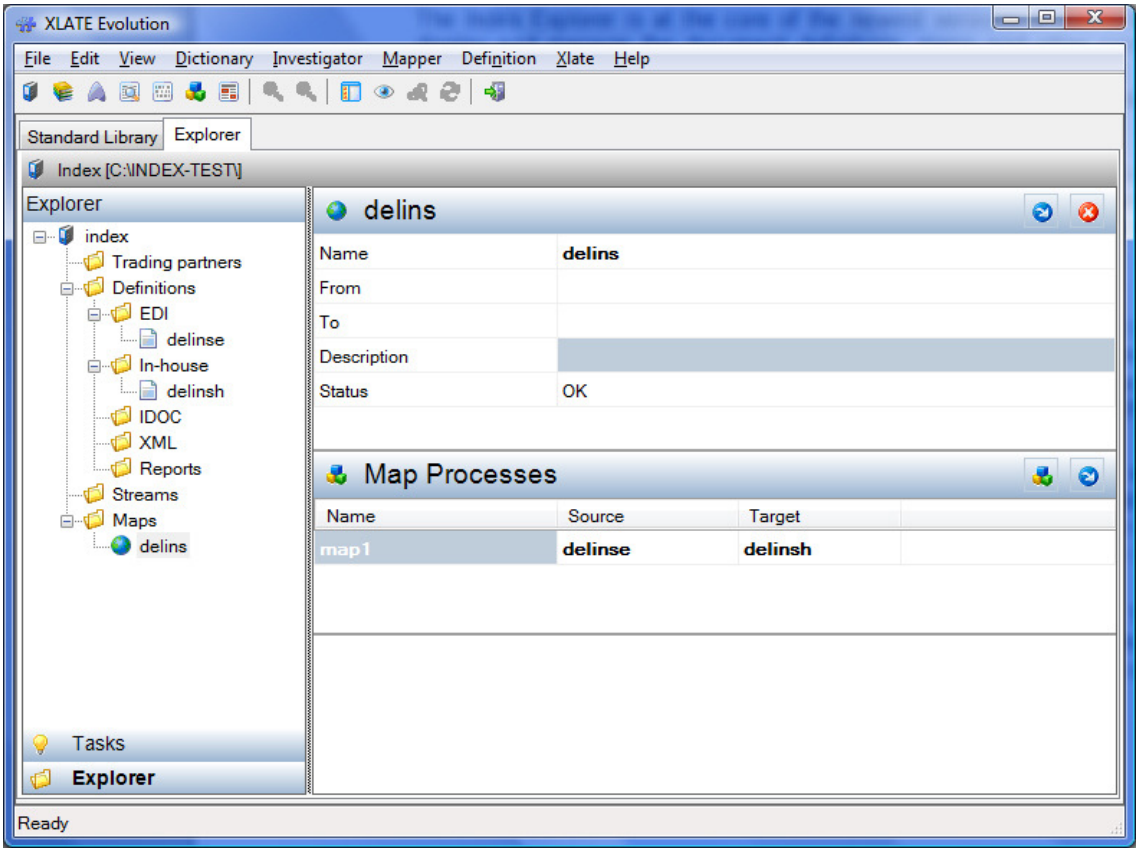
The Xe index is an XML file that contains the configuration information that the mapping engine needs to run. In the past, indexes were created by hand, or on-the-fly by the mapper. The index explorer component now allows index information to be viewed and edited directly using a simple interface.

The Index Explorer has two views. The first, which is the default shown when Xe is first run, is called the Tasks view. The Tasks view provides access to wizards that help you perform common functions, such as creating a definition or editing a map and is shown below:



When the tasks view is shown the left hand part of the view consists of buttons that provide access to the wizards. The right hand part of the view shows two lists, of the definitions and maps configured in the currently displayed index.

The alternative view is the Explorer itself, which is shown below:



In this view, the left hand side is a tree representing the entities defined in the current index, divided up by type. The right hand side is populated by one or more lists according to what is selected in the tree. Use the tree to navigate maps, definitions and other entities configured in the index and the list views to edit them.

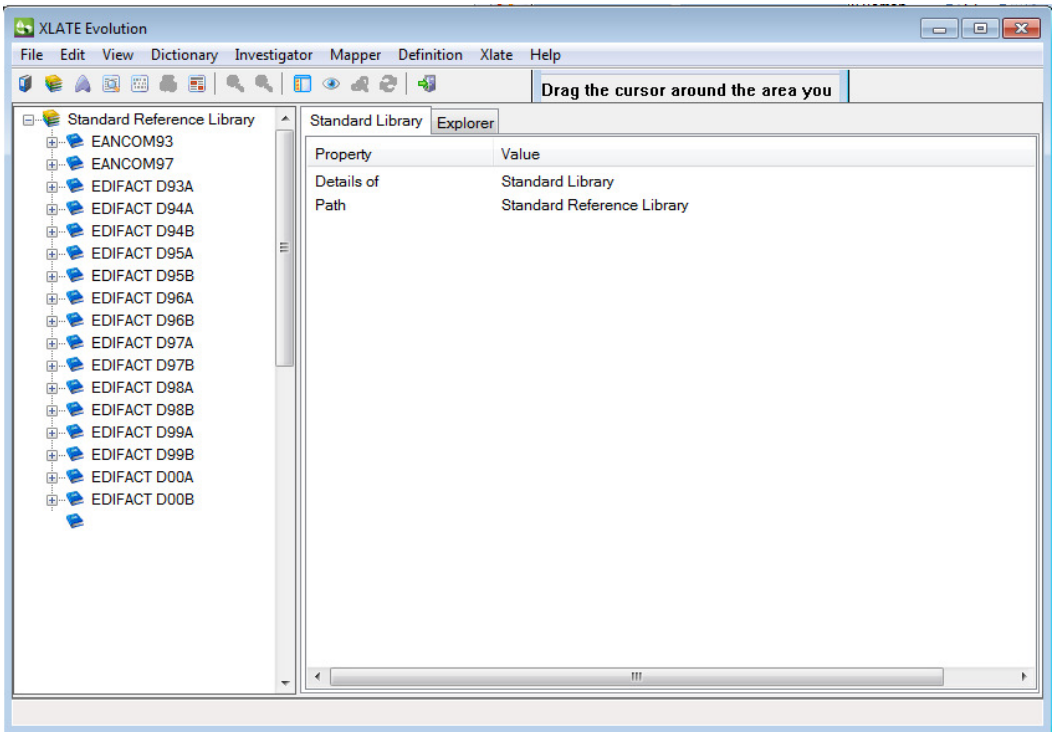
You can switch between the Tasks and Explorer views using the selection bars at the bottom left of the view. The text on the bar relating to the currently selected view is shown in bold. Click the other bar to switch views.

Full details on how to use the Explorer can be found in the section entitled “The Index Explorer”.

3.4 Standard Reference Library

The Standard Reference Library can be used both as a reference tool, to view the contents of different messages and standards in a variety of ways, and as a creative tool, to generate definition files for EDI messages and in-house files.

When you open Xe for the first time, or when you click the Standard Reference Library icon  , you will see the following screen.



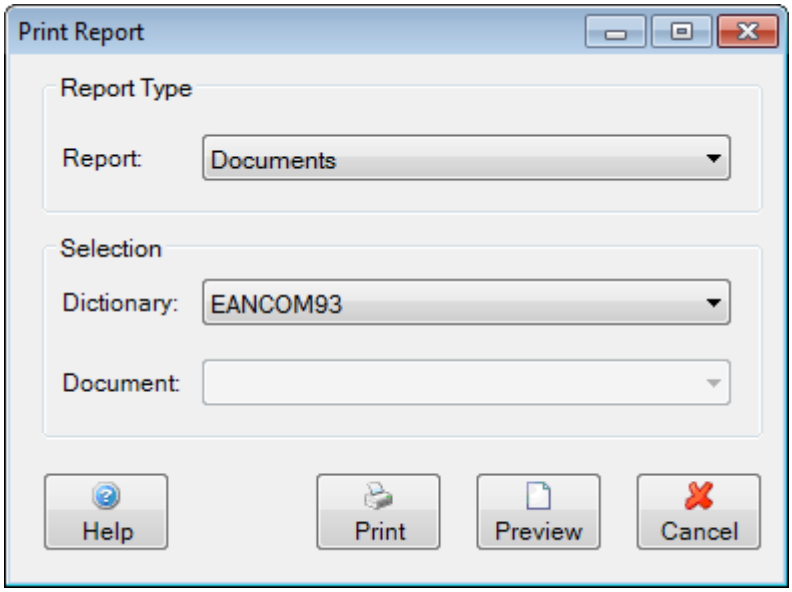
At first, the right-hand panel will be completely empty, until you select something from the tree view. If you click on any entity in the tree view, a description of that entity will appear in the right-hand panel, in the Standard Library window. The Standard Library window cannot be closed.

The library contains a large selection of EDIFACT, Tradacoms, VDA, ANSI X12 and Eancom dictionaries.

Each dictionary appears in the left-hand pane as a node in the tree view. By clicking on the plus sign against each node you can expand each dictionary to view the messages, segments, composites, elements and codelists of which it is comprised. There are then a variety of actions that can be performed against each dictionary or message, as described below.

3.4.1 Reports

There are a wide variety of reports that can be created from the Library. You can right-click on any node in the tree view and select the Print Reports option to see the following dialog:



Please note that the term "document" is interchangeable with the term "EDI message".

Report type – Report

In the Report field, use the dropdown arrow to select the information you want to see on the report. You have the choice of Documents, Document structure (summary), Document structure (detail), Segments, Composites, Elements and Codelists from which to create a report.

Selection – Dictionary

Use the dropdown arrow to select the dictionary to be used for the contents of the report.

Selection – Document

This field is only enabled if you have chosen either of the Document structure options in the Report field above. All the other options produce a report based on all documents.

Use the dropdown arrow to choose the Document you want to base the report on.

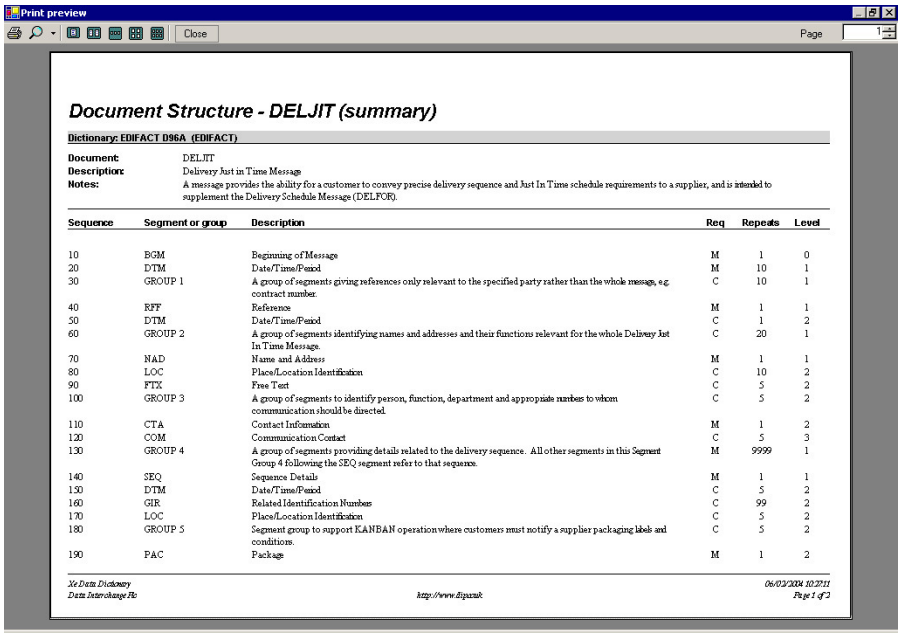
Print

Click this button to print the report immediately. Please note that you can only print the whole report, not part of it.

Preview

Click this button to see a preview of the report. You can then choose whether or not to print the report on paper. It is always advisable to use this option to see how many pages the report fills before deciding to print it. You will be able to print the report from the preview screen but, as with the Print option, you can only print the whole report, not part of it.

A sample report is shown below. This is a Document Structure (summary) of the DELJIT document in the EDIFACT D96A dictionary.


Print preview

Close

Page 1

Document Structure - DELJIT (summary)

Dictionary: EDIFACT D96A (EDIFACT)

Document: DELJIT
Description: Delivery Just in Time Message
Notes: A message provides the ability for a customer to convey precise delivery sequence and Just In Time schedule requirements to a supplier, and is intended to supplement the Delivery Schedule Message (DELFOR).

Sequence	Segment or group	Description	Req	Repeats	Level
10	BGM	Beginning of Message	M	1	0
20	DTM	Date/Time/Period	M	10	1
30	GROUP 1	A group of segments giving references only relevant to the specified party rather than the whole message, e.g. contract number.	C	10	1
40	RFF	Reference	M	1	1
50	DTM	Date/Time/Period	C	1	2
60	GROUP 2	A group of segments identifying names and addresses and their functions relevant for the whole Delivery Just In Time Message.	C	20	1
70	NAD	Name and Address	M	1	1
80	LOC	Place/Location Identification	C	10	2
90	FTX	Free Text	C	5	2
100	GROUP 3	A group of segments to identify person, function, department and appropriate numbers to whom communication should be directed.	C	5	2
110	CTA	Contact Information	M	1	2
120	COM	Communication Contact	C	5	3
130	GROUP 4	A group of segments providing details related to the delivery sequence. All other segments in this Segment Group 4 following the SEQ segment refer to that sequence.	M	9999	1
140	SEQ	Sequence Details	M	1	1
150	DTM	Date/Time/Period	C	5	2
160	GIR	Related Identification Numbers	C	99	2
170	LOC	Place/Location Identification	C	5	2
180	GROUP 5	Segment group to support KANBAN operation where customers must notify a supplier packaging labels and conditions.	C	5	2
190	PAC	Package	M	1	2

Xe Data Dictionary *06/03/2004 10:27:11*
Data Interchange Re *http://www.ibpuk.* *Page 1 of 2*

It shows, in tabular format, all the structural information of the document, together with descriptions of the document, segment groups and segments. This particular sample report covers two pages.

A Document Structure (detail) report would show each segment on a new page, listing each composite and element of each segment. This report, based on the DELJIT D96A document, would cover 40 pages.

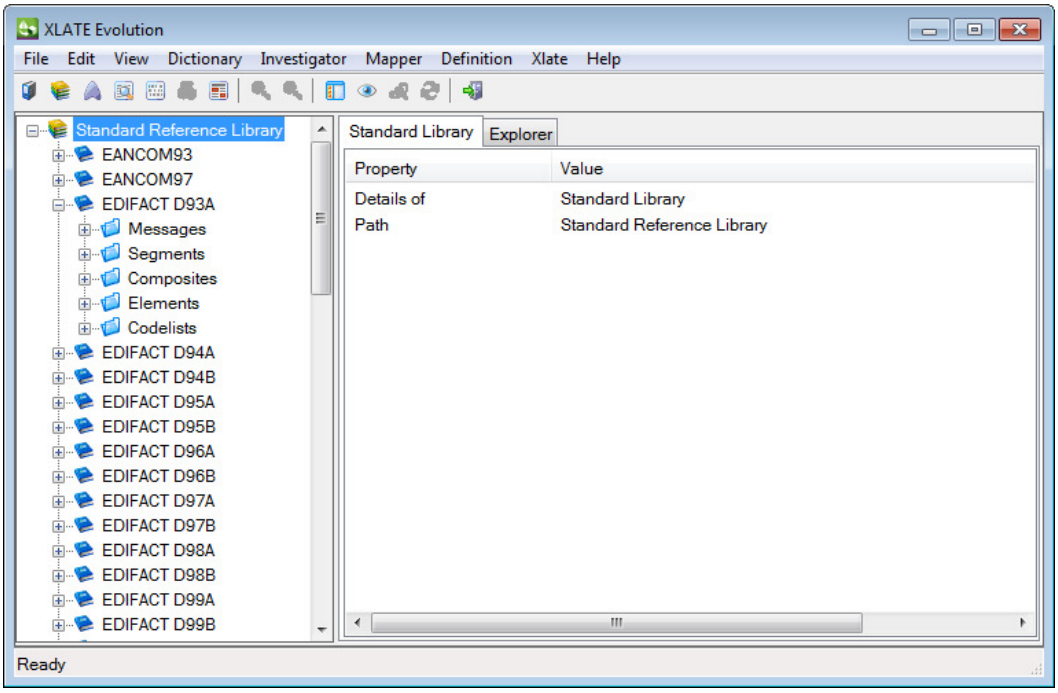
Cancel

Click this button to leave the Print Report dialog without creating a report.

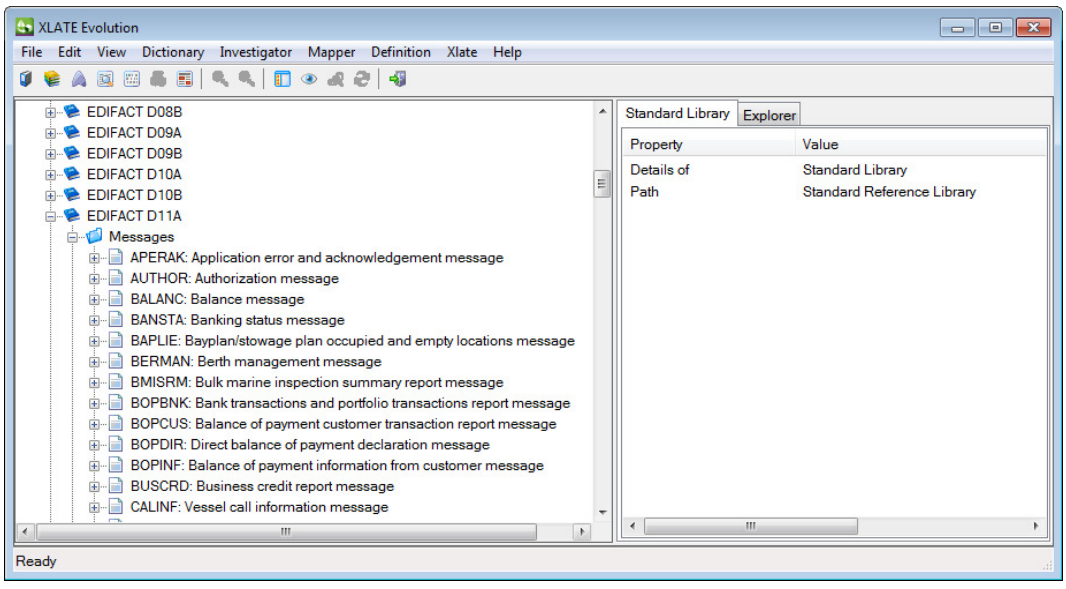
3.4.2 Generate Xlate tables

You can generate Xlate tables for any document in the Library, in the following way.

Choose the appropriate dictionary from the tree view and expand it by clicking on the plus sign to its left, as shown below.



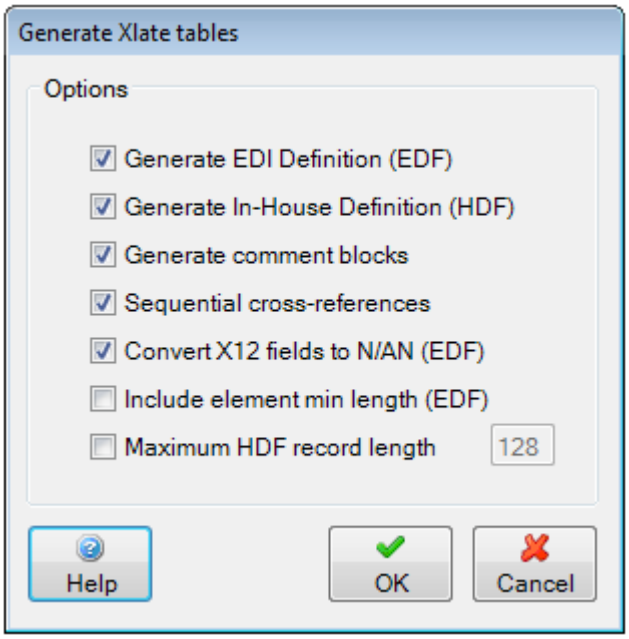
Now expand the Messages node by clicking on the plus sign to its left, as shown below.



This shows you a list of all the EDI messages that are included in the dictionary you have selected.

To generate Xlate tables for a message, right-click on the message for which you want to generate the tables. This will bring up a context menu, from which you should select the Generate Xlate Tables option.

You will now see the Generate Xlate Tables dialog, shown below.



By default, the first four options are selected, but you may select whichever options are suitable to your purposes.

Select "Generate EDI Definition" to generate an EDF file.

Select "Generate In-House Definition" to generate an HDF file.

Select "Generate comment blocks" to create banner comments throughout the generated definition files.

"Sequential cross-references" refers to the Xlate cross-references that are assigned to map each segment data element in the EDF file to a field in the HDF file. All generated cross-references will be unique, whether you select this option or not. If you select this option, the numbers used in the cross-references will start at 10 and continue sequentially throughout the file. If you do not select this option, the numbers used in the cross-references will start at 10 for each new segment.

Select “Convert X12 fields to N/AN (EDF)” to convert ID, DT and TM fields in the data dictionary to A or AN fields in the EDF.

Select “Include element min length (EDF)” to include a minimum field lengths in the EDF if present in the data dictionary definition.

Select "Maximum HDF record length" if you need to restrict the length of HDF records. If this option is selected, the value field on the right will become enabled, allowing you to specify the maximum length of each HDF record. Remember that you must take account of the Record Type length in the length of the HDF record.

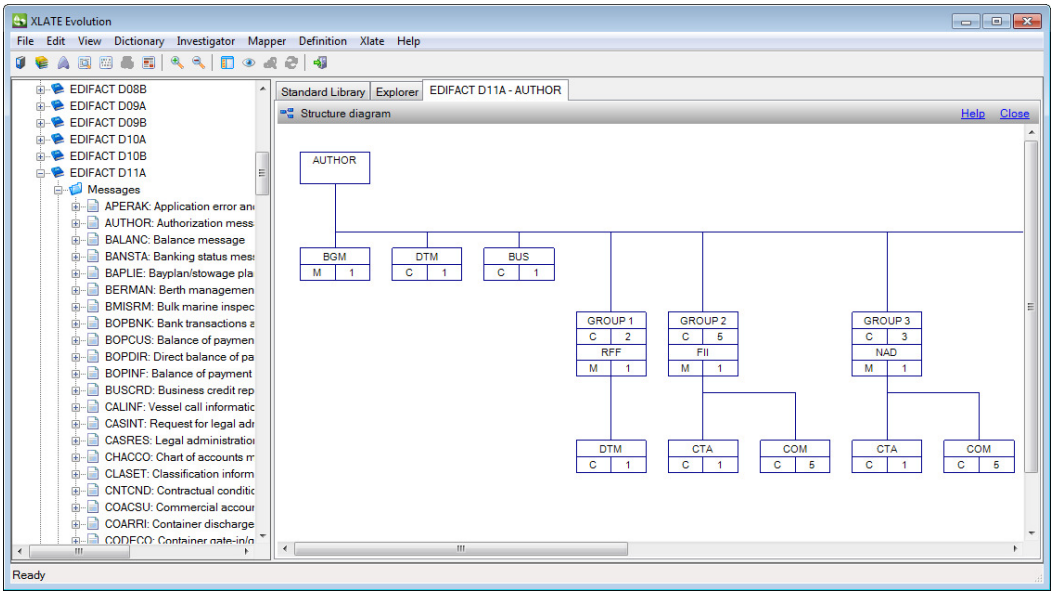
Click **OK** to generate the definition file(s).

Click **Cancel** to leave the Generate Xlate Tables dialog without generating any files.

3.4.3 Display Structure Diagram

You can generate a structure diagram for any document in the Library, in the following way. Choose the appropriate dictionary from the tree view and expand it by clicking on the plus sign to its left. Now expand the Messages node by clicking on the plus sign to its left. To display a structure diagram for a message, right-click on the message for which you want to see the structure. This will bring up a context menu, from which you should select the Display Structure Diagram option.

This will display the structure diagram in the right-hand panel, as shown in the example below.



Use the scroll bars to view those parts of the diagram that are not visible.

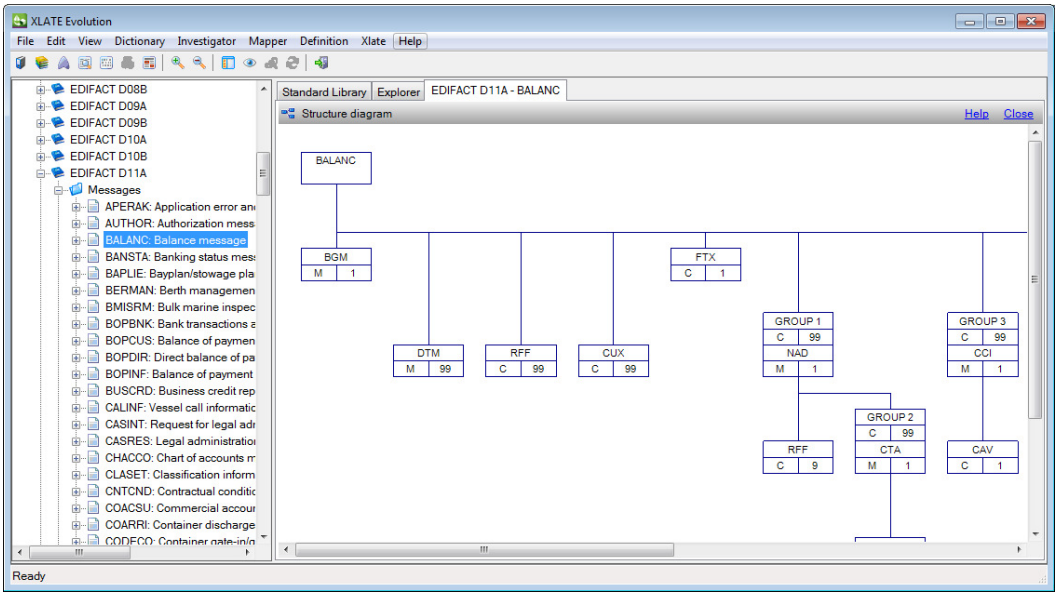
Use the **Close Window** button on the right-hand pane to close the window.

3.4.3.1 Print structure diagram

Any structure diagram can be printed by using the **File >> Print** option from the main menu.

3.4.3.2 Locate in Dictionary

This is a useful feature, allowing you to see exactly where any segment or segment group in the structure diagram occurs in the EDI message. Right-click on a segment (or group) in the structure diagram, then click on the Locate in Dictionary option that pops up. The selected segment or group will immediately be highlighted in the tree view, automatically expanding the tree if necessary. An example is shown below.



3.4.4 Import Dictionary

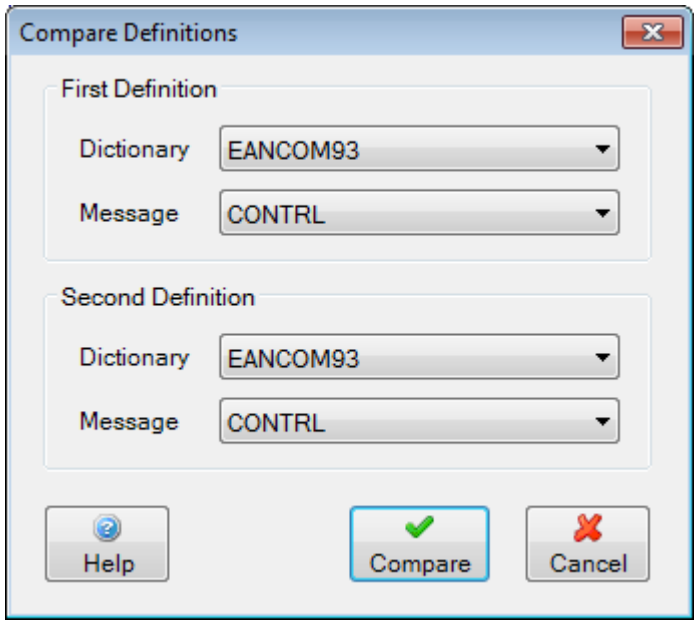
Further dictionaries may be imported into the Standard Reference Library if you have the appropriate .ddd files.

Select **Dictionary >> Import Dictionary** from the main menu, or right-click anywhere within the tree view and select Import Dictionary from the context menu that appears.

3.4.5 Compare Messages

Another feature of the Standard Reference Library is the ability to compare two messages, to see if and how they differ.

Select **Dictionary >> Compare Messages** from the main menu, which will bring up the following dialog.

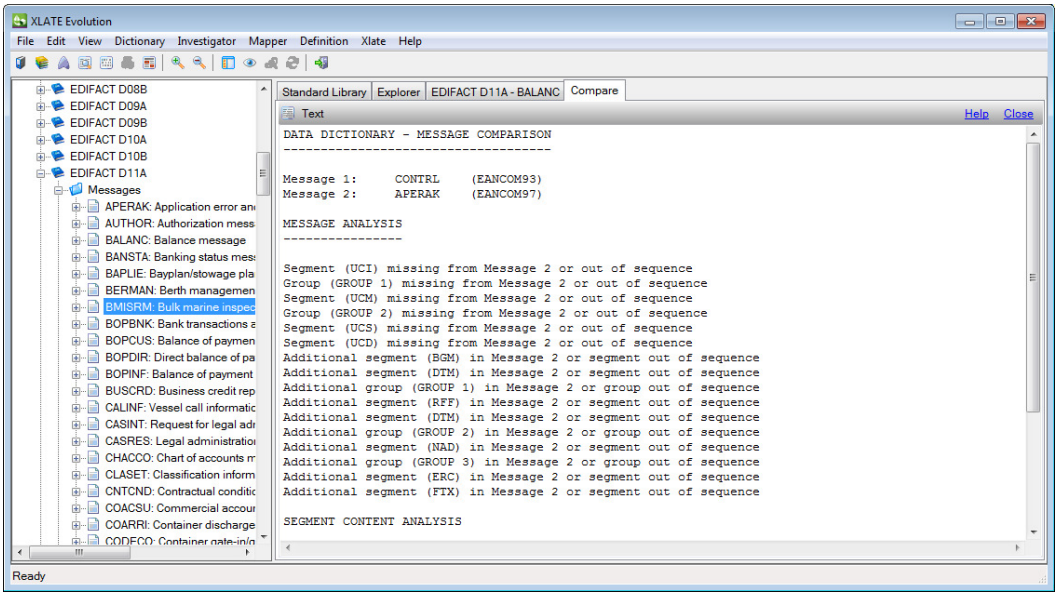


Select a dictionary and message in the First Definition section, to compare to another selected dictionary and message in the Second Definition section.

This option is most useful for comparing two versions of the same message, so that you can see how much they differ, if at all.

Click the Compare button to produce a report, opened as a new window in the right-hand panel, showing Message Analysis, Segment Content Analysis,

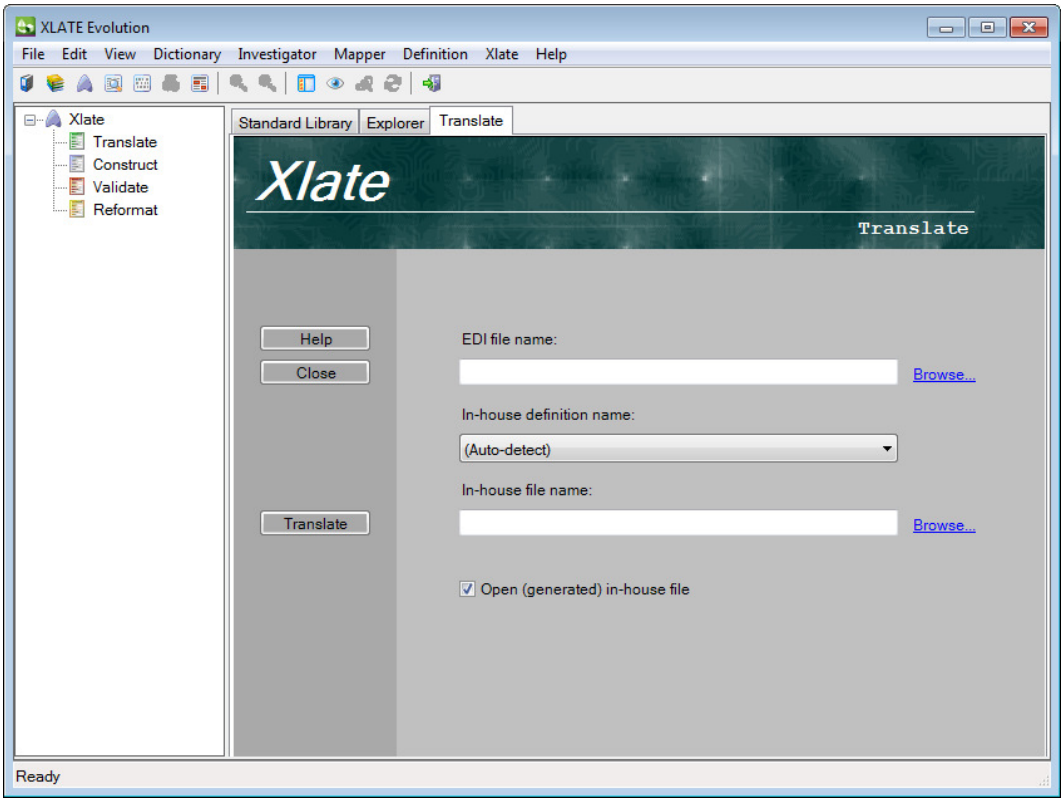
Composite Content Analysis and Element Format Analysis. An example is shown below.



3.5 Xlate

The Xlate component allows you to use existing mapping tables to produce EDI files and in-house files and to validate or reformat EDI files.

When you click the Xlate icon , you will see the following screen.



This gives you access to all the different functions of Xlate (Translate, Construct, Reformat, Validate).

Alternatively, you can use the Xlate menu option to select the function you require, or use the following shortcuts:

Ctrl+F5 for Translate

Ctrl+F6 for Construct

Ctrl+F7 for Reformat

Ctrl+F8 for Validate

In the right-hand pane you will see the window for the function you have selected, with the Log area below it.

In the left-hand pane you can see the Xlate tree view, which allows you to open windows for the other functions of the Xlate component.

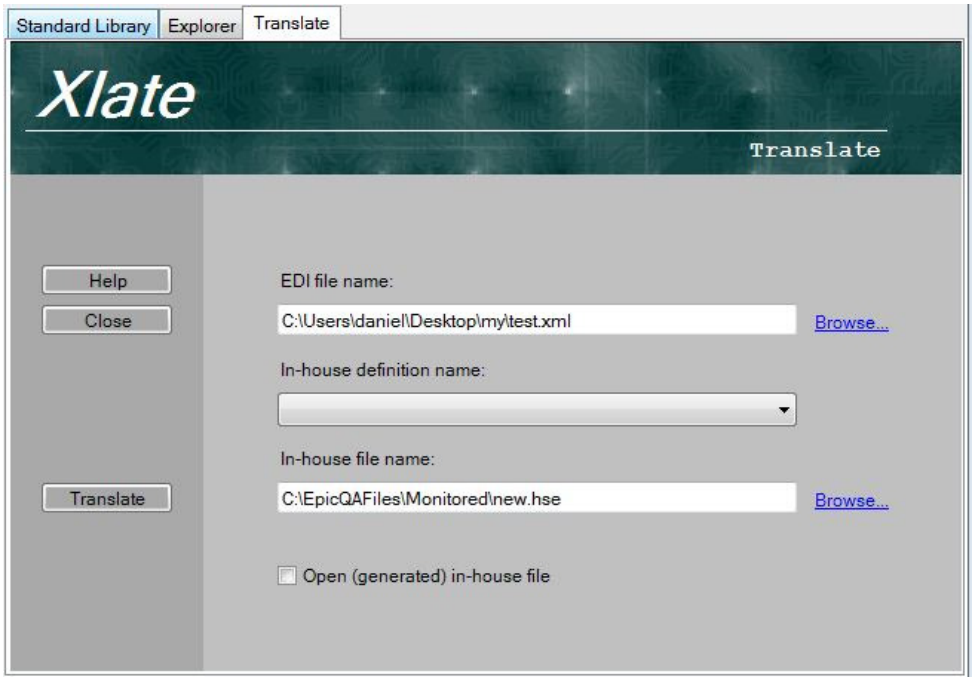
You can open as many Xlate functions at once as you require, but no more than one of each. A separate window will be opened for each in the right-hand panel.

Use the **Close Window** button on the right-hand pane to close the window.

The Xlate functions are described below.

3.5.1 Translate

The Xlate Translate window is shown below. The Translate window allows you to translate an EDI file into an in-house file format.



Before using this window you should ensure that the appropriate entries have been made in the Xlate Index file. For details of the Index file and its requirements, please refer to the Xlate on-line Help.

The fields and buttons in this window are:

EDI file name – if your EDI file is located in the Table Files Directory as provided in the Xlate Options dialog, you can simply type in the name of the EDI file. Otherwise, use the Browse option to provide the full directory and file path for the EDI file.

In-house definition name – use the dropdown arrow to select the appropriate in-house definition name. If there are no multiple entries for any EDI messages in the Index file, you can use the Auto-detect option instead, and Xlate will determine which in-house definition name should be used.

In-house file name – if you want the output in-house file to be located in the Table Files Directory as provided in the Xlate Options dialog, you can simply type in the name of the in-house file. Otherwise, use the Browse option to provide the full directory and file path for the in-house file.

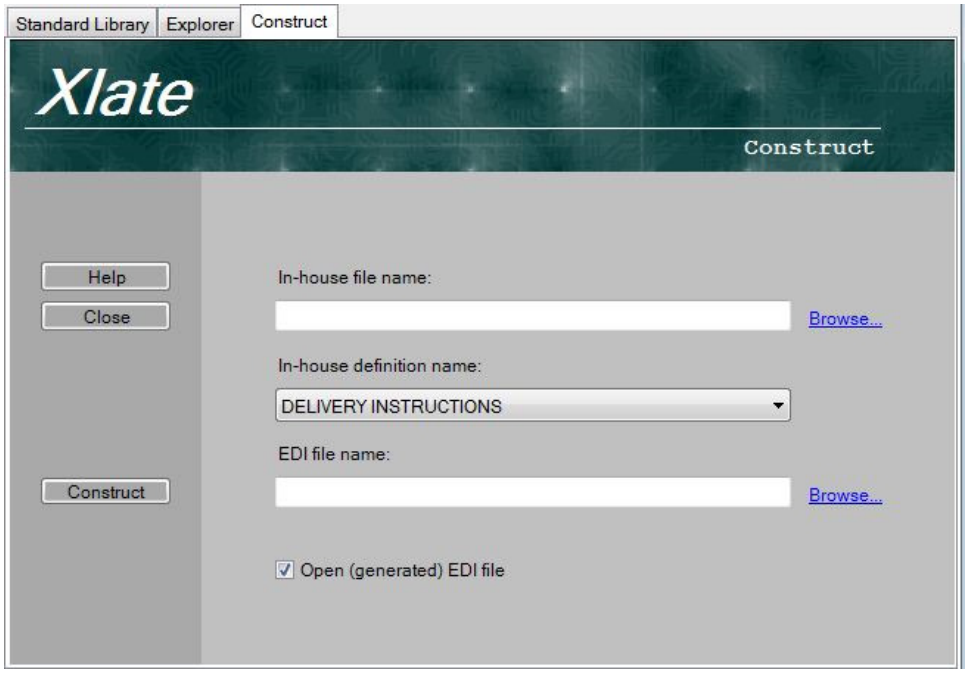
Open (generated) in-house file – if you select this tickbox, Xe will open a new window in the right-hand pane, containing the successfully generated in-house file.

Translate – click this button to initiate the translation of the EDI file into the in-house file.

Close – click this button to close the Translate window.

3.5.2 Construct

The Xlate Construct window is shown below. The Construct window allows you to construct an in-house file into an EDI file.



Before using this window you should ensure that the appropriate entries have been made in the Xlate Index file. For details of the Index file and its requirements, please refer to the Xlate on-line Help.

The fields and buttons in this window are:

In-house file name – if your in-house file is located in the Table Files Directory as provided in the Xlate Options dialog, you can simply type in the name of the in-house file. Otherwise, use the Browse option to provide the full directory and file path for the in-house file.

In-house definition name – use the dropdown arrow to select the appropriate in-house definition name. If there are no multiple entries for any EDI messages in the Index file, you can use the Auto-detect option instead, and Xlate will determine which in-house definition name should be used.

EDI file name – if you want the output EDI file to be located in the Table Files Directory as provided in the Xlate Options dialog, you can simply type in the name of the EDI file. Otherwise, use the Browse option to provide the full directory and file path for the EDI file.

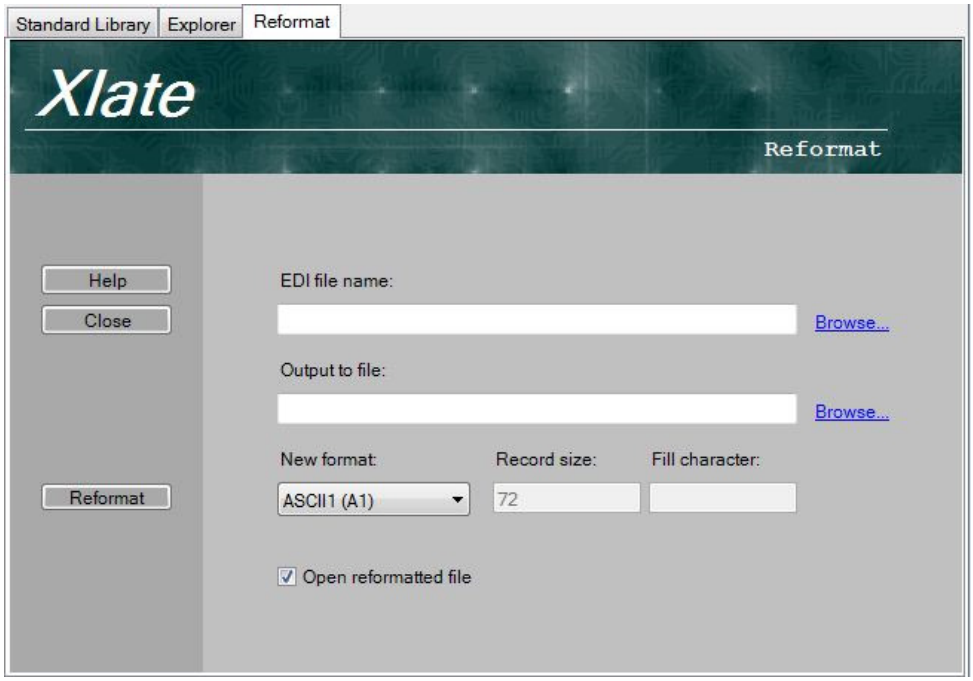
Open (generated) EDI file – if you select this tickbox, Xe will open a new window in the right-hand pane, containing the successfully generated EDI file.

Construct – click this button to initiate the construction of the in-house file into the EDI file.

Close – click this button to close the Construct window.

3.5.3 Reformat

The Xlate Reformat window is shown below. The Reformat window allows you to reformat an EDI file. The Reformat option is useful if you want to view or edit the contents of a file, as it allows you to reformat an EDI file into ASCII, which, for many editors, is the only format they will work with.



The fields and buttons in this window are:

EDI file name – if the EDI file to be reformatted is located in the Table Files Directory as provided in the Xlate Options dialog, you can simply type in the name of the EDI file. Otherwise, use the Browse option to provide the full directory and file path for the EDI file.

Output to file – if you want the reformatted file to be located in the Table Files Directory as provided in the Xlate Options dialog, you can simply type in the name of the reformatted file. Otherwise, use the Browse option to provide the full directory and file path for the reformatted file.

New format – use the dropdown arrow to select the new format for the EDI file. The options are as follows:

- **U** – unstructured. This option means that there is no file format – the file is written as a string of bytes. Carriage return (C/R) and Line feed (L/F) characters and leading spaces are dropped if the input file is ASCII format.
- **F** – fixed length records of 'record size' bytes are written and padded with 'fill character' where necessary. New messages are begun on 'record size' boundaries.
- **A1** – ASCII format 1. This option means that segments are terminated by C/R L/F to make the file more readable when displayed on a screen.
- **A2** – ASCII format 2. This option is as A1 above, but in addition C/R L/F are inserted every 'record size' bytes to make it possible to edit the file with a text editor that displays, for example, only the first 72 characters of a line.
- **A3** – ASCII format 3. In this option, C/R L/F are inserted every 'record size' bytes to make the file compatible with COBOL programs that require 'line sequential' input.

Record size – only valid for formats F and A2. Type in this field the record size to be written. The default value is 72.

Fill character – only valid for format F. This is the padding character to be used to ensure records are 'record size' bytes long and correctly aligned. Simply type in a single alphanumeric padding character to be used, without any quotes.

Open reformatted file – if you select this tickbox, Xe will open a new window in the right-hand pane, containing the successfully reformatted file.

Reformat – click this button to reformat the specified EDI file.

Close – click this button to close the Reformat window.

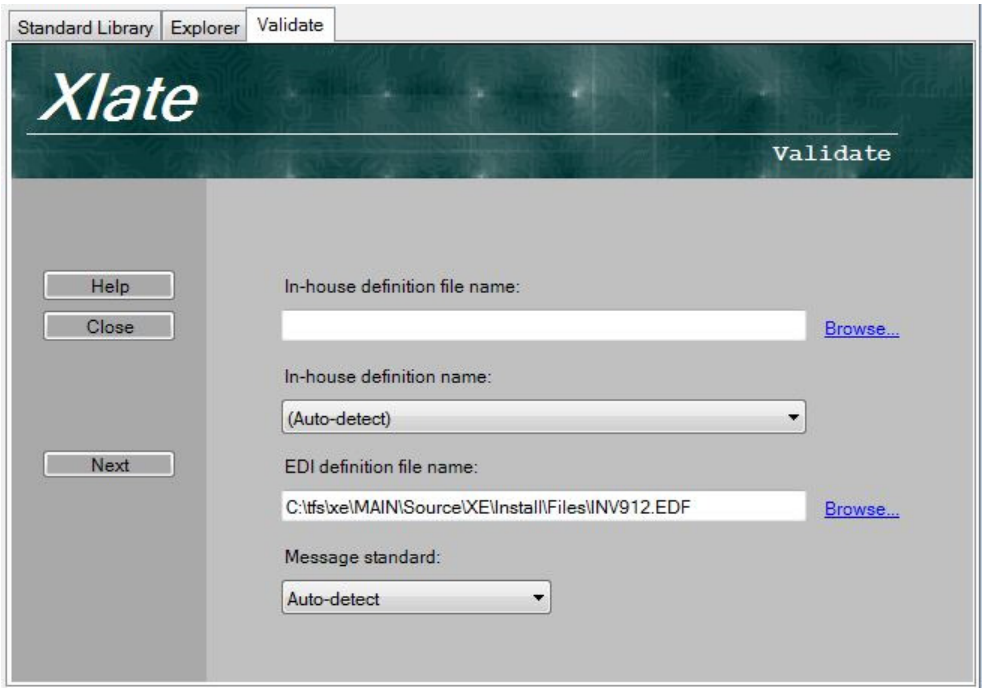
3.5.4 Validate

Having coded or generated your table definitions, this option can be used to validate them before you attempt translation or construction.

Output from validation is a formatted report of the tables, that can be printed or written to a file for further processing.

Input is either an in-house definition or a message definition table or both. When both kinds of definition are supplied, the links between the two are made and validated.

The Xlate Validate window is shown below.



Before using this window you should ensure that the appropriate entries have been made in the Xlate Index file. For details of the Index file and its requirements, please refer to the Xlate on-line Help.

The fields and buttons in this window are:

In-house definition file name – if your in-house definition file is located in the Table Files Directory as provided in the Xlate Options dialog, you can simply type in the name of the in-house definition file. Otherwise, use the Browse option to provide the full directory and file path for the in-house definition to be validated.

In-house definition name – use the dropdown arrow to select the appropriate in-house definition name. This is the name of an Index file HSE record that

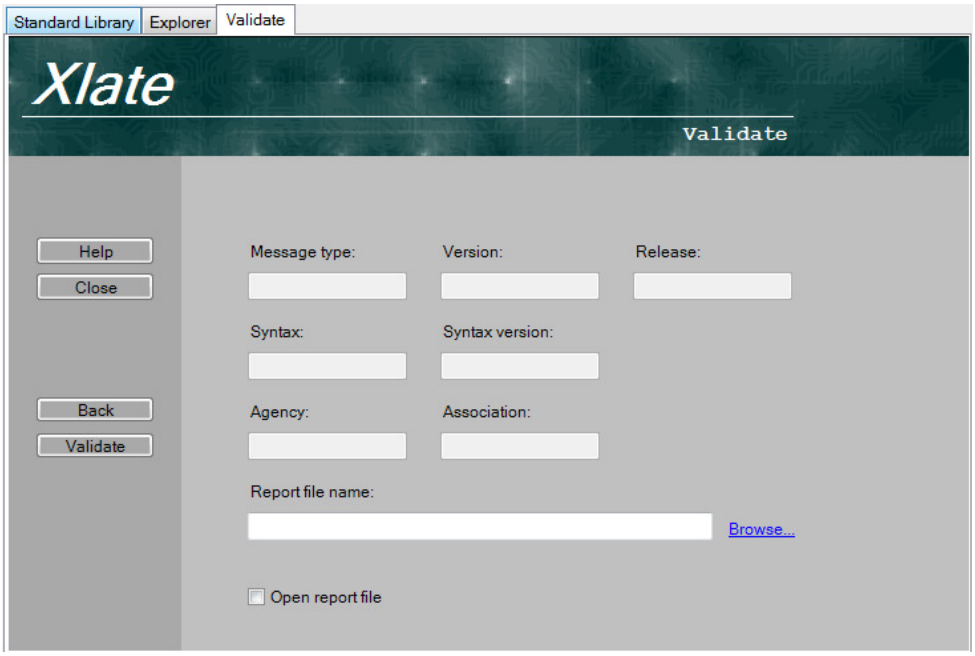
refers to the definition above. If there are no multiple entries for any definition files in the Index file, you can use the Auto-detect option instead, and Xlate will determine which in-house definition name should be used.

EDI definition file name – if your EDI file is located in the Table Files Directory as provided in the Xlate Options dialog, you can simply type in the name of the EDI file. Otherwise, use the Browse option to provide the full directory and file path for the EDI message definition to be validated .

Message standard – select the appropriate message standard using the dropdown arrow, or use the Auto-detect option.

Close – click this button to close the Validate window.

Next – click this button to see the next window of fields for this option. The next window is shown below.



If, on the previous Validate window, you selected a specific message standard, you will see that all the fields in this window are enabled. If you selected Auto-detect instead, only the Report file name field will be enabled.

The field names will differ according to whether you selected EDIFACT/UNGTDI or ANSI X12.

Message type / **Version** / **Release** / **Syntax** / **Syntax version** / **Agency** / **Association** – together these fields are the message identity of an Index MSG record that refers to an EDIFACT or UNGTDI standards definition file given in the previous window.

Transaction set / **Functional group** / **Functional group version** / **Agency** / **Interchange segment** / **Interchange standard** / **Interchange version** – together these fields are the message identity of an Index MSG record that refers to an ANSI X12 standards definition file given in the previous window.

Report file name – to view the report online, you have two options. If you want the output report file to be located in the Table Files Directory as provided in the Xlate Options dialog, you can simply type in the name of the report file. Otherwise, you must provide the full directory and file path for the report file that is to be generated. You can either type this in or use the **Browse** button to select a file path.

Open report file – if you select this tickbox, Xe will open a new window in the right-hand pane, containing the successfully generated report file. We suggest that you make use of this option, to avoid any potential difficulty opening the generated file from Windows Explorer ®.

Back – this button will return you to the previous Validate window.

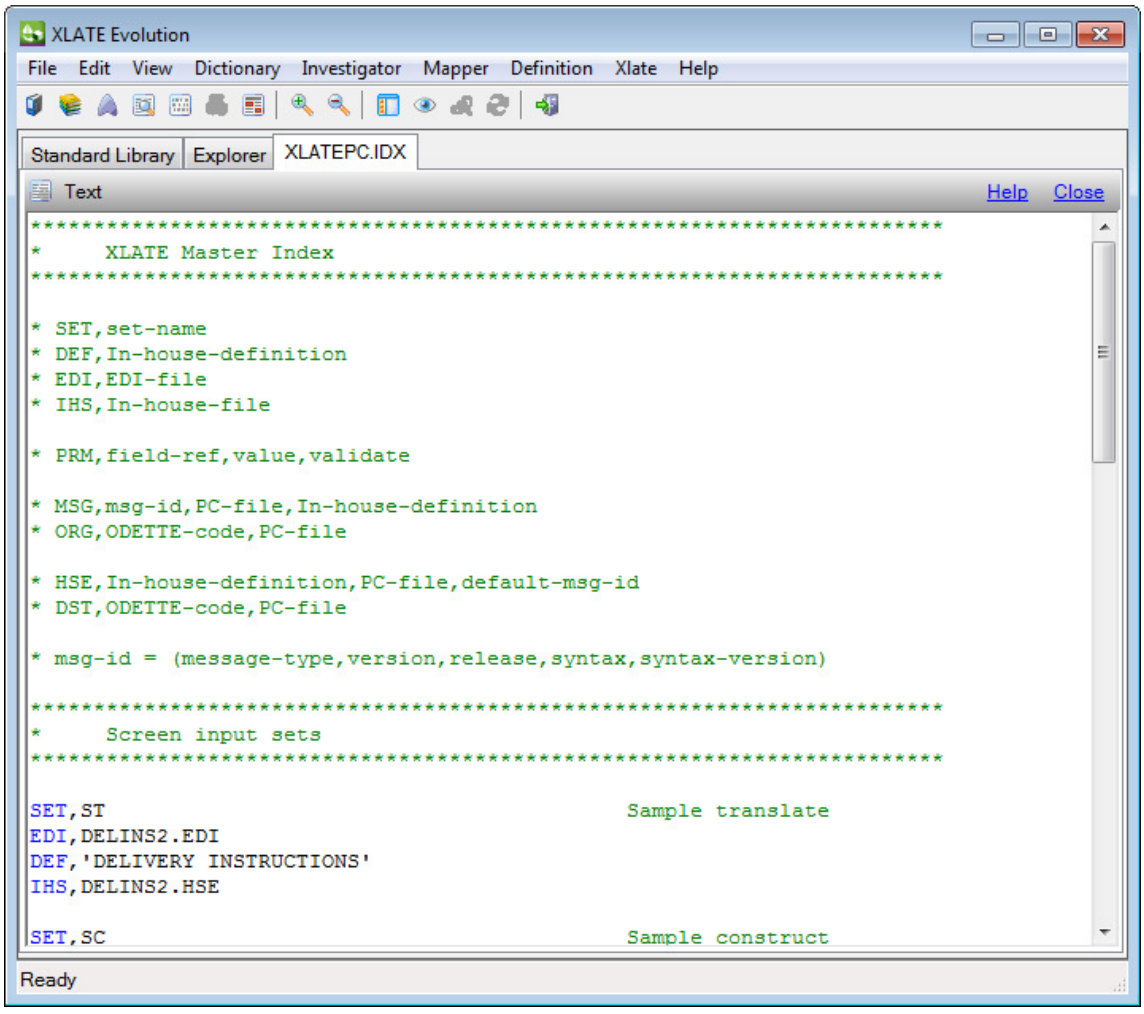
Validate – click this button to perform validation on the file(s) you have specified in the previous window.

Close – click this button to close the Validate window.

3.5.5 Xlate Index File

Xe is shipped with an existing Xlate index file (Xlatepc.idx), which can be found in the Xe installation directory. It contains several basic examples of how to format entries for the Xlate index.

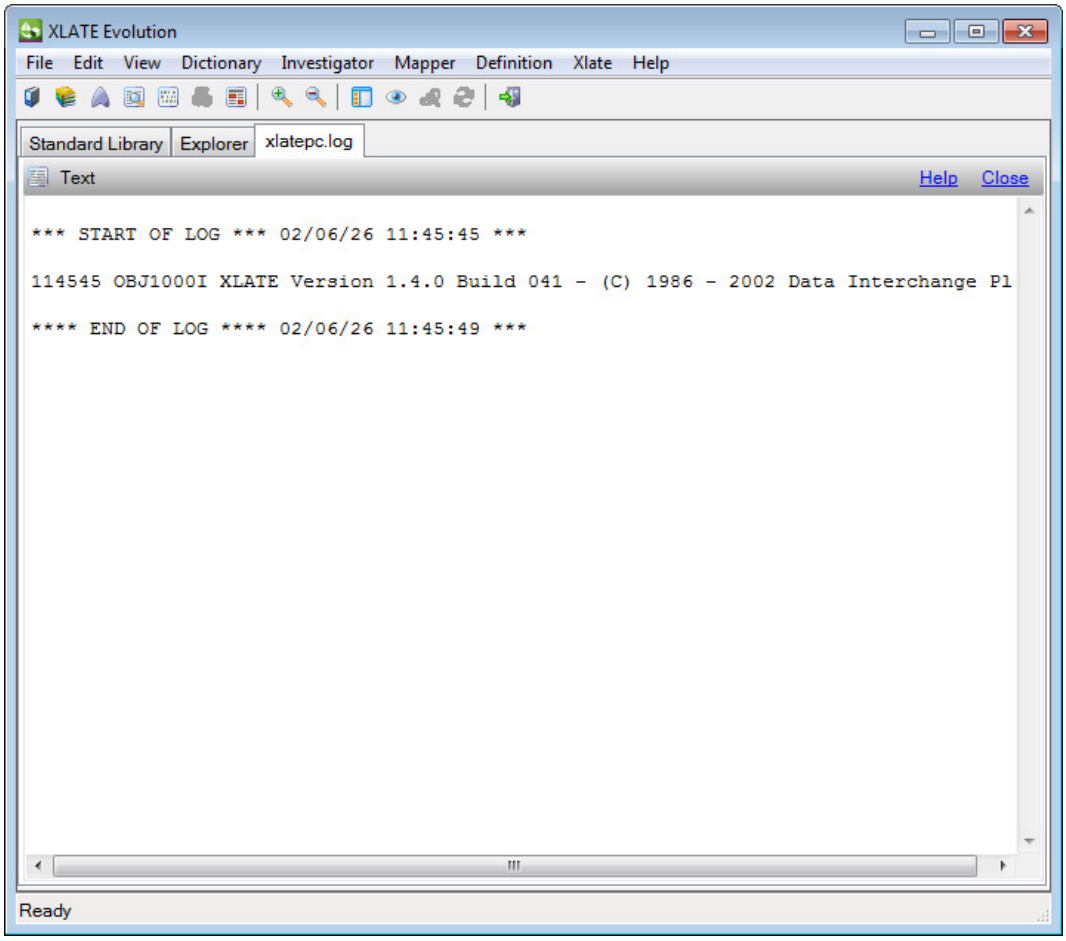
To view and edit the Xlate Index file, select **View >> Index File** from the main menu, or press **Alt+V+I**.



For full details of the Index file and its requirements, please refer to the Xlate on-line Help.

3.5.6 Xlate Log File

The contents of the Xlate log file (xlatepc.log) can be viewed by selecting select **View >> Log File** from the main menu, or press **Alt+V+L**.



The log file is overwritten each time a different Xlate function is performed (e.g. Translate, Validate etc), so if you want to retain any logged information you should print or copy the log file before you perform another Xlate function.

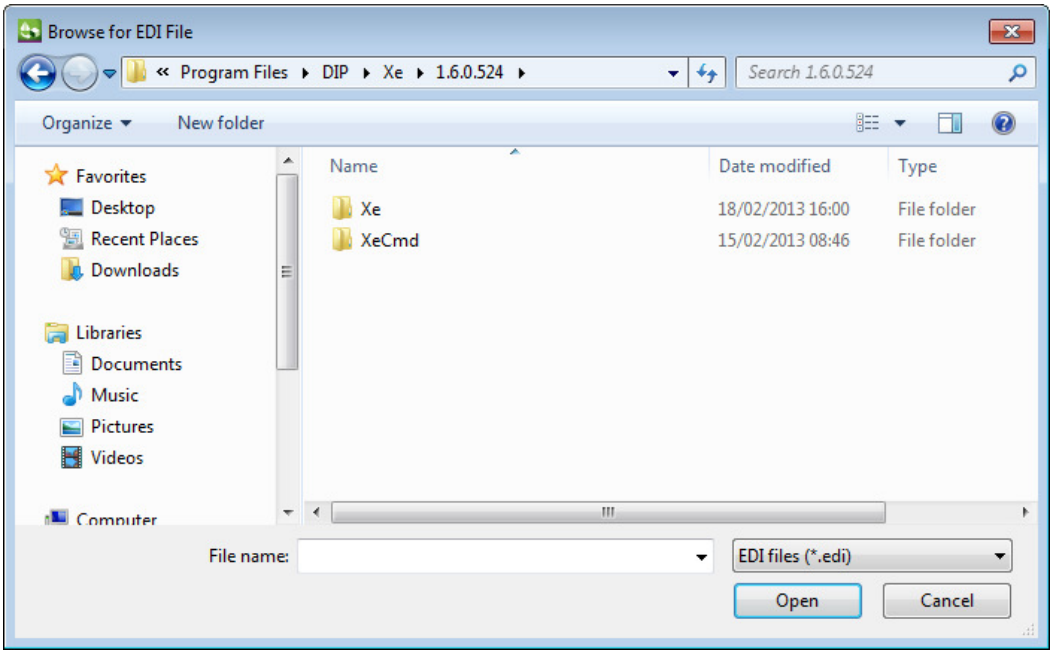
3.6 Investigator

The Investigator actually comprises three elements: the File Investigator, the Hex Viewer and the Object Packager.

3.6.1 File Investigator

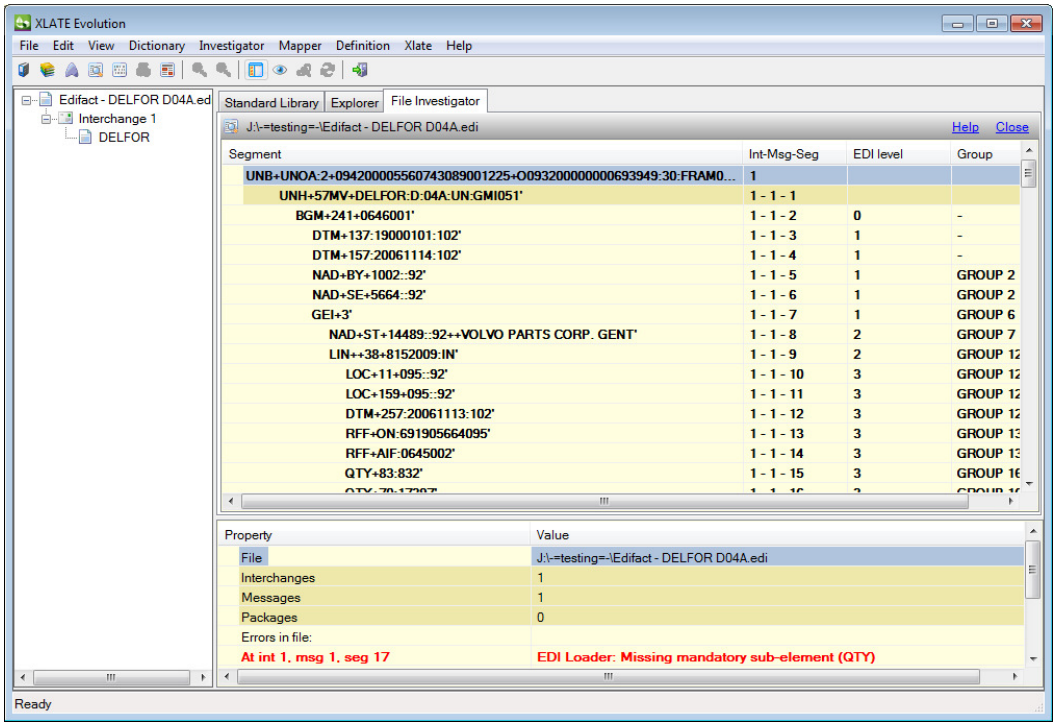
The File Investigator examines one EDI file at a time and shows you the contents in a user-friendly format, together with warnings if there are any errors in the file.

Select **Investigator >> File Investigator** from the main menu, or press **Ctrl+I**, or click the File Investigator icon  on the toolbar. This will bring up the 'Browse for EDI File' dialog:



Use this dialog to select an EDI file to be investigated. The file extension does not have to be .edi, but the file must contain one or more valid EDI messages whose syntax can be recognised by Xe, in order for the investigator to work.

Select an EDI file from your system and click **Open** to see the File Investigator window in the right-hand panel, as shown in the example below.



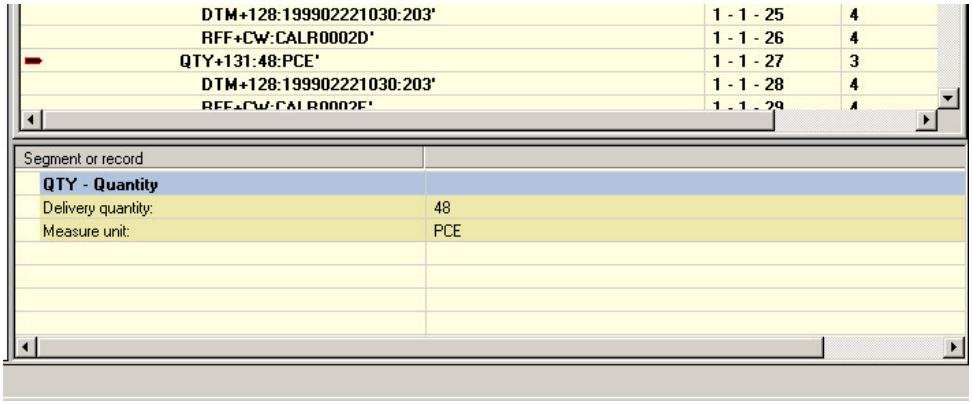
The tree view in the left-hand panel now shows a representation of the interchanges and messages within the EDI file.

The right-hand panel, now showing the File Investigator window, has been divided into two.

The top section shows the entire contents of the EDI file (you may need to use the vertical scroll bar to see it all).

The lower section initially shows the details of the EDI file under investigation: its full path and file name, and the number of interchanges, messages, packages and errors contained in it.

If you highlight any of the segments in the top section, the File Investigator will attempt to interpret the data contents of the segment in the lower section and display them in a meaningful way. For example, in an EDIFACT message, the meaning of data is often indicated by a data qualifier, without knowledge of which the data is meaningless. So for a QTY (quantity) segment such as QTY+131:48:PCE' (indicated by the red arrow in the sample below), the File Investigator will tell you that this indicates a Delivery Quantity of 48 pieces, as shown at the bottom of the sample below.

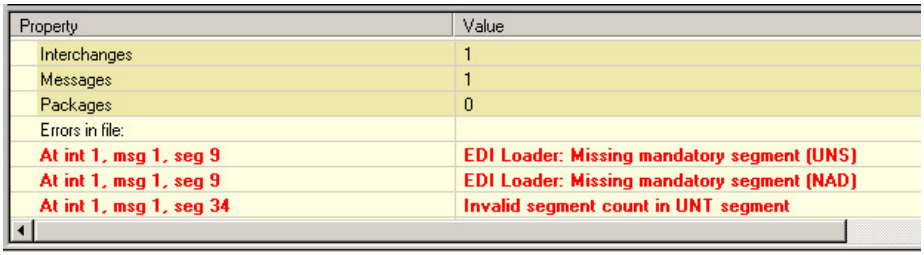


Use the **Close Window** button on the right-hand pane to close the window.

3.6.1.1 Errors in file

If the EDI file under investigation contains any errors detectable by Xe, this will be indicated by a red exclamation mark in the tree view, immediately below the file containing the error(s).

Any errors resulting from a mismatch between the Dictionary definition of the message and the EDI message itself will be displayed in red in the lower section of the File Investigator window, as shown in the example below.



These errors are highlighted while you have the actual file selected (highlighted) in the tree view. Field errors can be seen by clicking on the specified segment in the upper section of the File Investigator panel.

3.6.1.2 Packages

You will see that one of the properties listed in the lower section of the File Investigator window is 'Packages'. The term ‘Packages’ refers to EDIFACT objects that can be included in an EDI file. For more details, please refer to the section entitled "Object Packager".

3.6.1.3 Menu options

The File Investigator has the following menu options.

View As Text – This option opens up a new window in the right-hand panel, bearing the name of the file under investigation. This window displays the file segment by segment, without any indentation.

View Structure Diagram – This option opens up a new window in the right-hand panel, bearing the name of the message type whose structure is displayed there. All segments in the message hierarchy are shown, but segments actually used in the message are highlighted by being displayed in yellow. If you right-click on a segment in the hierarchy, you will see a further option – Locate in Investigator – which will take you to the File Investigator window and highlight the selected segment.

Sign Entity – This option allows you to sign an EDIFACT v4 interchange, message or group. To do this you will require the necessary security hardware, such as a smartcard and reader.

Save Object – This option allows you to save an object that you have inserted in an EDI file. You will be asked to provide a file path for the object to be saved to.

Colour Errors – This option, if selected, displays in red any segments containing errors. Otherwise they are displayed in black.

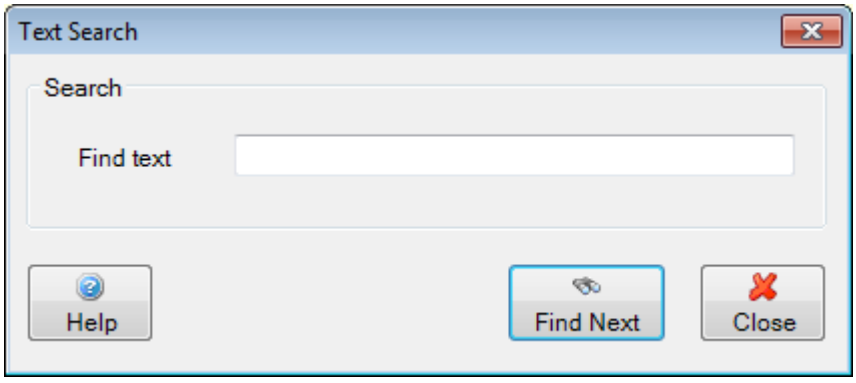
Escape Data – If you select this option, any escape characters contained in the file under investigation will be suppressed. If you deselect this option, they will be shown.

Indent Data – This option toggles between indenting and not indenting the data. Indentation gives you a better feel for the hierarchical structure of the message.

Format Segments – If this option is selected, the lower section of the File Investigator will display only the essential segment data together with its description. If this option is deselected, the lower section will display each composite and element name, together with the data contained in those composites and elements and their meaning.

Create Log – This option, if selected, will create a log file each time you initiate the File Investigator. This log file, called log.txt, can be found in the Xe installation directory. Any existing log.txt file will be overwritten.

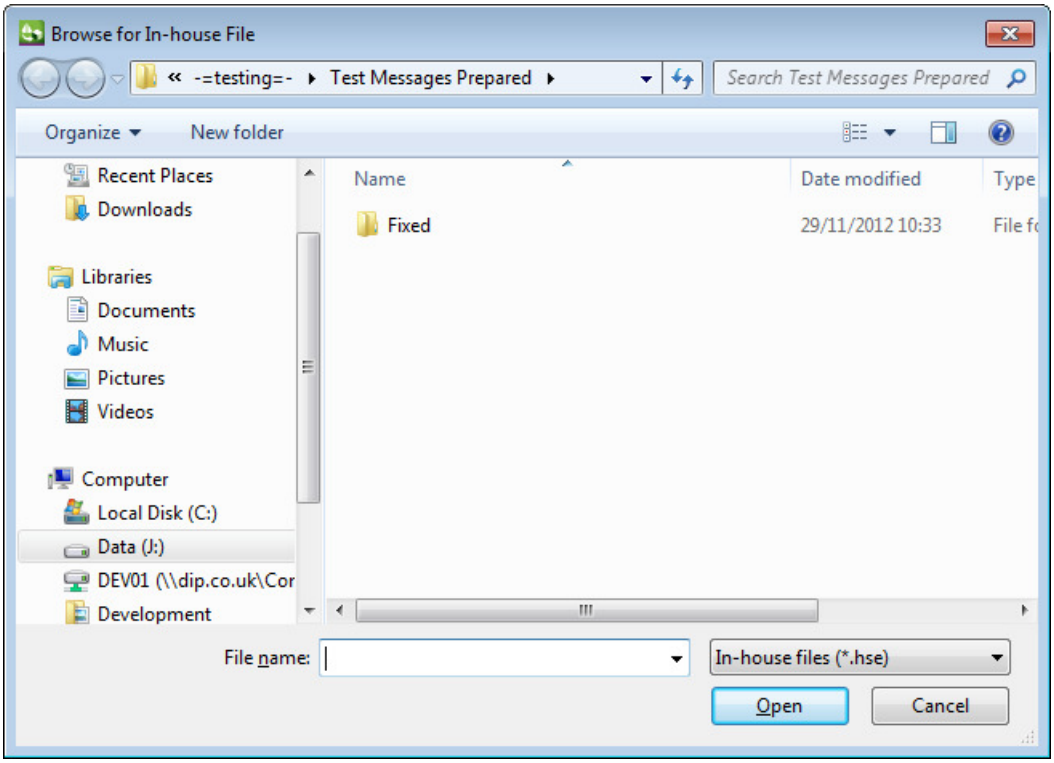
Text Search – This option allows you to search for specific text in the message. If you choose this option, you will see the following dialog.



Simply type in the exact text that you want to find and click the **Find Next** button. Click **Close** when you have finished searching.

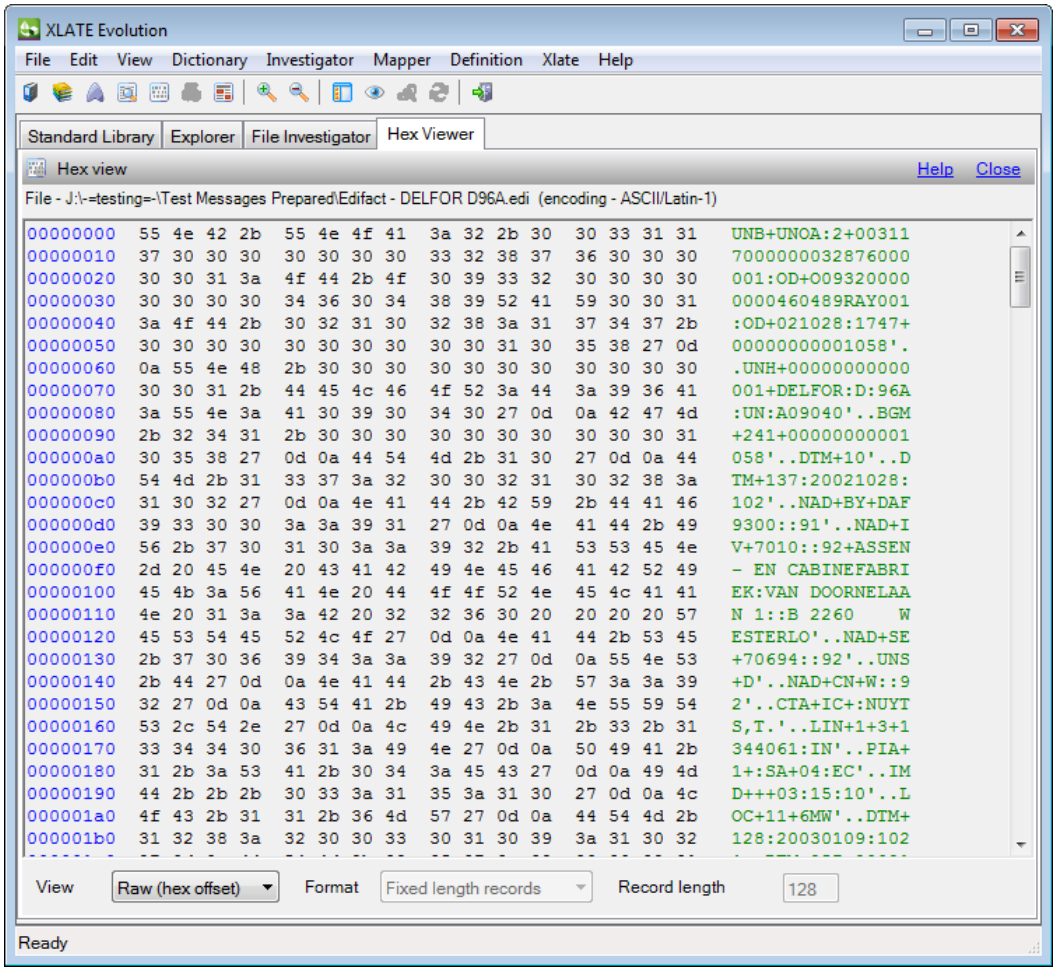
3.6.2 Hex Viewer

The Hex Viewer allows you to view any of your in-house files in hex format. Select **Investigator >> Hex Viewer** from the main menu, or press **Ctrl+J**, or click the Hex Viewer icon  on the toolbar, to bring up the following dialog.



The Browse for In-house file dialog allows you to select the in-house file you want to view in hex format. Select the file by double-clicking it or by highlighting it and clicking the **Open** button. The Hex Viewer window will then be opened in Xe, displaying the file in hex format, as shown in the example below.

Please note that the file may have any extension you choose. Simply select ‘All files (*.*)’ in the ‘Files of type’ field if you want to view a file that has an extension other than .hse.



This window will fill the entire Xe screen, but you can click the Toggle icon to display the tree view as usual if you wish.

This window is made up of three sections, which will be displayed differently according to the view options that are selected at the bottom of the screen.

The first two display options are: **View** and **Format**. The name of the third option field will change to reflect whichever Format you have selected, though the field will not be enabled if you have selected one of the Raw view options.

View allows you to change the way in which the data is displayed. The options are:

- Raw (hex offset)
- Raw (decimal offset)
- Formatted (text)
- Formatted (full)

Format works in conjunction with either of the Formatted view options and allows you to change the format in which the data is displayed. The options are:

- Fixed length records
- Variable length records
- Record delimited
- Character separated values
- Character delimited fields

If you choose one of the Formatted view options, the third view option field will have one of the following values:

- Record length
- Record delimiter
- Field delimiter

Record length allows you to specify the length of the records if you have selected “Fixed length records” as the Format.

Record delimiter allows you to specify the record delimiter if you have selected “Record delimited” as the Format.

Field delimiter allows you to specify the field delimiter if you have selected “Character separated values” or “Character delimited fields” as the Format.

N.B. Please note that the record delimited format may have an undesired effect if the delimiter you choose in the third view option is not actually used as a delimiter in the file.

These options are available to allow you to experiment with display formats, in case you are dealing with a file whose format you do not know. By selecting different combinations of these options, you may be able to determine the best way to display the file.

When you have selected the required options, click once with the left mouse button to see the result of your selection.

Use the **Close Window** button on the right-hand pane to close the window.

3.6.2.1 View option

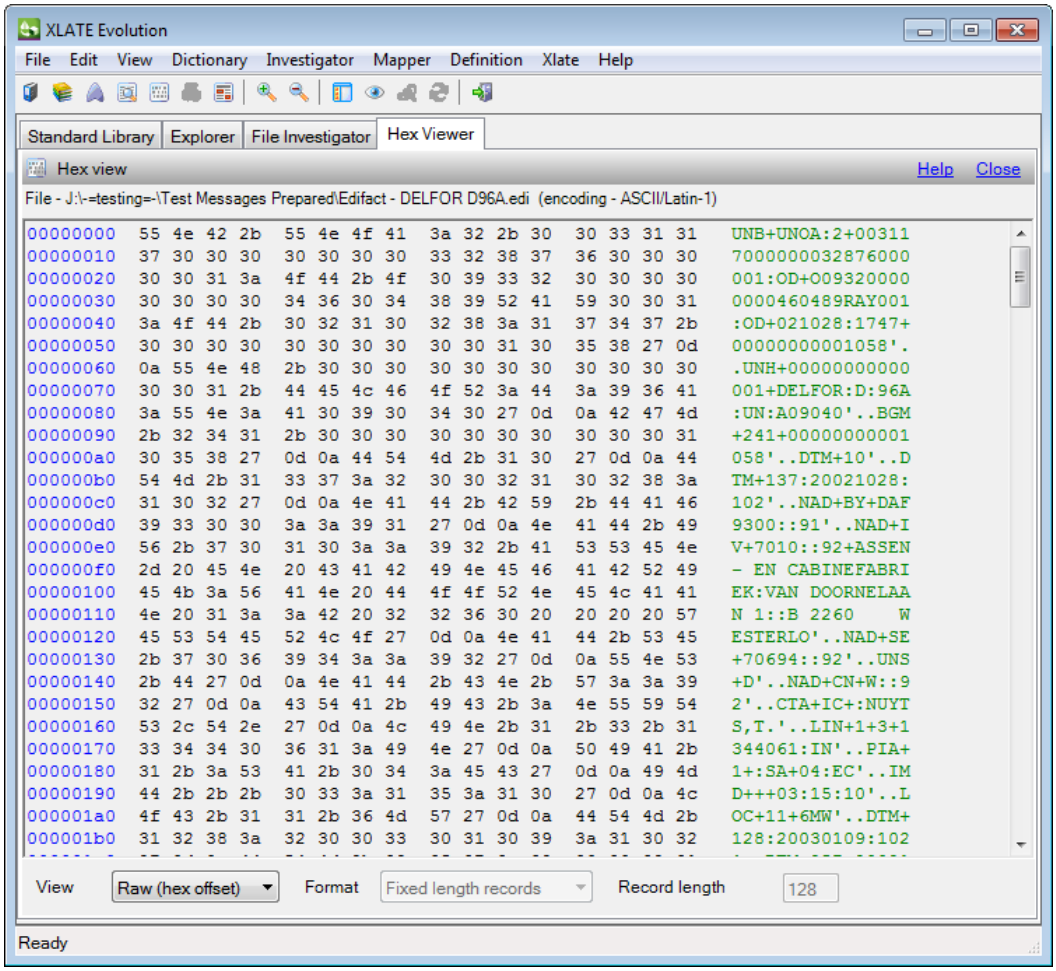
Raw

For both Raw (hex offset) and Raw (decimal offset), the sections are displayed as follows.

On the left is a blue column, which gives the character offset in hex or decimal format.

In the middle is a black section, which displays the hex representation of the selected file. Sixteen characters are formatted on each line. These are divided into four blocks of four columns, for better legibility.

On the right is a green section, which displays the actual content of the selected file. The contents of each green line equates to the contents of each black line. An example of a file displayed as Raw (hex offset) is shown below.



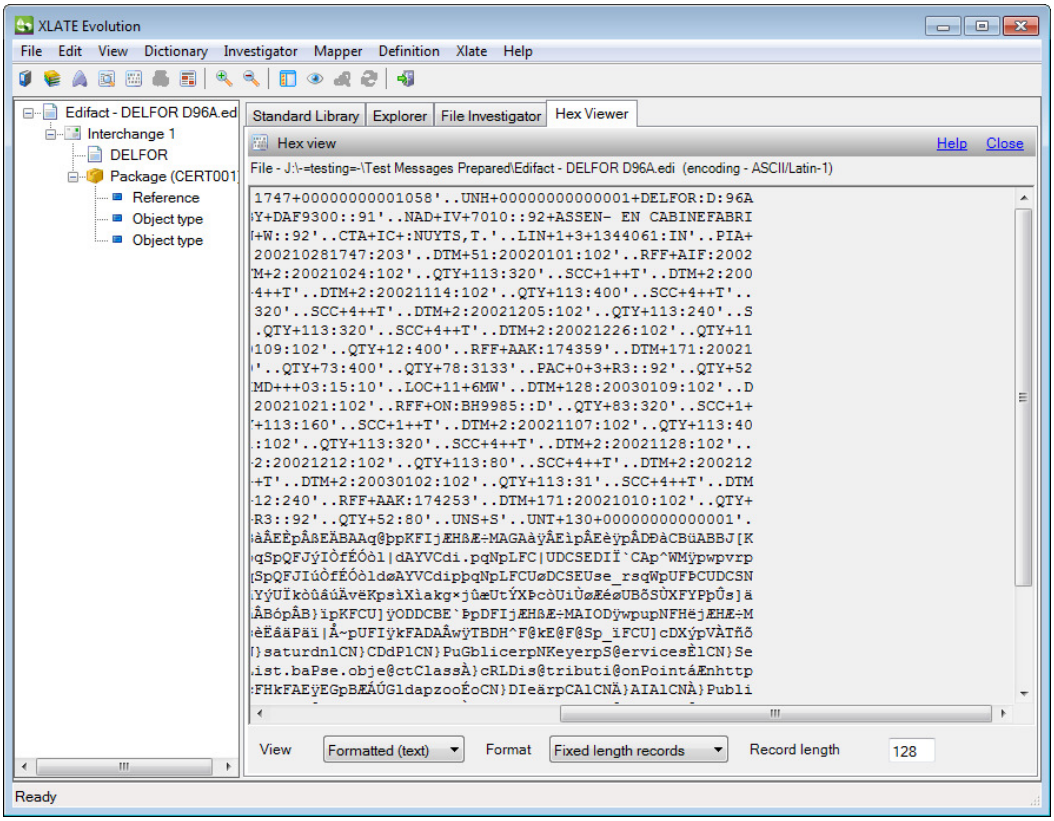
If you select either of the Raw options, the Format and Record Delimiter options will remain disabled.

Formatted

The Formatted options display the file according to the format you select and the selected value in the third view option.

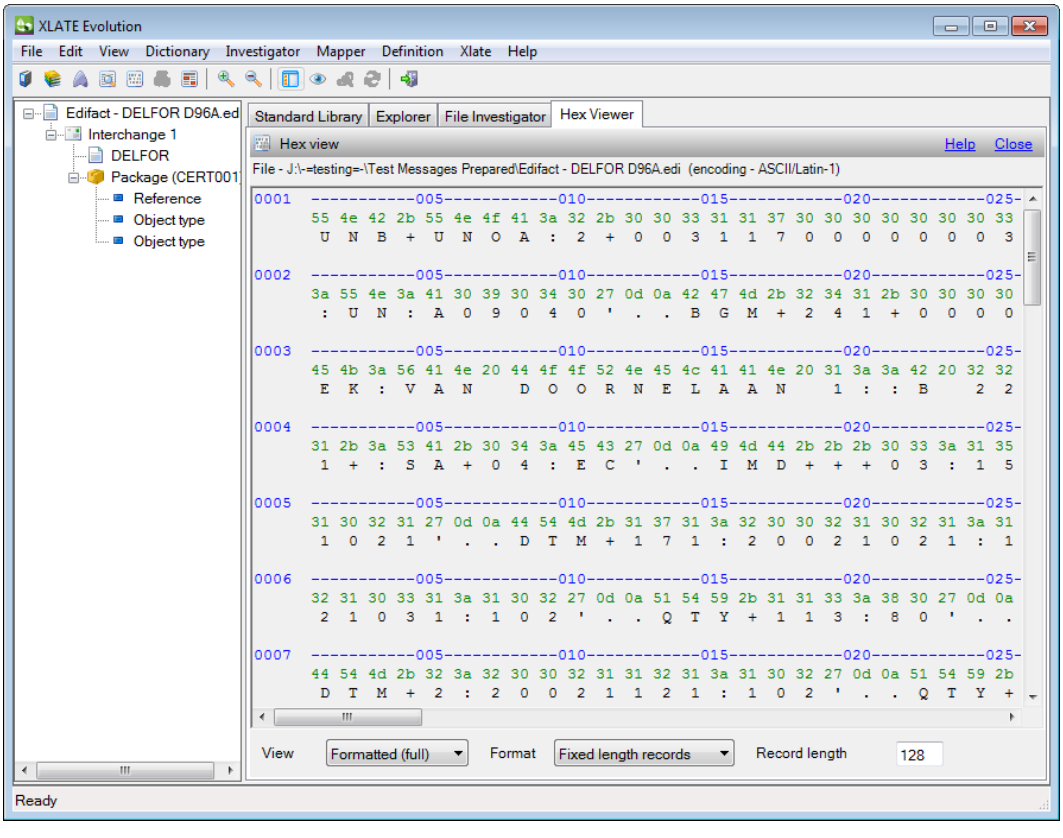
For example, if you choose “Record delimited” and the appropriate record delimiter is selected in the third view option, the file will be displayed record by record, with each new record beginning on a new line.

For Formatted (text), only two sections will be displayed. This time the blue section on the left displays a sequential record number and the black section on the right shows the contents of each record. An example of a file displayed as Formatted (text) with records delimited by CRLF is shown below.



For Formatted (full), three sections will be displayed. Whereas in the Raw View option the sections are displayed across the window, in this option the sections are displayed down the window, repeating for each record.

An example of a file displayed as Formatted (full) with records delimited by CRLF is shown below. As you can see, each record is displayed in three lines: first is a blue line showing the character offset, next is a green line showing the hex format of the record, and last is a black line showing the actual record content.



For both Formatted (text) and Formatted (full), the name of the third option field will change to reflect whichever Format you have selected. However, depending on which Format is selected, the name of the third option field may or may not be enabled. If it is enabled, you may type in an appropriate value.

If you select Fixed length records, you may type in a record length.

If you select Variable length records, the third option field will be disabled.

If you select Record delimited, you may type in a record delimiter. Note that a carriage return/line feed (which is the default) should be indicated by {crlf}.

If you select Character separated values, you may type in the appropriate character that is used to separate each value in each record.

If you select Character delimited fields, you may type in the appropriate character that is used to separate each field in each record.

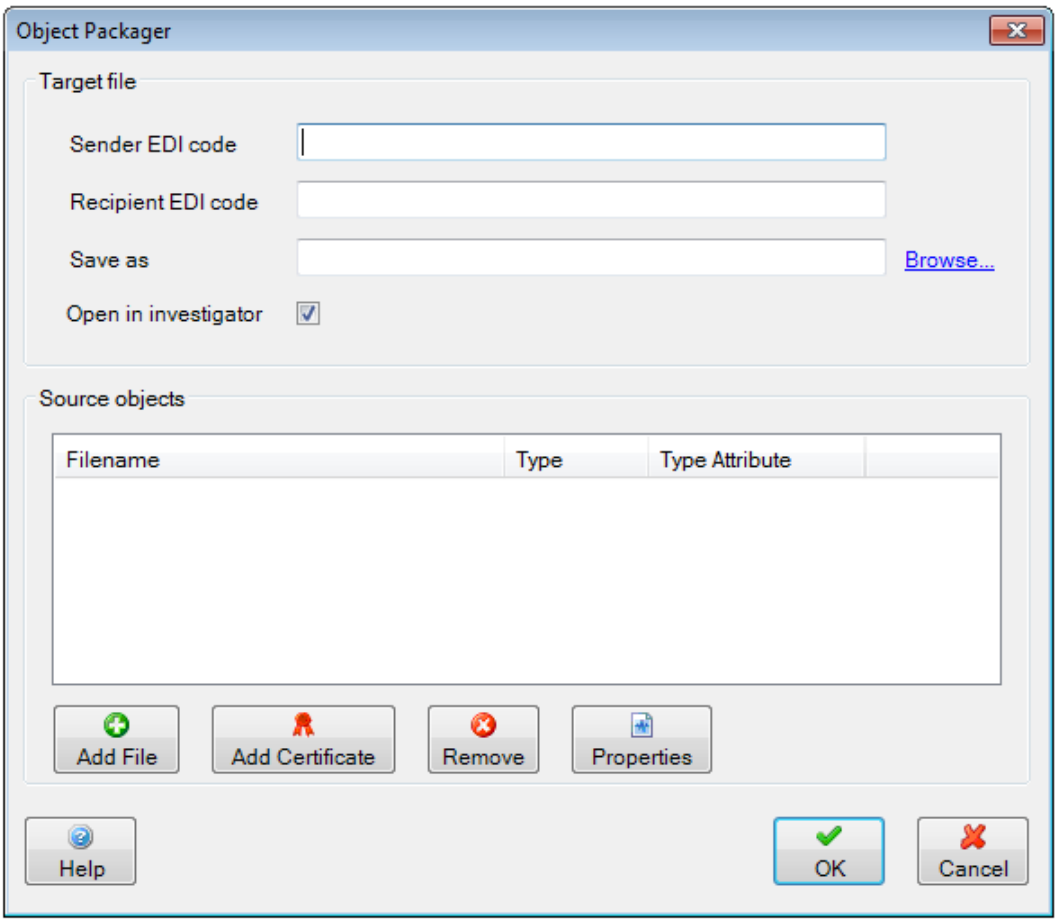
3.6.3 Object Packager

The Object Packager allows you to create an EDI file containing one or more objects. An object is an EDIFACT concept introduced in version 4 of the standard. Adding an object results in the insertion of a UNO-UNP segment pairing into the EDI file, at the same level as a message i.e. within a group or interchange. A UNO-UNP pairing is inserted for each file or certificate added to the file and acts as an “envelope” for the object.

A UNO segment indicates the beginning of an object. It contains a value that is the length of the inserted object. A UNP segment indicates the end of an object. It also contains a value that is the length of the inserted object. In between the two segments is the inserted object.

An object may be another file or a security certificate. A security certificate is packaged up as the public key of the file, so in this way the Object Packager can be used to send public keys to your trading partners.

Select **Investigator >> Object Packager** from the main menu, or press **Ctrl+K**, to bring up the following dialog.



The target file section allows you to specify who the EDI file is from, who it will be sent to, and where it should be saved.

Target file – Sender EDI code

Type in this field the EDI code of the sender of the file. The Object Packager will insert this value in the Sender EDI code element of the UNB.

Target file – Recipient EDI code

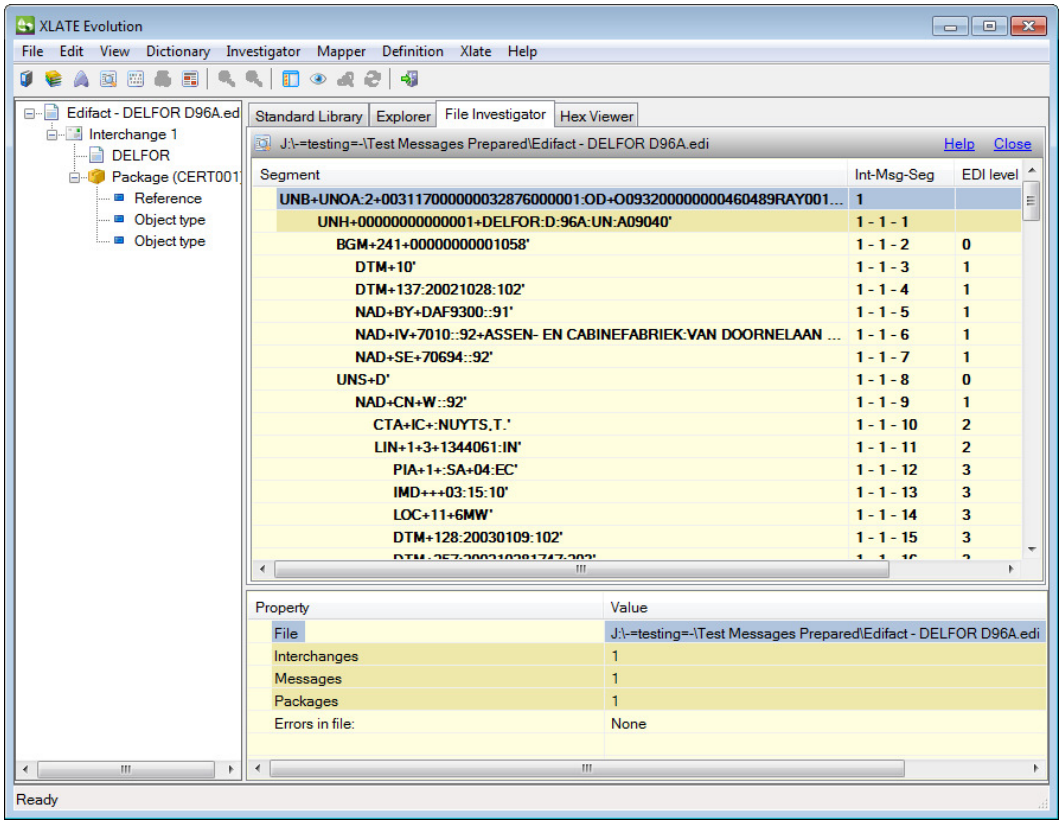
Type in this field the EDI code of the intended final recipient of the file. The Object Packager will insert this value in the Recipient EDI code element of the UNB.

Save As

Either type in the full path and filename of where you want the file to be saved, or use the **Browse** button to select an existing file location.

Open in investigator

Select this tickbox if you wish to open and view the created file in the File Investigator. An example of such a file displayed in the File Investigator is shown below.



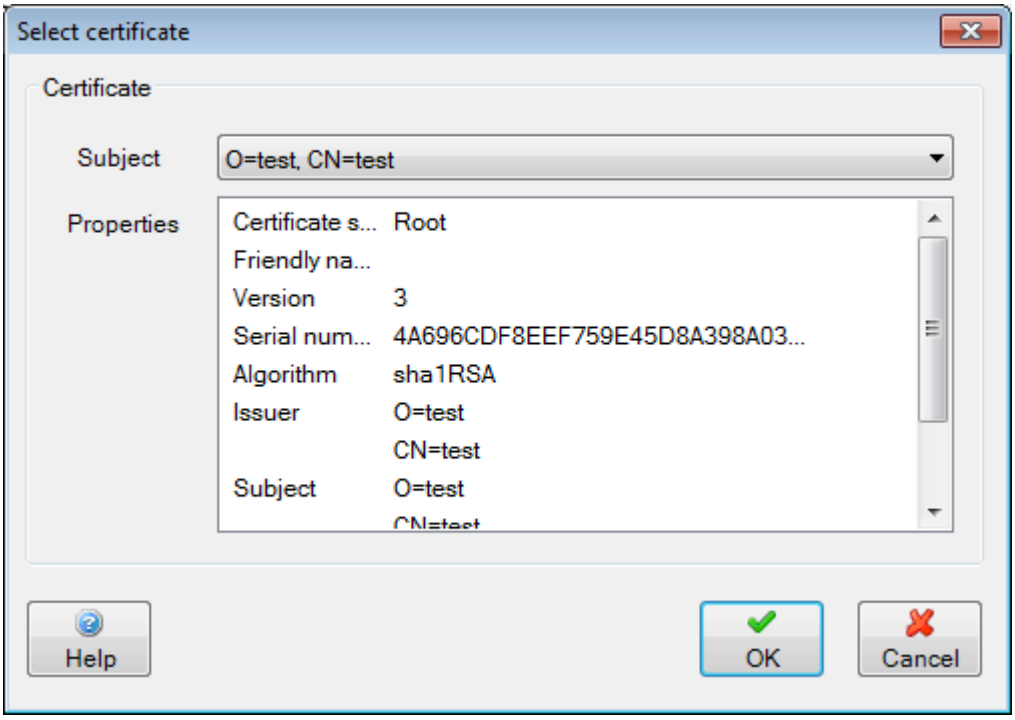
This shows the UNB with the Sender and Recipient EDI codes designated by you, and two objects inserted into the EDI file.

Add File

Click this button to bring up a Browse for File dialog.

Add Certificate

Click this button to bring up the Select certificate dialog, as shown below.



Use the dropdown arrow to select a certificate in the Subject field.

The Properties area will display any properties of the certificate you have selected.

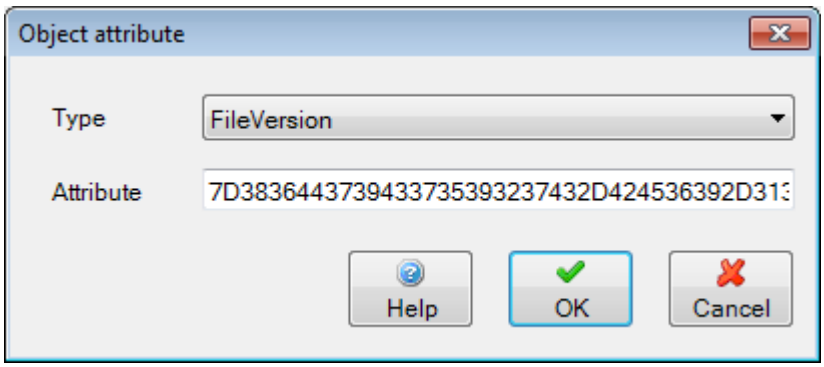
Click **OK** to return to the Object Packager dialog.

Remove

To remove an object from the file you are creating, highlight an entry in the Source Objects area and click **Remove**.

Properties

To see the properties for any object you have included in the file you are creating, highlight an entry in the Source Objects area and click **Properties**. This will bring up the Properties dialog, shown below.



You may edit the attribute of any type if you wish. Use the dropdown arrow to select the attribute type, then type the appropriate attribute value in the field below.

Click **OK** to save your changes or **Cancel** to close this dialog without saving your changes.

3.7 Mapper

3.7.1 Introduction

The Mapper is a powerful new feature of Xe, intended for use when Xlate cannot provide the flexibility you need.

Using a combination of drag-and-drop and built-in functions, it allows you to create file transformations based on a source file definition and a target file definition.

Once you have loaded up the source and target file definitions, the Mapper GUI allows you to map the fields or data elements from the source file definition into the target file definition and see a visual representation of the map on the GUI.

Alternatively, you can edit an existing map containing details of a file to be mapped from (source file definition) and a file to be mapped to (target file definition).

Behind the scenes, the Mapper creates a script file which is used to convert the mapping into a DLL. This DLL can then be executed against the source file in order to create the required target file.

3.7.2 Features

The main features of the Xe Mapper are:

- Support for a wide range of file types: EDI (EDIFACT, ANSI X12, VDA, UNGTDI), flat files, CSV, XML, SAP Idocs, TRA format.
- Ability to map from any file type to any other file type in a single pass.
- Performs multiple maps on a single source file.

- Uses a mapping script to generate a .NET DLL to perform each map.
- Built-in functions that provide support for common actions, such as manipulating text data and testing conditions.
- Ability to customise maps by using C# or VB.NET code in the mapping script.
- Splits output files across multiple files according to configured rules.
- Uses tables to define and validate EDI and in-house files.
- Supports optional validation of XML files using W3C conformant Schemas.

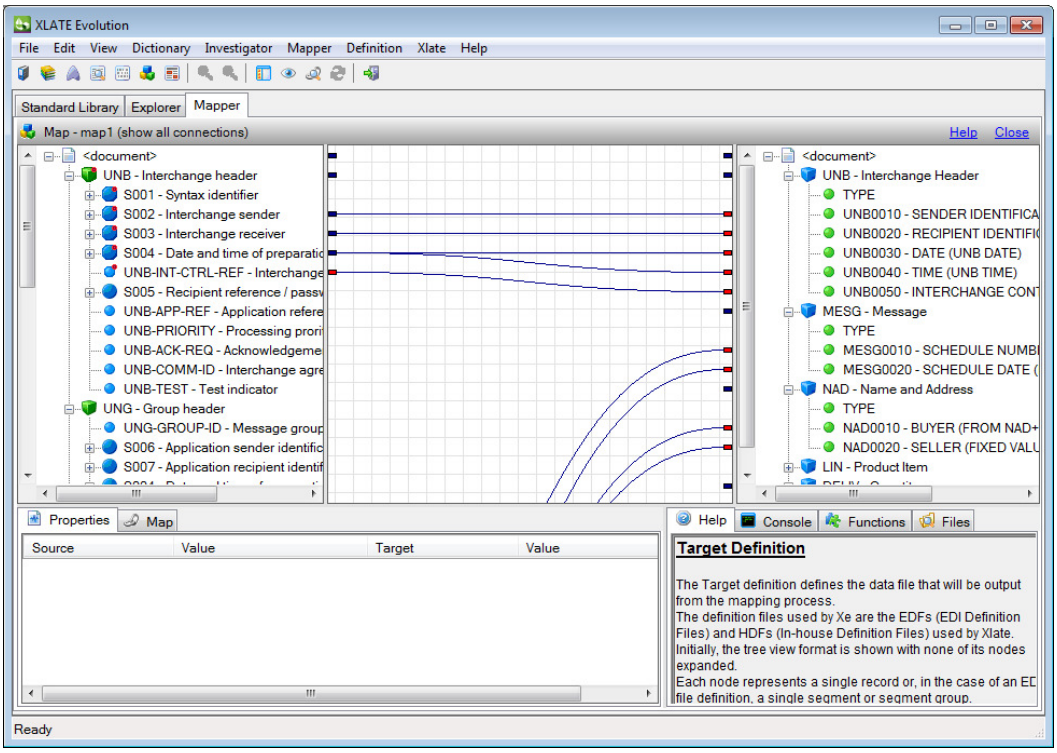
Input may be in ASCII, EBCDIC, Unicode, UnicodeBe or UTF-8 format. The default output format is ASCII, but overrides may be used to create the output in any of those formats.

Xe mapping makes use of stream processing to create and validate sequence numbering such as Interchange Control References. Values of such numbers are maintained by Xe for future use, thus ensuring that sequential numbering can be controlled for each individual trading partner.

Xe also provides the ability to set up counters, which are available throughout each individual mapping process. Counters allow you to maintain a running total, for example, which can then be mapped at the end of the mapping process.

3.7.3 The Mapper GUI

Once you have loaded a map project, the Mapper GUI will appear, as shown in the example below.



This window comprises five sections. Top left is a tree view display of the file definition to be mapped from (source file). Top right is a tree view display of the file definition to be mapped to (target file).

Between these two sections is the mapping area. Once a map has been created or loaded, and the mapped nodes are visible, this area will show mapping links between the nodes being mapped from and the nodes being mapped to.

Bottom left is an area showing the properties and values of highlighted items from the source and target files or of global items, while bottom right are Help, Console and Function page tabs. The Help page displays context-sensitive help for any item you click on. The Console page tab displays logging information generated by the **F6**, **F7** and **F8** processing options. The Function tab displays Xe built-in functions that can be used to manipulate data.

Each area is described in detail in the appropriate sections of the chapter entitled “Using the Mapper”.

3.7.4 Mapping

There are two ways to use the Mapper:

- Create a new map project by specifying a source file definition, a target file definition and a path and filename for the new map file
- Load an existing map project for viewing, editing or running

Loading a map file, which contains a reference to the source file definition and the target file definition, provides you with ready-made mapping links between the source file definition and target file definition. You can still edit and manipulate the mapping further if you wish, and generate a new map file from your changes.

Loading a source file definition and a target file definition allows you to set up the mapping links from scratch. To set up a simple one-to-one mapping link from a source node to a target node, simply left-click on the node to be mapped and drag it across to the target node (all the way into the right-hand panel). You can also achieve the same thing by dragging from the target to the source, since the Mapper will work both left-to-right and right-to-left.

Any changes you make, whether to a new map or existing map, should be saved if you do not want to lose them. To save them, simply use the Generate Script option (**F5**).

Drag-and-drop will achieve the simplest mapping requirements. More complicated requirements can be achieved via the Properties and Mapping page tabs, built-in functions, or some simple coding.

3.7.5 Help

This area of the window displays context-sensitive help. Simply click your mouse on the area or item you want to know more about, and the relevant help information will appear. Links within the context-sensitive help will take you to related items of information.

3.7.6 Console

This area of the window displays General log messages, including error messages, produced when you generate a map assembly or when you test a map. For more detailed error messages you should refer to the log file you specified on the Configuration – Logging page.

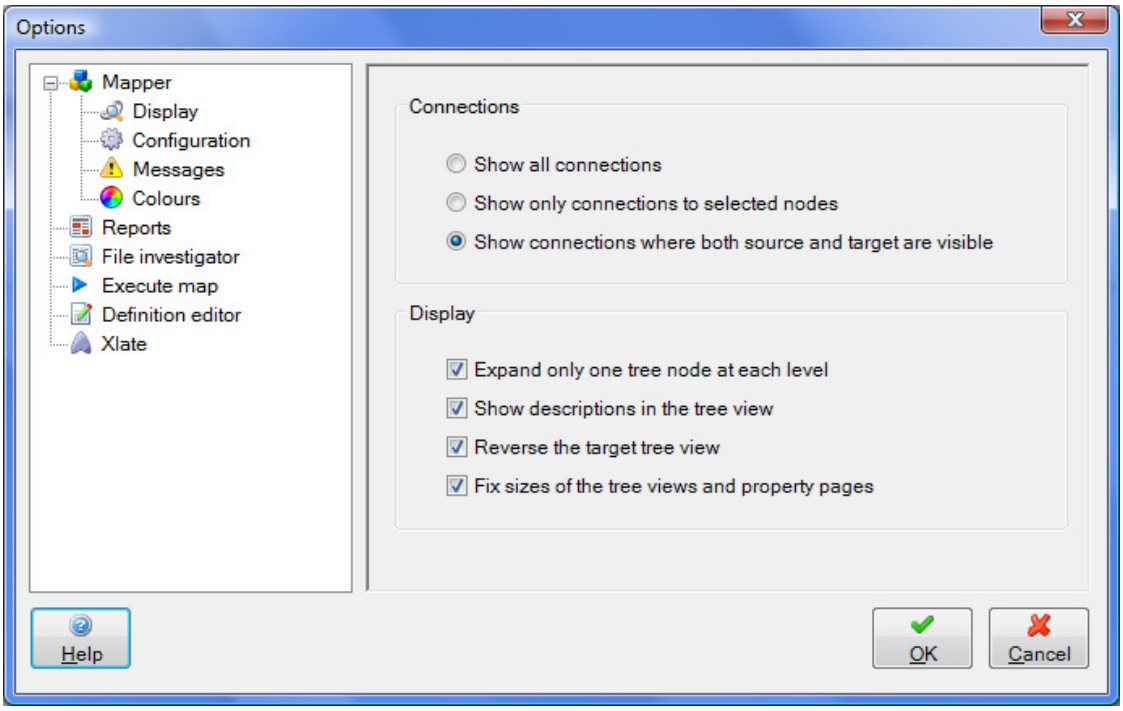
3.7.7 Functions

This area of the window shows the built-in functions available in Xe. These functions can be used to carry out common actions, such as manipulating text data and testing conditions. Functions can be filtered according to category.

They are used by dragging a function from the tab onto the main mapping area and then creating connections to and from the function as required.

3.8 Options

The Options dialog allows you to specify preferences for various components of the Xe application. It is accessed using the **File > Options** menu item.

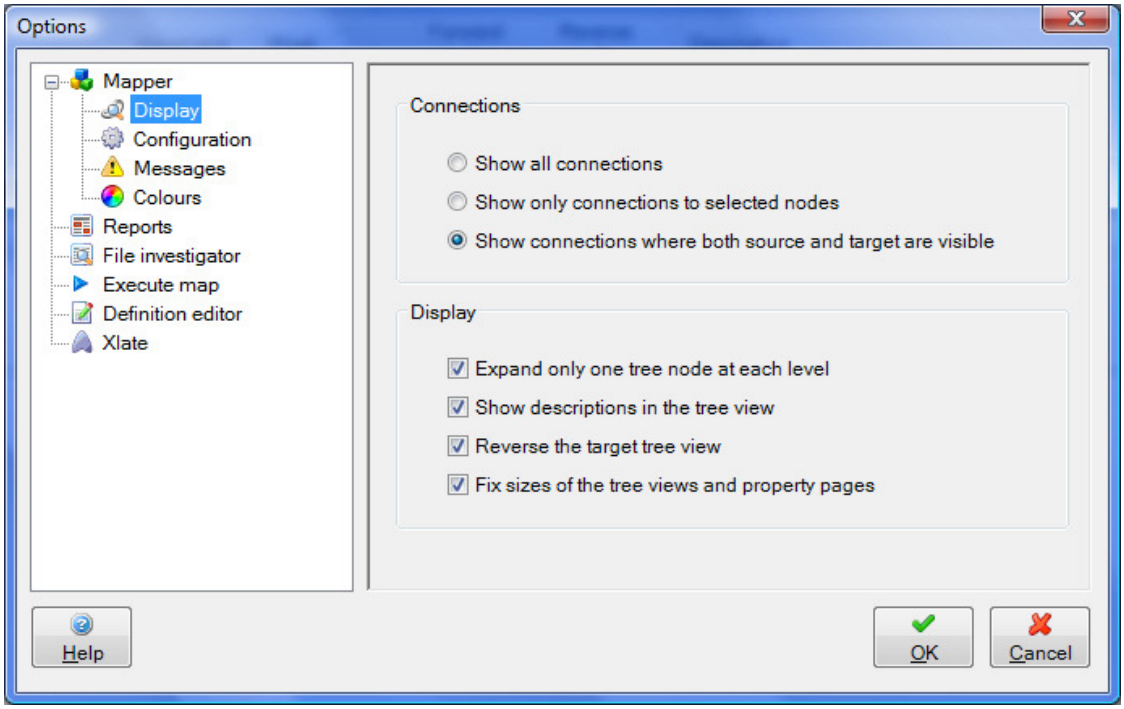


The dialog includes a number of property pages, each containing the properties for one component of the application (or part of it). Select the required property page by clicking on the tree view to the left of the dialog and the page will be displayed on the right. When you have finished editing the property pages, click **OK** to save your changes or **Cancel** to exit the dialog without saving.

The following pages contain details of each of the property pages available.

3.8.1 Property Page - Mapper – Display

This page is used to set display options for the Mapper view:



The **Connections** section allows you to choose which connections are shown in the Mapper view. Choose between:

- All connections
- Connections to selected tree nodes
- Connections to visible tree nodes

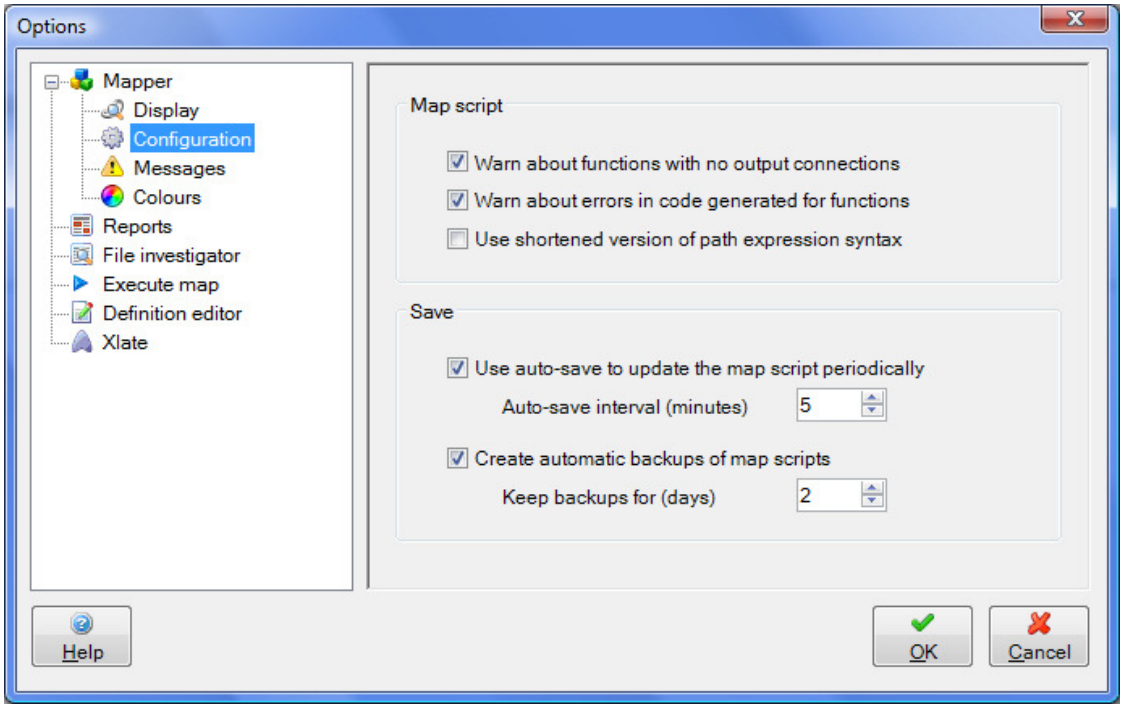
Typically you would choose to restrict the connections displayed when editing a complex mapping with a large number of connection lines.

The **Display** section offers a number of options affecting the way the Mapper view is displayed:

- Expand only one tree node at each level – prevents the source and target tree views from becoming hard to navigate by collapsing other nodes at the same level as the expanded node.
- Show descriptions in the tree view – means that the node descriptions in the definition files (if present) are displayed in the source and target tree views
- Reverse the target tree view – aids mapping by displaying a right-to-left tree view for the target definition.
- Fix sizes of the tree views and property pages – forces each part of the Mapper view to occupy a fixed proportion of the screen and disables the splitters so these proportions cannot be changed manually.
- Highlight map connections on mouse over – highlights the map connection when the cursor mouse is over the link. The style of the highlight can be defined by the user by selecting Colour node.

3.8.2 Property Page - Mapper – Configuration

This page is used to set script options for the Mapper view:



The **Map script** section includes these options:

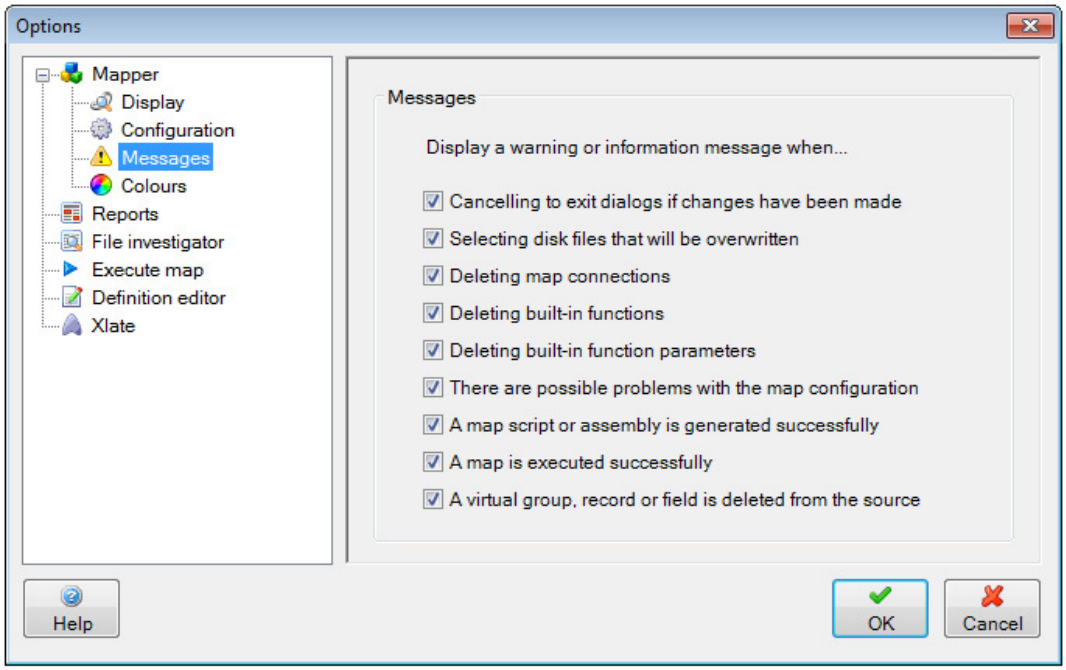
- Warn about functions with no output connections. Functions that are not connected to other functions or target nodes will not cause errors, but will not have any effect when mapping. Check this option to issue a warning when generating assemblies for affected maps.
- Warn about errors in code generated for functions. Functions are implemented by generating code snippets that are added to the map. If an error is detected when generating the code for a function it will not prevent the map assembly from being generated, but the map may not work as expected. Check this option to issue a warning when generating assemblies for affected maps.
- Use shortened version of path expression syntax. If selected this option causes the mapper to generate the map script using an abbreviated form of path expression. This makes the script less verbose, but also less easy to read. Use this option only if you are not likely to want to examine the map script.

The **Save** section allows you to configure auto-save and backup options:

- Choose auto-save to have Xe perform a background save of the map script at the specified interval. Turning auto-save on also ensures that the script is saved without prompting you if there are unsaved changes when the Mapper view is closed or Xe is shut down.
- Choose auto-backup to keep copies of old versions of the script. Auto-backup provides you with the ability to recover if work is lost in exceptional circumstances (power failure while the map is being saved, for instance) If this option is selected you can specify a retention period after which old backups will be removed.

3.8.3 Property Page - Mapper – Messages

This page is used to set warning message options for the Mapper view:



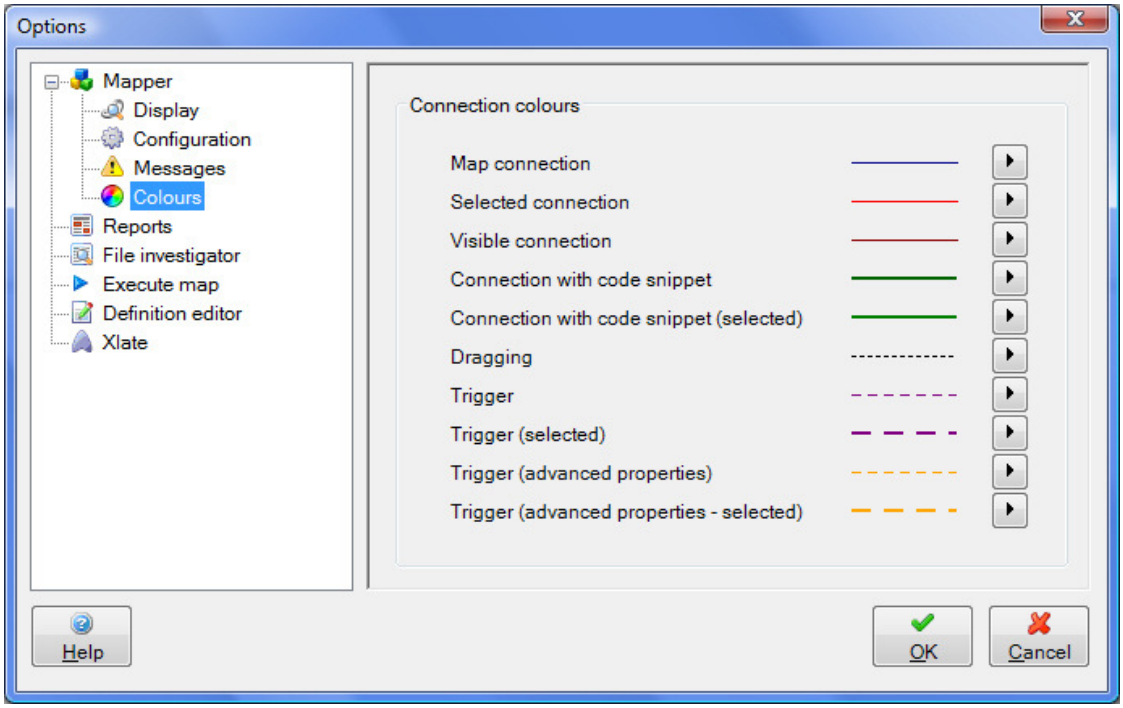
Warning or information messages are shown when certain actions are performed in the mapper. This section can be used to turn them on or off. By default, all of these warnings will be shown. It is recommended that those new to using the Mapper leave them switched on for a time, to get accustomed to the types of warnings that may be displayed and their implications.

- Cancelling changes – if this box is checked then Xe will warn you when cancelling changes made in a dialog.
- Selecting files that will be overwritten – if this box is checked then Xe will warn you when you choose files that it may overwrite (e.g. target files, log files.)
- Deleting map connections – if this box is checked then Xe will ask you to confirm deletion of map connections.
- Deleting functions – if this box is checked then Xe will ask you to confirm deletion of built-in functions.
- Deleting function parameters – if this box is checked then Xe will ask you to confirm deletion of built-in function parameters.
- Map configuration problems – if this box is checked then Xe will warn you if the map configuration is not complete, or if there are potential problems (e.g. the map execution will be slowed down by the logging options chosen.)
- Map script or assembly generated – if this box is checked then a message will be displayed each time you generate the script or assembly (you will always be notified if the operation fails.)
- Map executed successfully – if this box is checked then a message will be displayed each time you execute a map and it runs to completion (you will always be notified if the map fails.)
- A virtual group, record or field is deleted from the source – if this box is checked then a message will be displayed when group, record or field is deleted from the source.

•

3.8.4 Property Page - Mapper – Colours

This page is used to set map connection colouring options for the Mapper view:

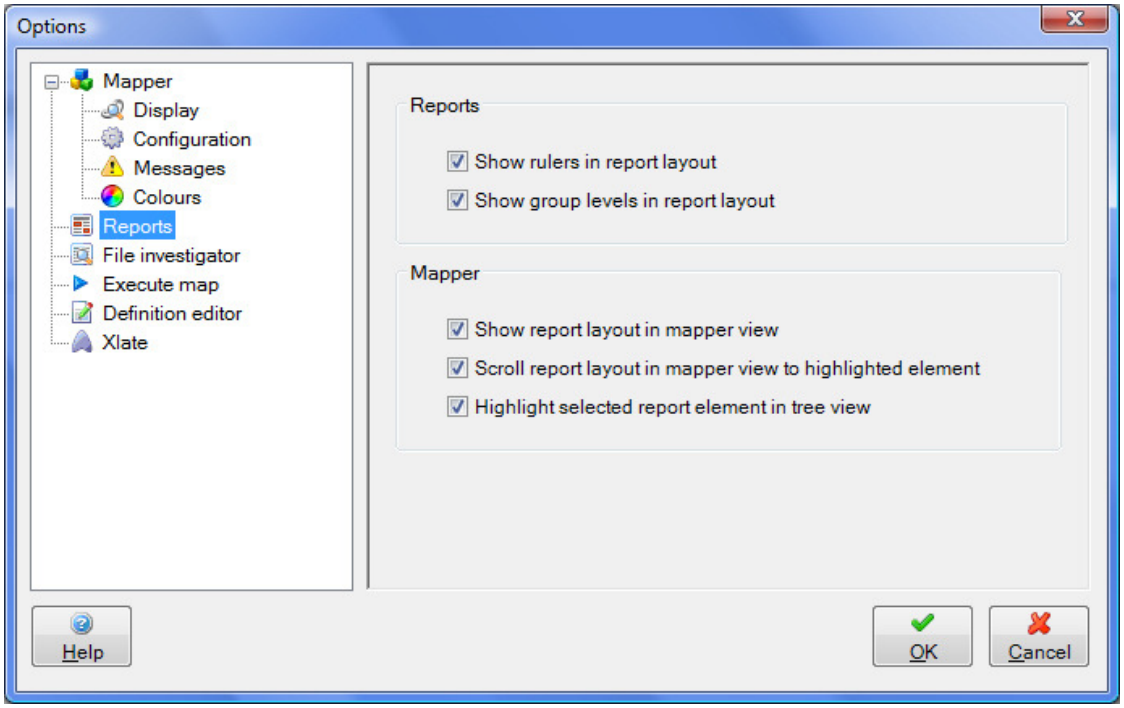


You can use this section to specify colours for various kinds of connection in the mapper view. Use the arrow buttons on the right to display a menu with these options:

- Colour – select this option to display a dialog that will allow you to change the line colour.
- Style – displays a sub-menu from which you can select the weight of the line.
- Dashed – displays a sub-menu from which you can select the dash pattern of the line.

3.8.5 Property Page - Reports

This page is used to set options for the Report designer and for reports in the Mapper view:



The **Reports** section includes these options:

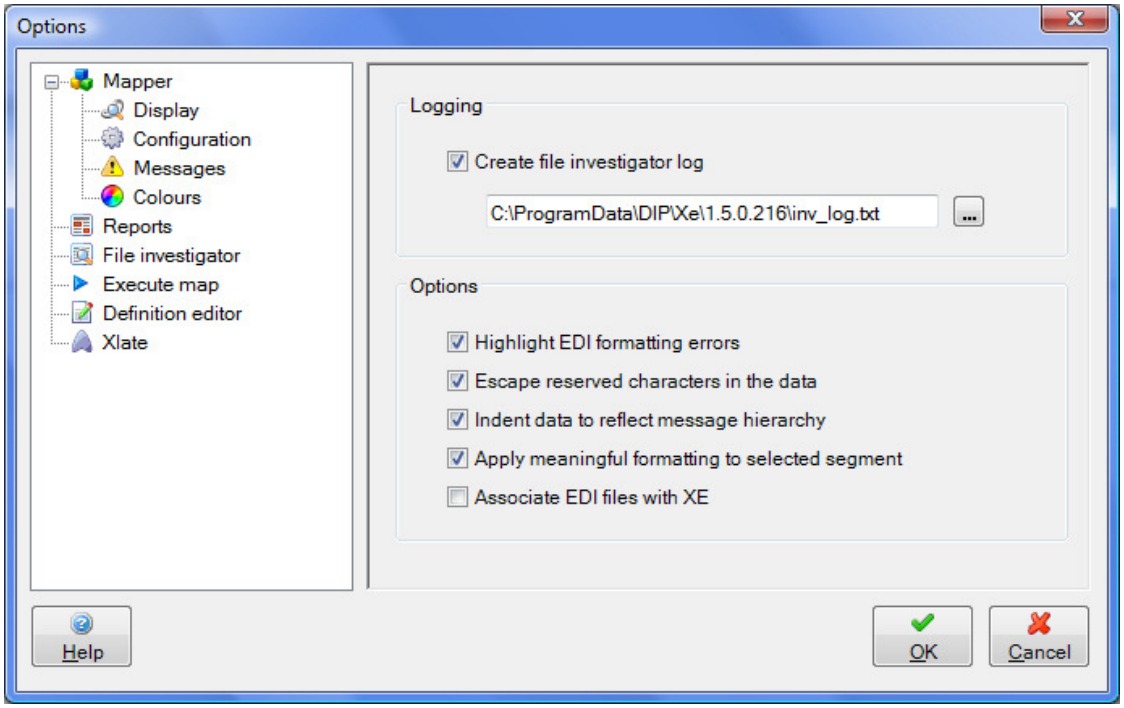
- Show rulers – select this option to display horizontal and vertical rulers in the layout section of the report designer view.
- Show group levels – select this option to display the hierarchical level beside a group start or end line in the layout section of the report designer view.

The **Mapper** section includes these options:

- Show report layout – select this option to display a read-only copy of report layout in a property tab in the mapper view.
- Scroll report layout – when the report layout is shown in the mapper view the matching report element is highlighted in the layout when a node is selected in the tree; select this option to scroll the report layout in the mapper view to ensure that the highlighted element is visible.
- Highlight selected element in tree – when the report layout is shown in the mapper view this option allows elements to be selected in the report layout and the matching node in the tree view to be highlighted and selected.

3.8.6 Property Page - File Investigator

This page is used to set options for the EDI file investigator:



The **Logging** section includes these options:

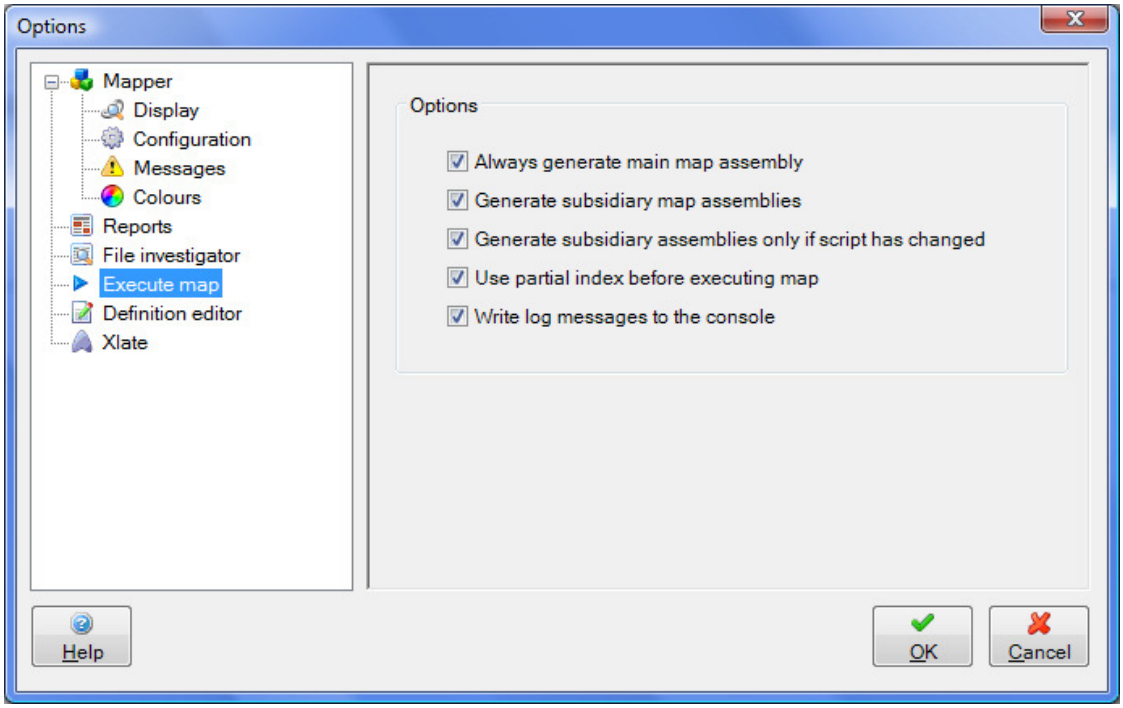
- Create log – select this option to indicate that the file investigator should create a log each time a file is loaded.
- Log file – if the log option is selected then a file path may be entered here, or click the button to browse for a file location.

The **Options** section includes these options:

- Highlight errors – select this option to indicate errors in red.
- Escape reserved characters – select this option to display escaped text without the escape character.
- Indent data – select this option to display segments indented according to the location in the message hierarchy.
- Format segment – select this option to display formatted information for selected known segments by interpreting the qualifiers.
- Associate EDI files with XE – select this option to 'grab' the EDI extension

3.8.7 Property Page - Execute Map

This page is used to set options affecting how maps are run from the GUI:

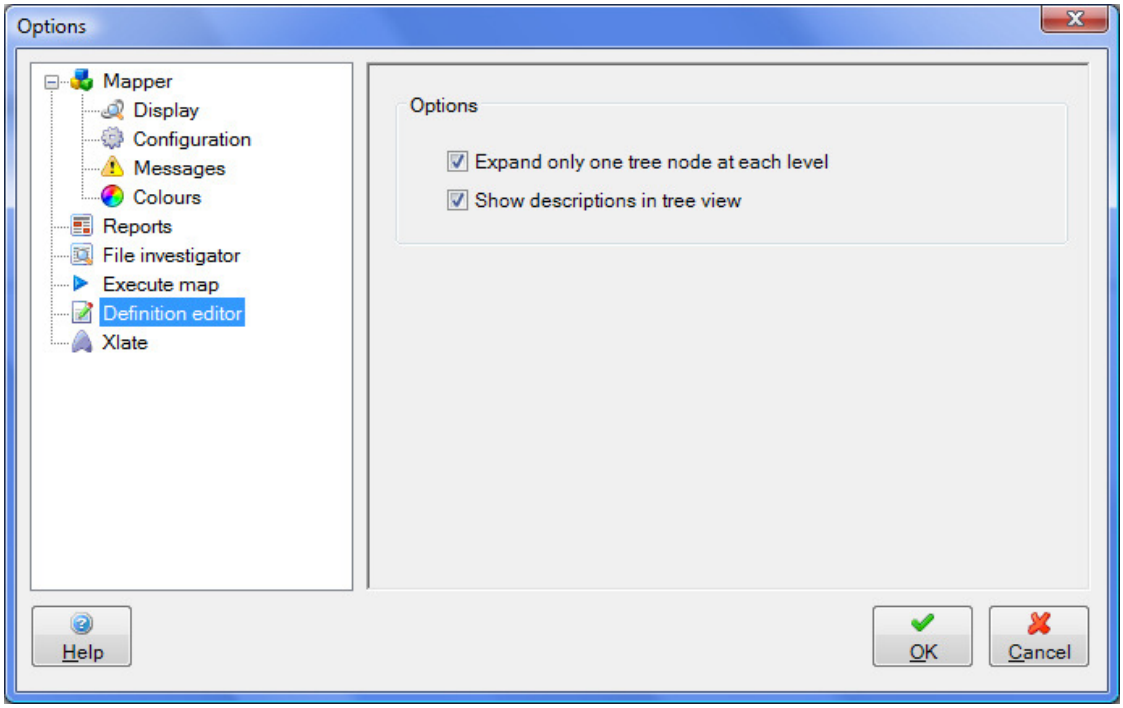


The **Options** section includes these options:

- Always generate assembly – select this option to generate the assembly before mapping in all cases (the alternative is that the assembly is generated only if the script has changed since it was last generated).
- Generate subsidiary assemblies – select this option to generate other assemblies required for the map (i.e. on other proc or override nodes) before mapping.
- Generate subsidiary assemblies if script changed – select this option to generate other assemblies only if they have changed since last generated.
- Use partial index – select this option to write a partial index for testing from the GUI (including only the transformation and associated entities for the current map).
- Log to console – select this option to write log messages to the GUI console during mapping.

3.8.8 Property Page - Definition Editor

This page is used to set options for the definition editor:

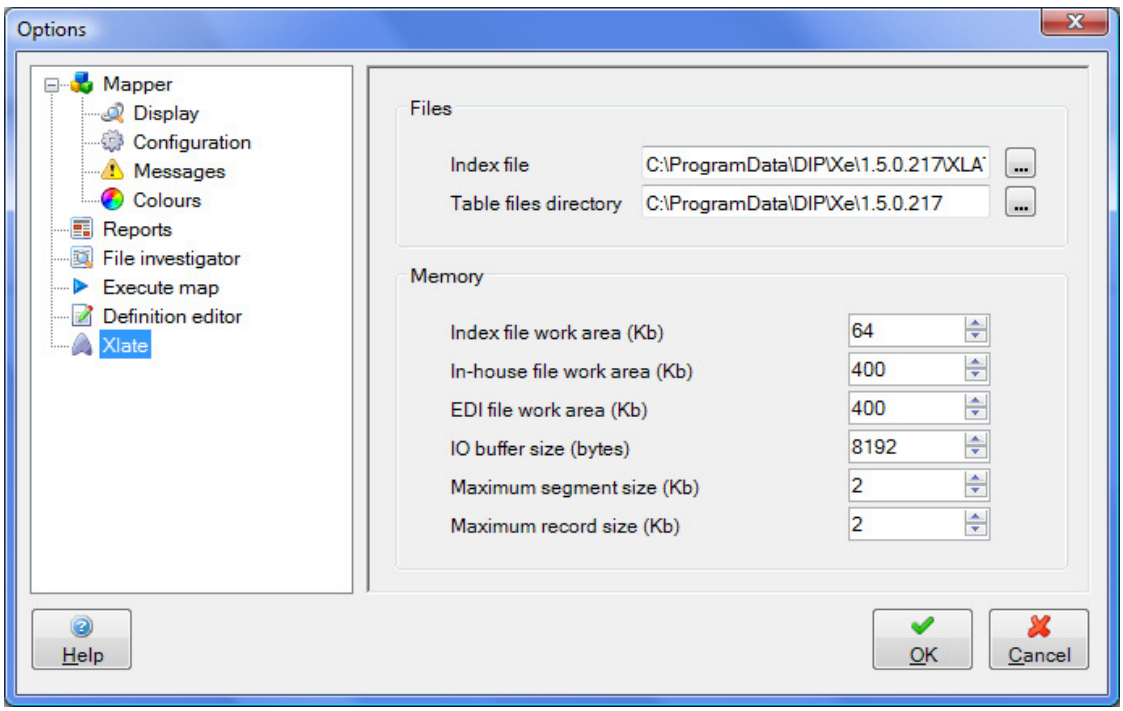


The **Options** section includes these options:

- Expand one tree node – select this option to allow only one tree node at each level to be expanded.
- Show descriptions – select this option to show descriptions alongside entity nodes in the tree view.

3.8.9 Property Page - Xlate

This page is used to set options for the XLATE mapping engine:



The **Files** section includes these options:

- Index file – specify the XLATE index file location.
- Table files directory – specify the directory where XLATE tables (EDF & HDF) can be found.

The **Memory** section includes these options:

- Index file work area – specify the memory set aside for the index.
- In-house file work area – specify the memory set aside for the in-house file definition.
- EDI file work area – specify the memory set aside for the EDI file definition.
- IO buffer size – specify the size of the read-write buffer.
- Maximum segment size – maximum permitted size of an EDI segment.
- Maximum record size – maximum permitted size of an in-house record.

4 The Index Explorer

We have seen in the overview above that the Index Explorer is used to configure Xe for mapping and to manage index entities such as trading partners, document definitions and maps. We have also seen that the component has two views, Tasks and Explorer, that allow the index to be manipulated in different ways. Later in this section these views will be described in full. First, however, we need to consider in more detail the function of the index and the information that it manages.

4.1 The Index

The Xe index is an XML file that contains the configuration information that the mapping engine needs to run. In the past, indexes were created by hand, or on-the-fly by the mapper. The index explorer component now allows index information to be viewed and edited directly using a simple interface.

The information provided by the index to the mapping engine allows it to identify a source document type, select one or more maps to perform on that document, identify the definition files required for each source and target document type, and format the output. An index that can be used in this way is referred to as a **runtime** index.

For this release of Xe, the index file format has been extended to allow it to store additional configuration information used only in the Index Explorer and Mapper. For instance, you can specify test files and logging options to use when running the map from the Mapper view. An index that contains this additional data is referred to as a **master** index.

The Index Explorer works only with master indexes created in Xe. There are several options available for importing data into a master index from an older version of Xe (which does not include the Index Explorer), and these are discussed later on.

4.1.1 Index structure

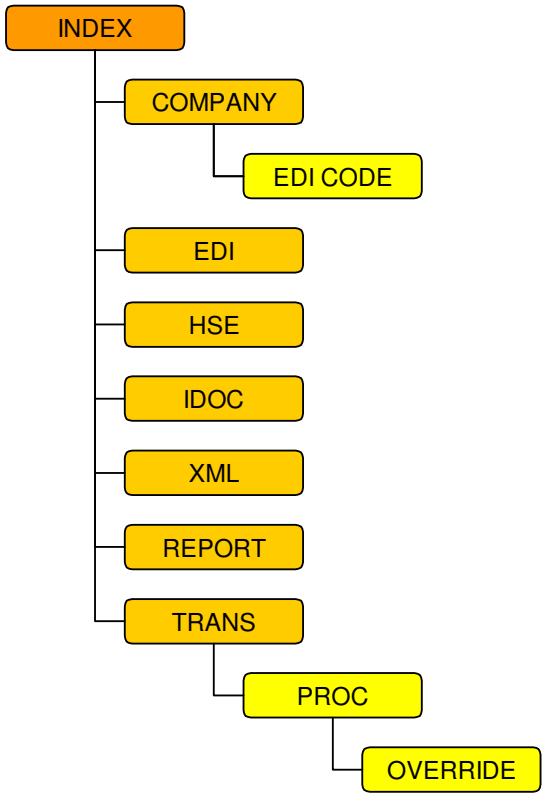
The index structure for Xe underpins the extended range of functionality in relation to XLATE. The index supports the following features:

- Supports any-to-any file format mapping
- Supports XML
- Supports multiple maps from the same source document

The index consists mainly of a series of collections of re-usable entities which can be combined to specify the actions that Xe takes on encountering a given source document or message type. Entities are made re-usable so that they do not have to be defined in more than one place in the index. For instance, a trading partner can be defined and can then be referred to from other entities in the index (which uses trading partners).

This technique is used frequently throughout the index and in many cases, the name (or ID) is used as the unique identifier for the entity. As a result, names are required to be unique not only within the given category of entity (e.g. trading partners) but across the whole index. You may want to adopt a coherent naming standard, in which the names chosen for index entities reflect the type of entity. For instance `"co.Ford"` for the trading partner (company) definition for Ford; and `"edi.DelforD96a"` for the EDI definition for a D69A DELFOR message.

The diagram below shows the principal index entities used to define Xe processing and how they relate to one another.



The index entities are described below (their use in the mapping process is described later on):

- The **Company** entity represents one trading partner and has one or more **EDI codes**. It can be referenced by **Trans** or **Override** nodes to specify that particular definitions and/or maps are used for messages to/from a given trading partner (note that a company may represent your organisation if defined with your EDI codes).
- The **EDI code** entity represents one EDI code, optionally including the qualifier and routing code.
- The **EDI** entity represents a definition of an EDI message, including an EDF filename, message name and EDI syntax.
- The **HSE** entity represents a definition of an in-house document type, including an HDF filename and HSE format (syntax).
- The **IDOC** entity represents a definition of a SAP IDOC, including an HDF filename and IDOC type.
- The **XML** entity represents a definition of an XML document type, including an XDF filename and the root node name and namespace.
- The **Report** entity represents a definition of an XE report, including an RDF filename.
- The **Trans** entity represents a transformation, that is, one map or related series of maps designed to transform a coherent set of one or more source messages or documents into a set of one or more target

documents. Most transformations are one-to-one, but there are a number of possible variants involving multiple maps such as when mapping a Tradacoms file that contains multiple related EDI messages, or when mapping an in-house file that to produce multiple EDI messages. Each map is represented by a **Proc** (process) in the index. A trans can be limited by sender and/or recipient (**Company** or **EDI code**).

- The **Proc** entity represents one map process, that is, a map from one source document or message to one target document or message. A process has a script associated with it, which can be opened, viewed and edited in the Mapper view.
- The **Override** entity represents one map process as well, but is distinct from a **Proc** in that it is associated with a specific sender and/or recipient (**Company** or **EDI code**). Like a **Proc** an **Override** has a script associated with it, which can be opened, viewed and edited in the Mapper view.

4.1.2 Index and mapping

The principal runtime purpose of the index is to inform the Xe mapping engine which definitions to use for source and target documents, and which maps to execute. When executing a simple map from EDI to in-house (with no trading partner-specific action) Xe uses the index in the following way:

- XE begins to read the source document, detecting the syntax and looking for message type information.
- Once the message type has been found the transformation in the index are searched for the one on which the source document reference (to an EDI entity) matches the message detected.
- The source document definition includes the definition file (EDF). The EDF is loaded and used to continue loading the EDI message.
- Once the message is loaded then the transformation is queried for the (map) process to execute.
- The process includes a reference to the target document (an HSE entity) which is used to identify and load the target definition file (HDF).
- The map process is executed by loading and running the map assembly identified by the proc entity in the index.
- When the mapping is completed the target document is written, using the loaded in-house definition model from the HDF.

There are variations on this according to the document types involved and the complexity of the mapping:

- If the source is XML then the document is loaded before the index is queried for transformations
- If there is are overrides specified in the index, the source is an in-house file and the source HDF specifies ORG and or DST details (originator and/or destination fields) then Xe needs to determine whether to use an override:
    - The values for ORG and/or DST are extracted from the source

- These are matched with each overrides on the process entity by comparing them with the company or EDI code referenced for sender and recipient on the override.
- If a match is found then the target document definition and map are those given on the override.
- If a match is not found then the target document definition and map are those given on the proc.

- If a sender and/or receiver company or EDI code is specified against a transformation and the source is an EDI message then the sender and receiver codes in the source are matched with those in the company/EDI code referenced by the transformation when choosing the transformation.
- If a source document matches more than one trans entity then all transformations are performed.
- If the source type is HSE and there are multiple processes defined against a transformation then each proc entity will have a trigger record defined against it. When each record is read from the source document Xe checks whether it is a trigger for a process. If it is then the process is used to identify the source and target definitions and the map in respect of the notional document represented by the trigger record. Data are loaded up to the next trigger record or the end of the input file, and the map is then performed.

4.1.3 Index and data files

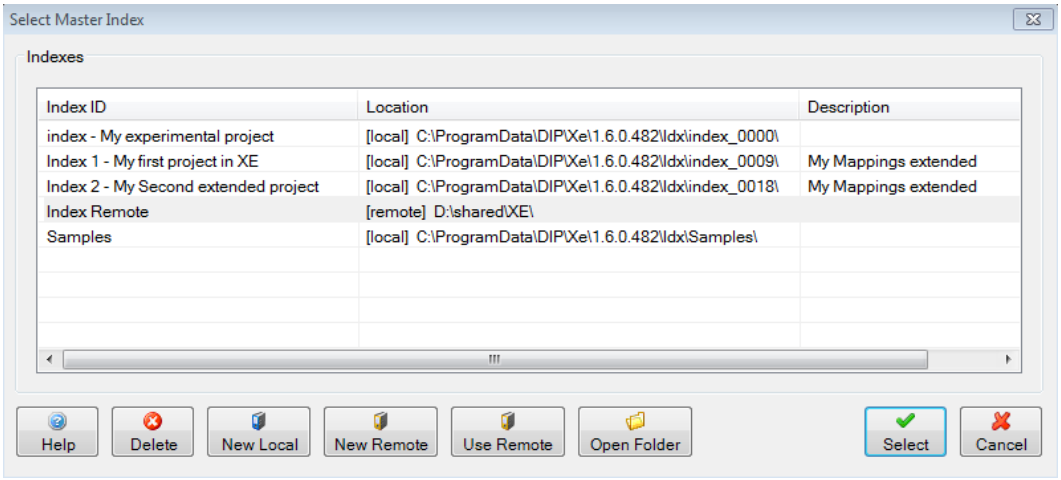
Before the introduction of the index editor, the files used by Xe (document definitions, lookup tables, map scripts and DLLs, test files) could be stored anywhere and referenced appropriately in the 'projects' that were used to describe single maps to the Mapper view. This has been changed now so that most data files required for Xe maps are managed by Xe and kept in a location within the Xe data directory. The directory structure used is as follows:

```
XeDataDir
  Idx
    index_0000
      trans_0000
        TestFiles
      trans_0001
        TestFiles
      &c
    index_0001
    &c
```

Any indexes created as 'local' indexes for a particular version of Xe will be stored in this structure. The index file will be saved to a directory named in the form `index_9999` created in Xe's `Idx` directory. As well as the index files, all definition files and lookup tables are stored here too. For each transformation a sub-directory is created in the form `trans_9999` and used to save the map script and assembly files. These sub-directories are also used when generating temporary indexes and config files for running the map from the GUI. Finally, each `trans` directory has a each `TestFiles` sub-directory into which test files may (optionally) be copied (when they are specified) and in which log and test output files are written by default. Test files (source, target and log) can still be referenced in any other location if required, but the advantage of using the test files directory is that the files there will be copied when performing an upgrade or backup.

4.2 Selecting An Index

The Index Explorer manages one index at a time. For many users one index is enough and you will specify all of your maps in it. However, multiple indexes can be created, if required, and you can also create indexes in 'remote' locations (that is, outside the described directory structure). The dialog used to manage indexes is show below:



This dialog is shown when starting Xe for the first time (unless you have upgraded from a previous version and created a single new index while doing so). Any indexes known to Xe are displayed in the list view.

The first column contains the index ID, which is the name you give the index in the Explorer. It pays to give the index a meaningful name, if you are likely to use more than one, so they can be distinguished easily in this list.

The second column contains the index location. This is included mainly for reference as you should not need to manipulate files there directly. The location is prefixed by either 'local' or 'remote' to indicate, respectively, that the index was created the `Idx` directory of the current instance of Xe, or in some other location.

The third column shows a description, which is also entered as a property of the index in the Explorer.

To open an index in the Explorer, highlight the required item in the list and click the **Select** button.

To create a new index locally click the **New Local** button.

To create a new index in another location that you specify click the **New Remote** button. You will be prompted to browse for a location where the index will be created. The complete index directory structure described above will be created in this location, so you should choose a directory with no other content.

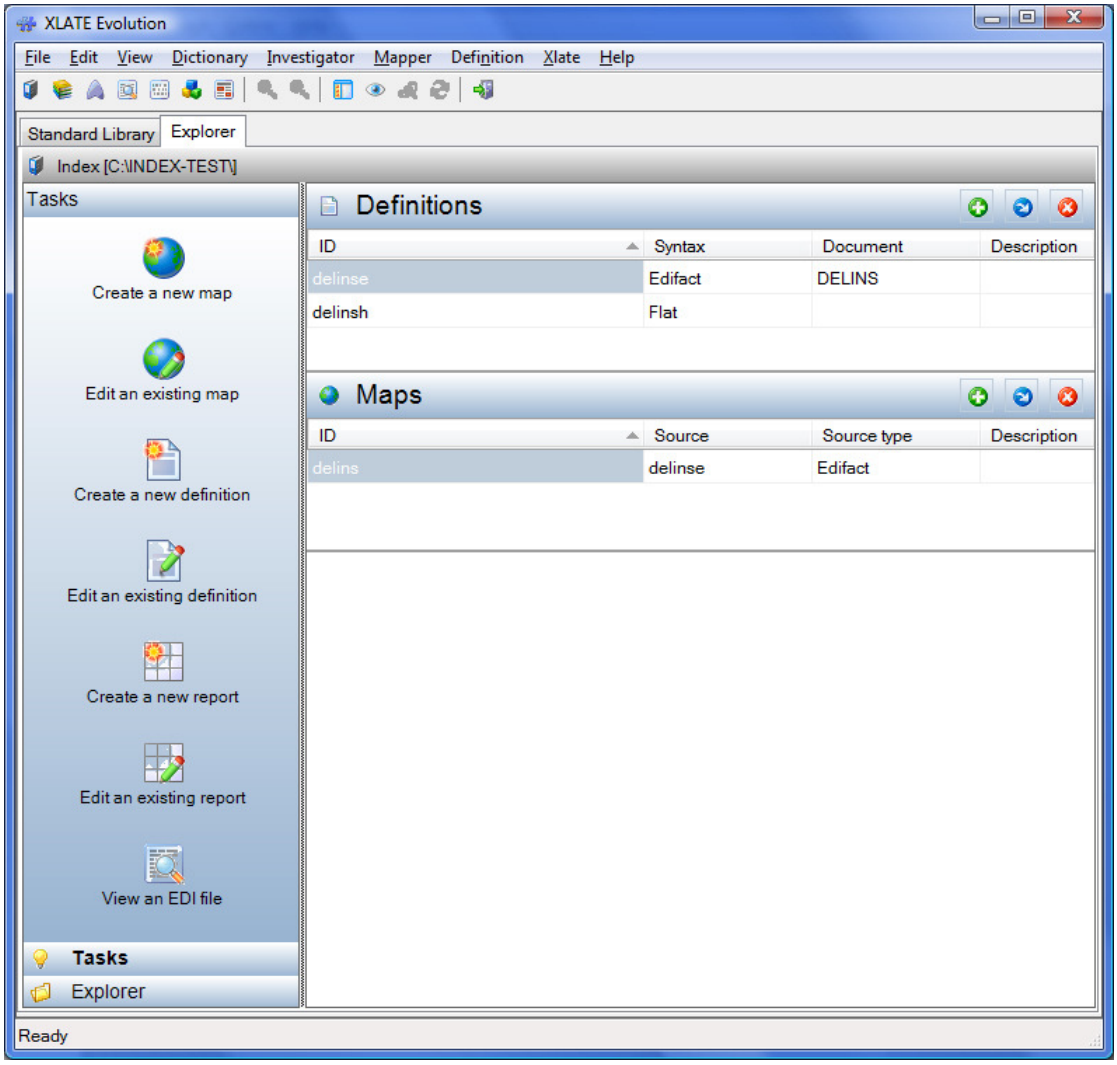
To locate an existing remote index click the **Use Remote** button. You will be prompted to browse for an existing index file. Xe will then establish a remote reference to that file.

To delete an index click the **Delete** button. You will be prompted to confirm that you want to delete the index. Remember that if you do delete it, all of the configuration information saved there will be lost. If you select a 'remote' index and choose delete then, by default, Xe just removes its reference to the file;

however, you will be prompted to delete the index and associated files instead. Use **Open Folder** button to open a location folder with project files from currently selected index.

4.3 The Tasks View

The Tasks view of the Index Explorer is show below.

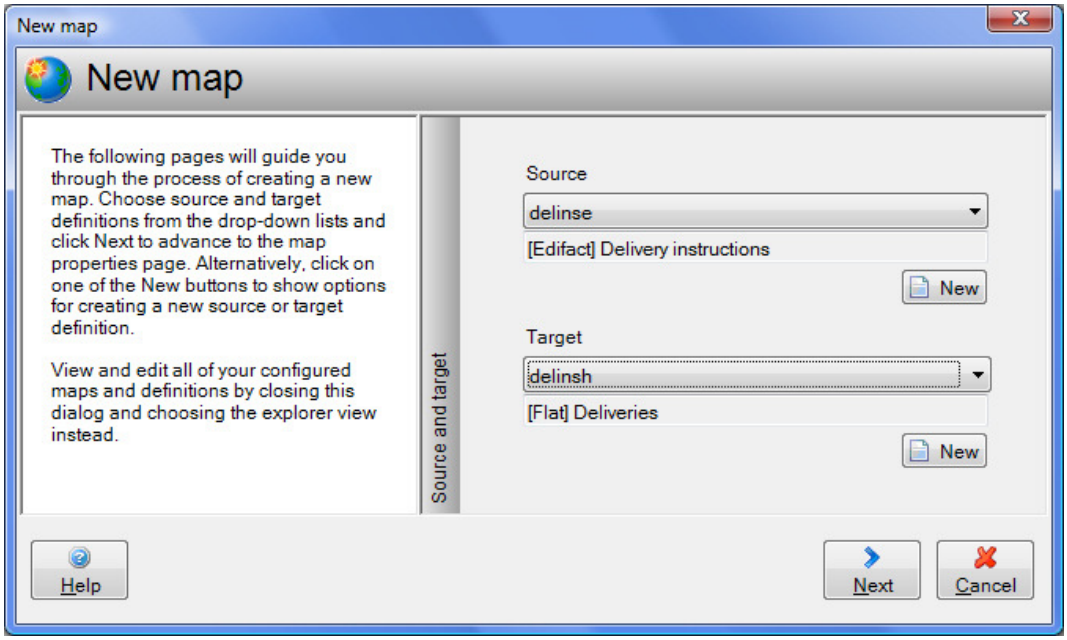


The purpose of the Tasks view is to provide access to various wizards that guide you through commonly used index activities. The tasks bar on the left includes a number of buttons, each of which launches one task wizard. The lists on the right hand side of the Explorer show, respectively, all of the definitions and maps defined in the current index.

4.3.1 New map wizard

4.3.1.1 New map – source and target

The new map wizard is used to create a simple map with one source and one target definition. More complex maps can be created by switching to the Explorer view. The first page of the new map wizard is used to select the source and target document definitions for the map.



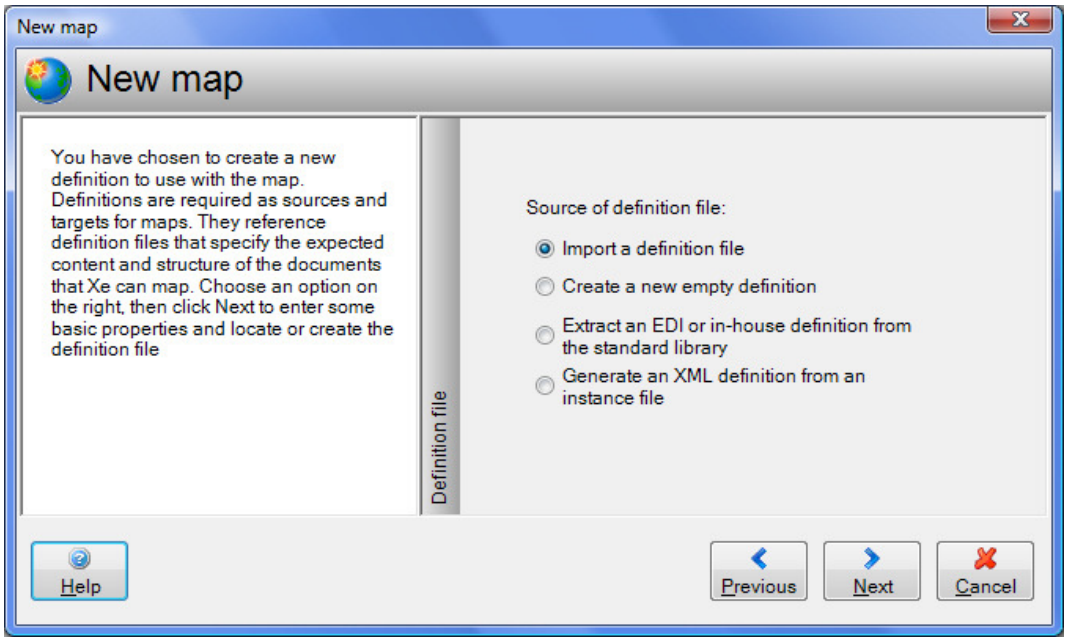
Use the drop downs to pick the source and target from lists of document definitions already specified in the index.

Alternatively, click one of the **New** buttons to create a new definition to use for the source or target of the map.

When you are satisfied with your selections, click the **Next** button to continue.

4.3.1.2 New map – definition file

This page of the new map wizard is shown when you choose to create a new source or target definition to use for the source or target of the map.



Select one option for the source then click the **Next** button to continue. Alternatively, click the **Previous** button to return to the source and target page ("New map – source and target") without creating a new definition.

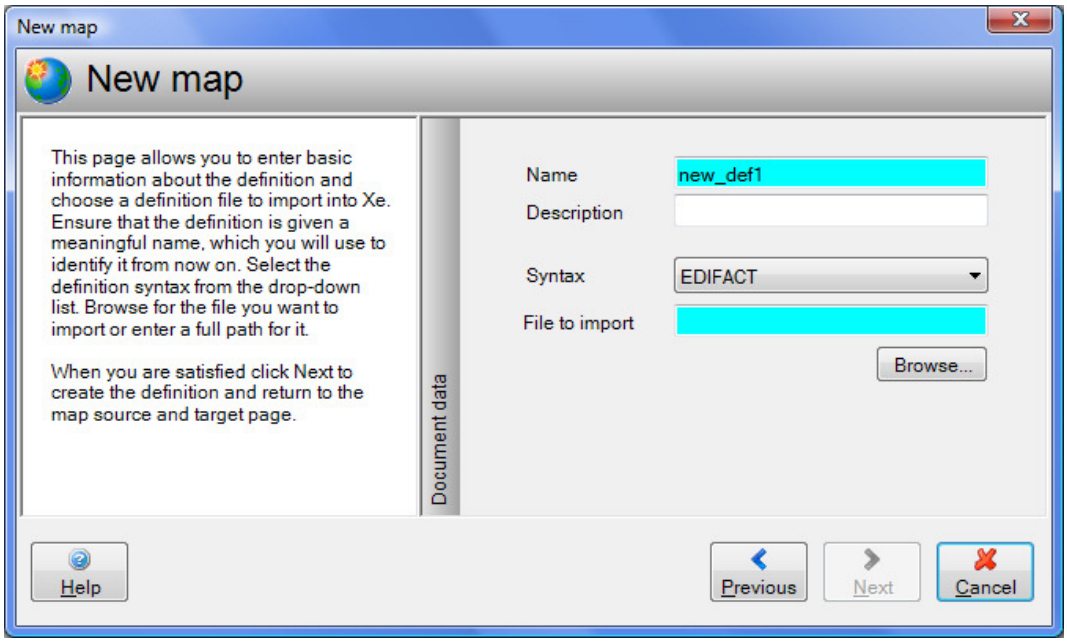
The options on this page are as follows:

- Import a definition file – import an existing definition file to use with the new definition (see "New map – import definition file").

- Create a new empty definition – create a blank definition using basic information you specify on the next page (see “New map – new definition file”).
- Extract an EDI or in-house definition from the standard library – select a dictionary and message from the data dictionary and extract an EDI definition or simple in-house definition (see “New map – create definition file from dictionary”).
- Generate an XML definition from an instance file – select an XML file and create a definition based on it (see “New map – create definition file from XML instance”).

4.3.1.3 New map – import definition file

This page of the new map wizard is shown when you choose to import a definition file to use with a new document definition you are creating.



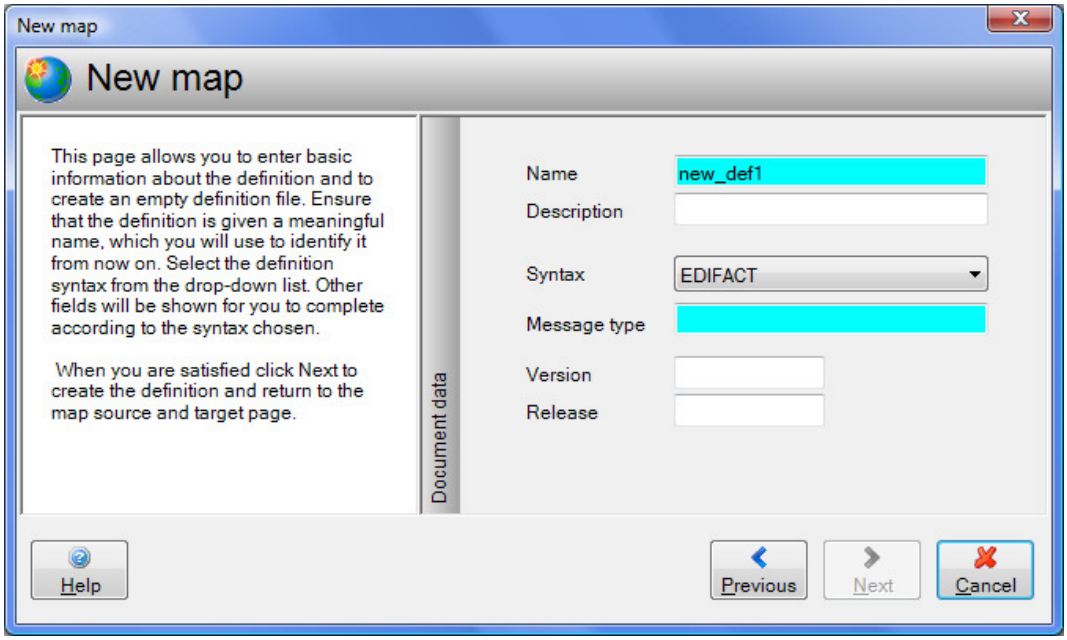
Enter a name for the new definition in the **Name** text box. The name will be used to refer to the definition in other index entities including the new map, so it must be unique within the index. Consider giving the definition a meaningful name so you can identify its purpose in the future. You may optionally enter a **Description** that provides more information about the definition.

Choose a **Syntax** from the drop-down list to specify the document type. Enter the full path of a suitable definition file to be imported, or click **Browse** to browse for a file. The **Next** button will not be enabled until a valid file and definition name are specified.

When you are satisfied, click the **Next** button to continue, or click the **Previous** button to return to the new definition page (see “New map – definition file”).

4.3.1.4 New map – new definition file

This page of the new map wizard is shown when you choose to create a new empty definition file to use with a new document definition you are creating.



Enter a name for the new definition in the **Name** text box. The name will be used to refer to the definition in other index entities including the new map, so it must be unique within the index. Consider giving the definition a meaningful name so you can identify its purpose in the future. You may optionally enter a **Description** that provides more information about the definition.

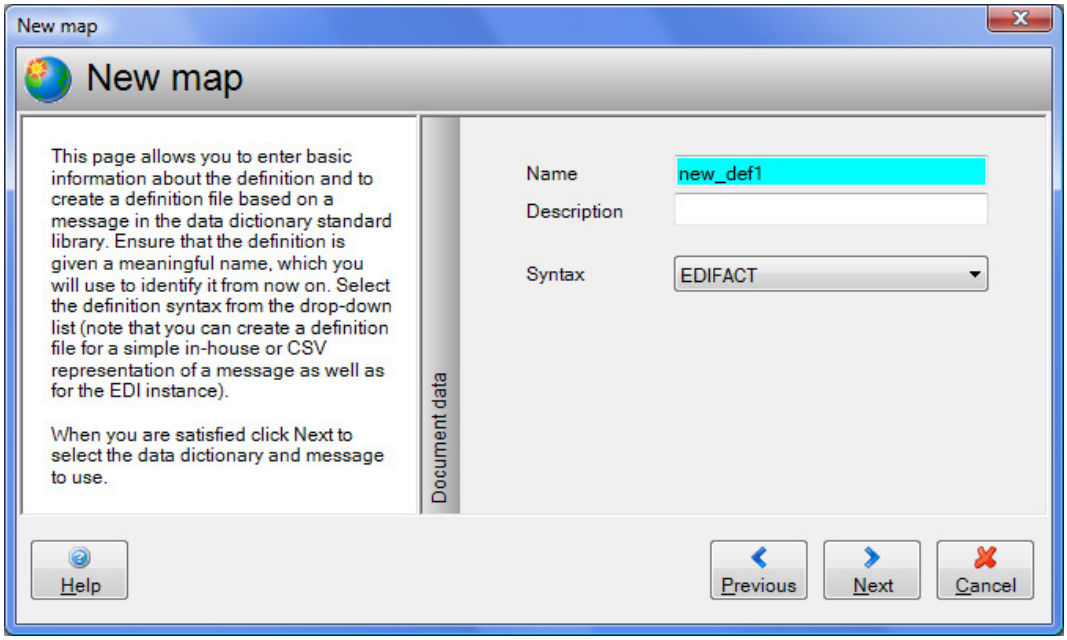
Choose a **Syntax** from the drop-down list to specify the document type. The remaining data you need to enter depends upon the syntax selected:

- For EDI syntaxes (except VDA) you are required to enter a message (or transaction set) name (such as DELFOR, 850 &c), and may optionally enter a version and release.
- For VDA you are required to enter a message name (such as 4905, 4913 &c), and may optionally enter a version.
- For IDOC and in-house definitions (flat, CSV and TRA), select a record format from the drop-down list.
- For XML and report definitions no further data is required.

When you are satisfied, click the **Next** button to continue, or click the **Previous** button to return to the new definition page (see “New map – definition file”).

4.3.1.5 New map – create definition file from dictionary

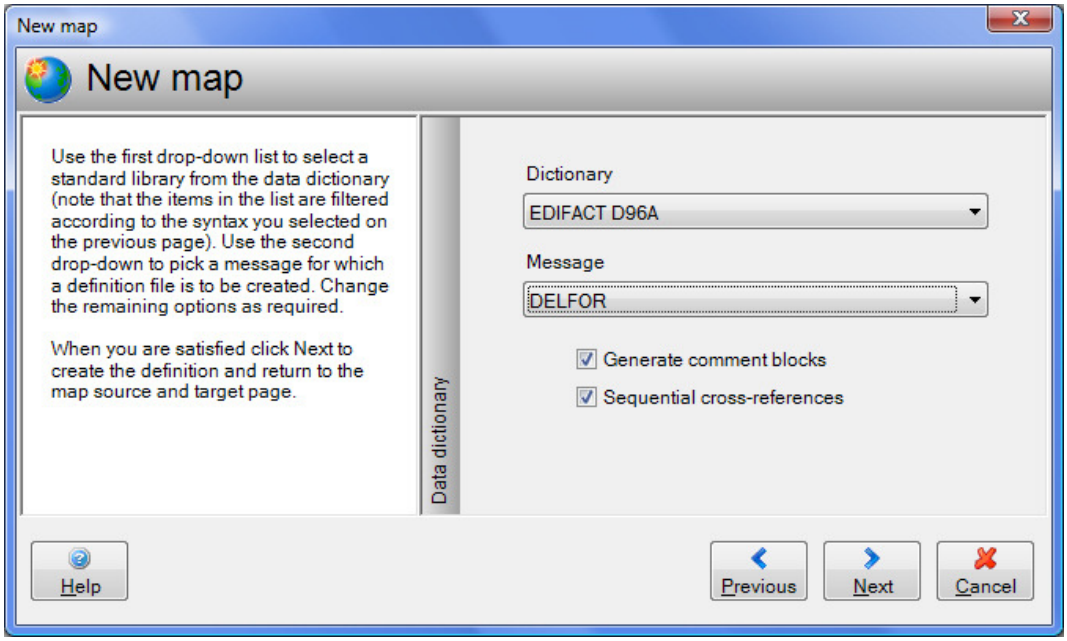
This page of the new map wizard is shown when you choose to create a new definition file from the data dictionary to use with a new document definition you are creating.



Enter a name for the new definition in the **Name** text box. The name will be used to refer to the definition in other index entities including the new map, so it must be unique within the index. Consider giving the definition a meaningful name so you can identify its purpose in the future. You may optionally enter a **Description** that provides more information about the definition.

Choose a **Syntax** from the drop-down list to specify the document type.

When you are satisfied, click the **Next** button to continue to the next page which allows you to select a dictionary and message. Alternatively, click the **Previous** button to return to the new definition page (see “New map – definition file”).



This page allows you to select a **Dictionary** and **Message** from the drop-down lists (the messages available depend on the dictionary selected).

The **Generate comment blocks** option allows you to specify whether the generated definition will contain comments for each record in a preceding block.

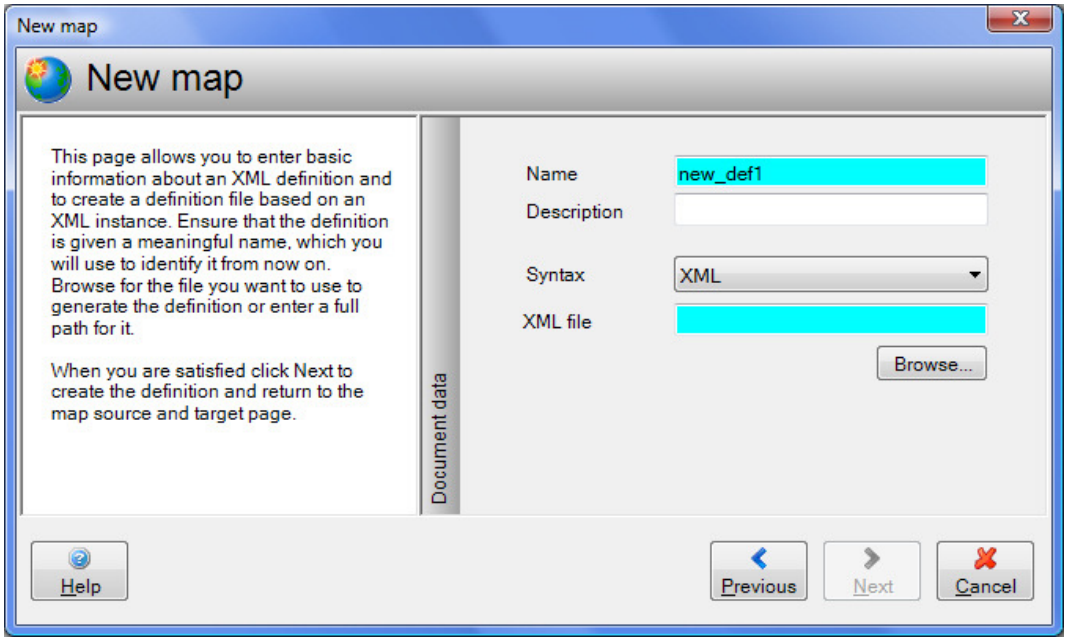
The **Sequential cross-references** option allows you to specify whether the element cross-references generated are strictly sequential (for example; RFF00510, RFF00520). The alternative is that they are based on the

occurrence number of the segment (for example; RFF020010, RFF020020 for the second instance of RFF in the definition).

When you are satisfied, click the **Next** button to continue, or click the **Previous** button to return to the previous page.

4.3.1.6 New map – create definition file from XML instance

This page of the new map wizard is shown when you choose to create a new XML definition file to use with a new document definition you are creating, based on an XML instance.



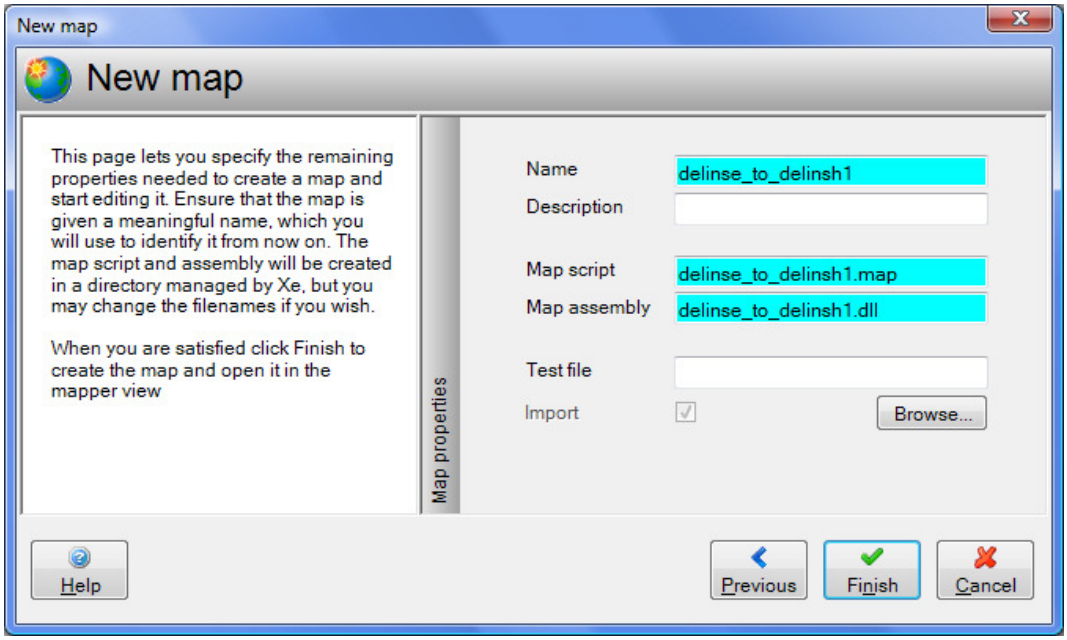
Enter a name for the new definition in the **Name** text box. The name will be used to refer to the definition in other index entities including the new map, so it must be unique within the index. Consider giving the definition a meaningful name so you can identify its purpose in the future. You may optionally enter a **Description** that provides more information about the definition.

The **Syntax** drop-down list can only have the value “XML”. Enter the full path of a suitable XML file upon which the definition will be based, or click **Browse** to browse for a file. The **Next** button will not be enabled until a valid file and definition name are specified.

When you are satisfied, click the **Next** button to continue, or click the **Previous** button to return to the new definition page (see “New map – definition file”).

4.3.1.7 New map – properties

This is the final page of the new map wizard, which allows you to specify the remaining data required to created the map.



Enter a name for the new map in the **Name** text box. The name will be used to refer to the map in the index, so it must be unique within the index. Consider giving the map a meaningful name so you can identify its purpose in the future. You may optionally enter a **Description** that provides more information about the map.

By default the **Map script** and **Map assembly** filenames are derived from the map name, though you may change them if required. As these files will be managed by Xe in a location that it chooses (alongside the index) a full path is not required.

You may optionally enter the full path of a **Test file** (source file) to use when running the map from the Xe mapper view. If you select a test file you may also choose the **Import** option, which means that Xe will take a copy of the file and keep it locally to the index (this means it will be copied during backups and upgrades when new versions are installed).

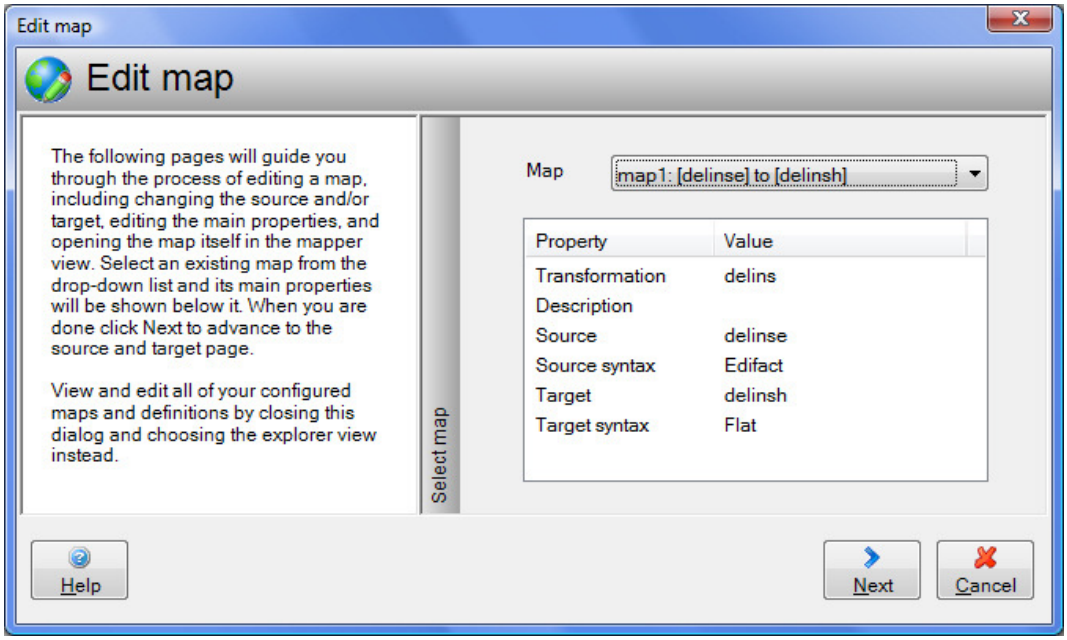
The **Finish** button is enabled only when the mandatory fields have been completed with valid values. If a test file path is entered, the **Finish** button will not be enabled unless the file exists.

When you are satisfied, click the **Finish** button to create the map and open it in the mapper view. Alternatively, click the **Previous** button to return to the new definition page (see “New map – definition file”).

4.3.2 Edit map wizard

4.3.2.1 Edit map – select map

The edit map wizard is used to edit a single map process (one source and one target definition) already defined in the index. Manipulation of more complex maps can be performed by switching to the Explorer view. The first page of the edit map wizard is used to select the map to be edited.

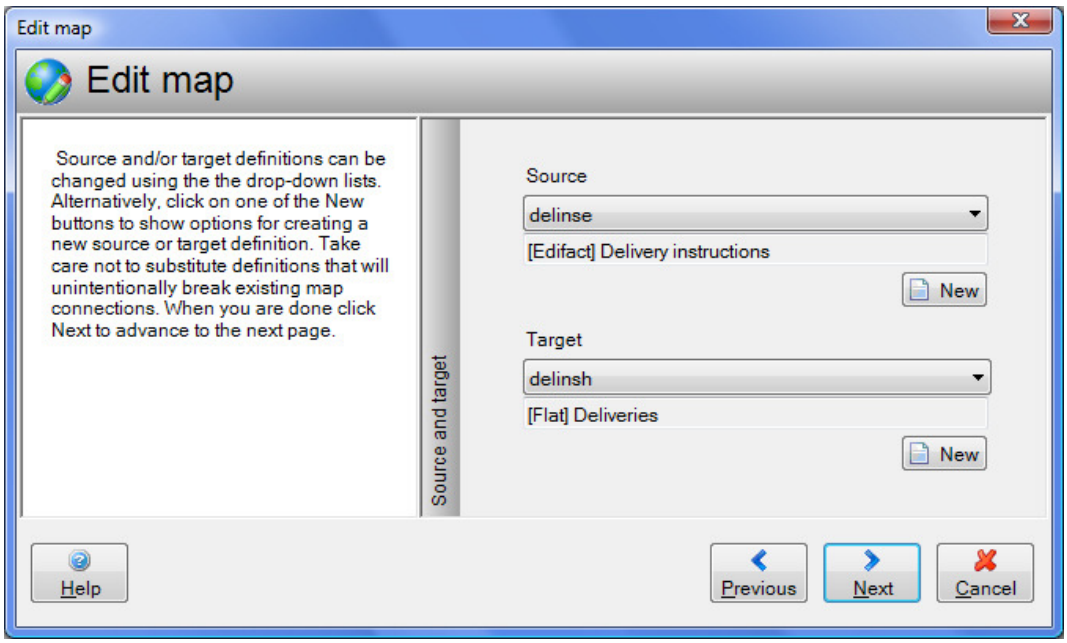


Use the **Map** drop-down to select the map to be edited. Each time you change the selection, the properties of the map will be shown in the list. Note that these properties include the transformation name. Remember that in the Xe index, a transformation may represent more than one map process to be carried out on a file. If complex maps with more than one process have been defined (in the explorer view) then each process appears in the drop-down.

When you are satisfied with the selection, click the **Next** button to continue.

4.3.2.2 Edit map – source and target

This page of the edit map wizard is used to select the source and target document definitions for the map.



Use the drop downs to pick the source and target from lists of document definitions already specified in the index.

Alternatively, click one of the **New** buttons to create a new definition to use for the source or target of the map.

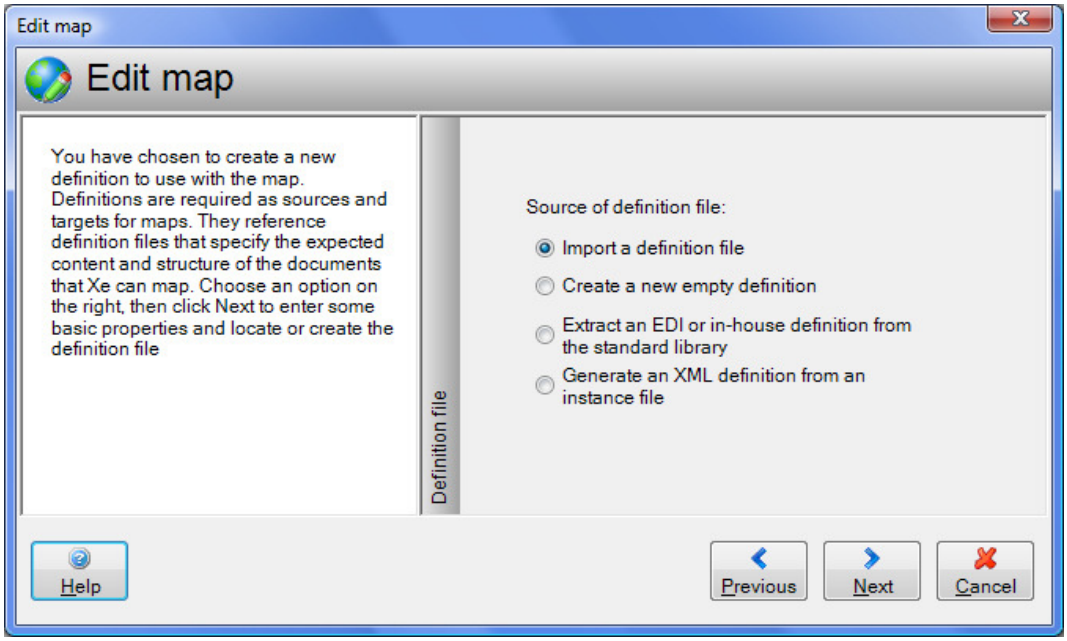
You should take care when changing the source and/or target for an existing map. Any map connections that have been added in the mapper view will

depend on the structure and names used in the definition, and so may be lost if the definitions are changed.

When you are satisfied with your selections, click the **Next** button to continue.

4.3.2.3 Edit map – definition file

This page of the edit map wizard is shown when you choose to create a new source or target definition to use for the source or target of the map.



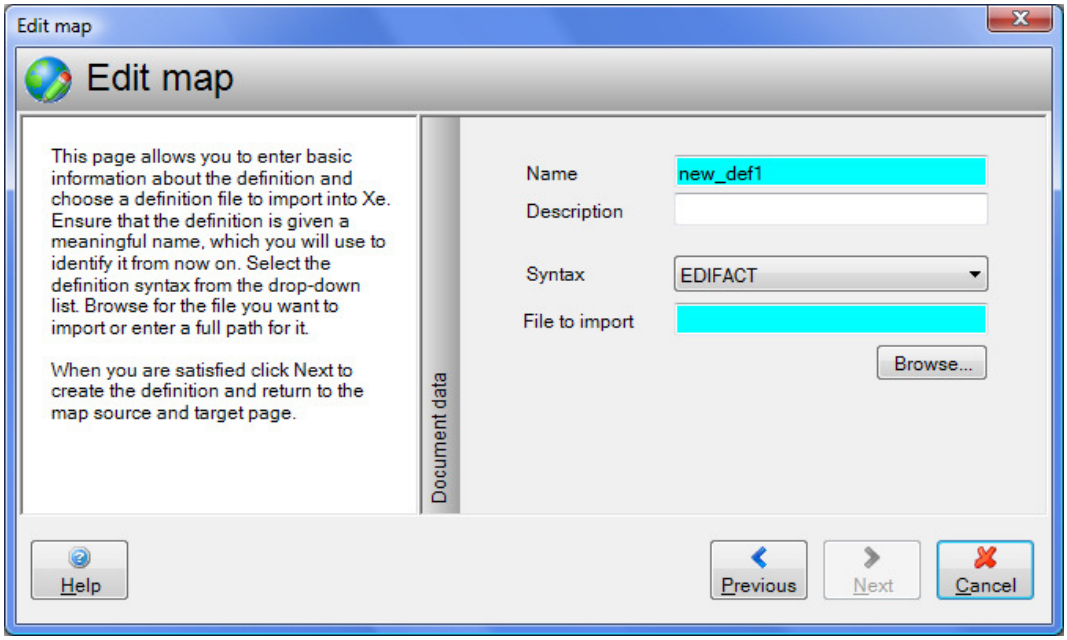
Select one option for the source then click the **Next** button to continue. Alternatively, click the **Previous** button to return to the source and target page ("Edit map – source and target") without creating a new definition.

The options on this page are as follows:

- Import a definition file – import an existing definition file to use with the new definition (see "Edit map – import definition file").
- Create a new empty definition – create a blank definition using basic information you specify on the next page (see "Edit map – new definition file").
- Extract an EDI or in-house definition from the standard library – select a dictionary and message from the data dictionary and extract an EDI definition or simple in-house definition (see "Edit map – create definition file from dictionary").
- Generate an XML definition from an instance file – select an XML file and create a definition based on it (see "Edit map – create definition file from XML file").

4.3.2.4 Edit map – import definition file

This page of the edit map wizard is shown when you choose to import a definition file to use with a new document definition you are creating.



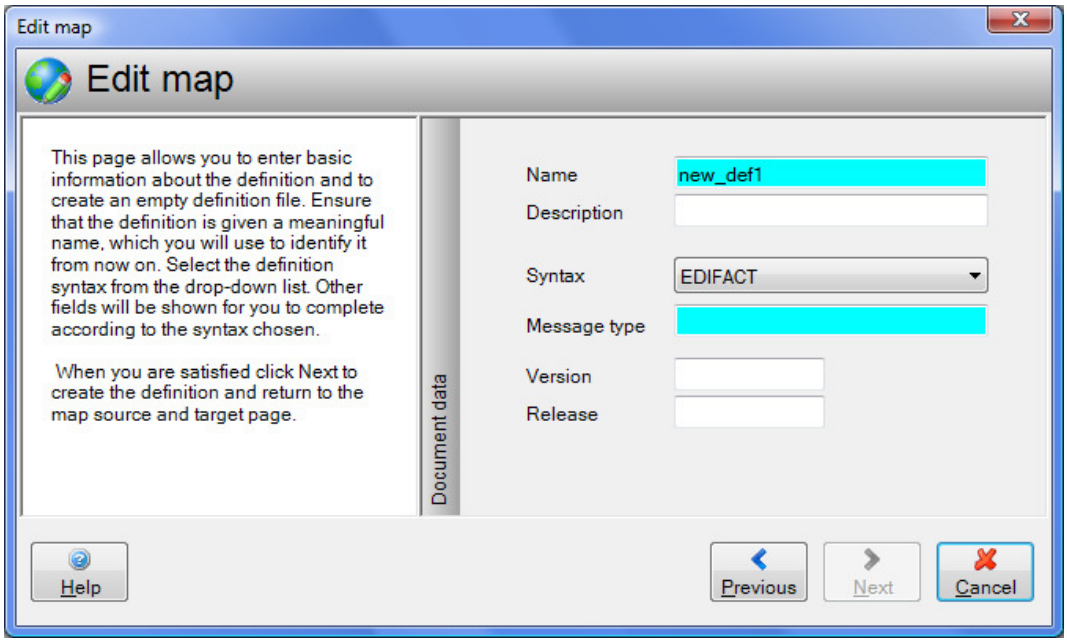
Enter a name for the new definition in the **Name** text box. The name will be used to refer to the definition in other index entities including the map, so it must be unique within the index. Consider giving the definition a meaningful name so you can identify its purpose in the future. You may optionally enter a **Description** that provides more information about the definition.

Choose a **Syntax** from the drop-down list to specify the document type. Enter the full path of a suitable definition file to be imported, or click **Browse** to browse for a file. The **Next** button will not be enabled until a valid file and definition name are specified.

When you are satisfied, click the **Next** button to continue, or click the **Previous** button to return to the new definition page (see “Edit map – definition file”).

4.3.2.5 Edit map – new definition file

This page of the edit map wizard is shown when you choose to create a new empty definition file to use with a new document definition you are creating.



Enter a name for the new definition in the **Name** text box. The name will be used to refer to the definition in other index entities including the map, so it must

be unique within the index. Consider giving the definition a meaningful name so you can identify its purpose in the future. You may optionally enter a **Description** that provides more information about the definition.

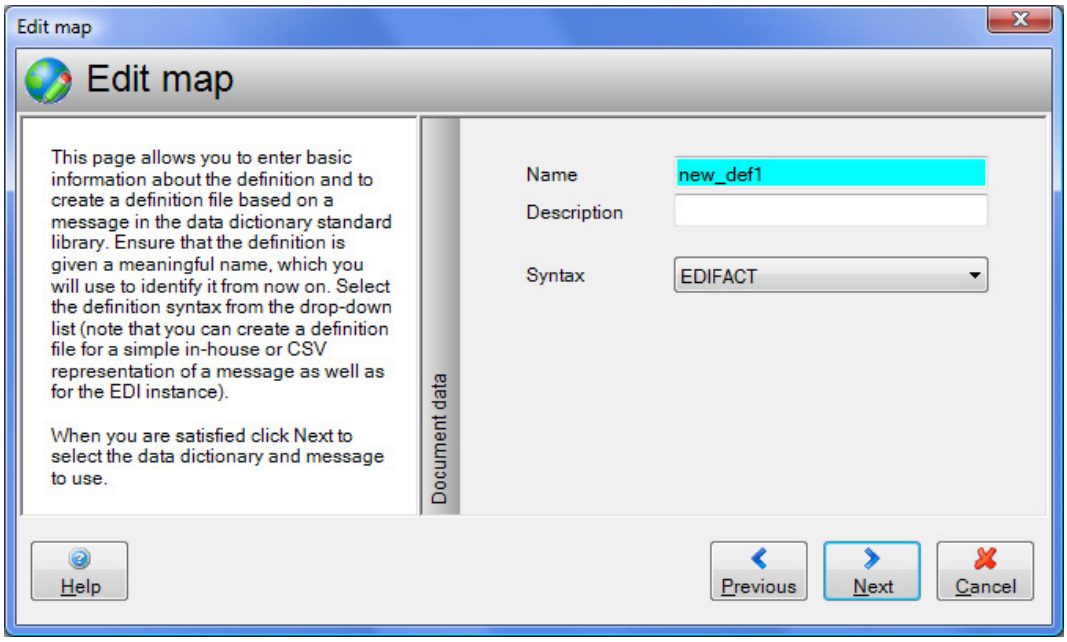
Choose a **Syntax** from the drop-down list to specify the document type. The remaining data you need to enter depends upon the syntax selected:

- For EDI syntaxes (except VDA) you are required to enter a message (or transaction set) name (such as DELFOR, 850 &c), and may optionally enter a version and release.
- For VDA you are required to enter a message name (such as 4905, 4913 &c), and may optionally enter a version.
- For IDOC and in-house definitions (flat, CSV and TRA), select a record format from the drop-down list.
- For XML and report definitions no further data is required.

When you are satisfied, click the **Next** button to continue, or click the **Previous** button to return to the new definition page (see “Edit map – definition file”).

4.3.2.6 Edit map – create definition file from dictionary

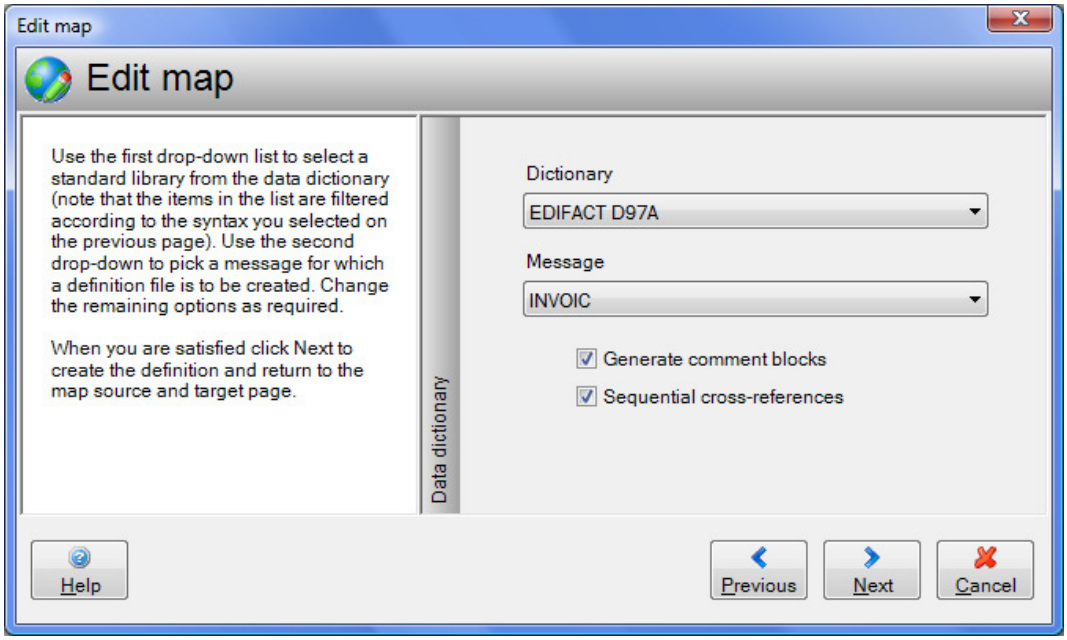
This page of the edit map wizard is shown when you choose to create a new definition file from the data dictionary to use with a new document definition you are creating.



Enter a name for the new definition in the **Name** text box. The name will be used to refer to the definition in other index entities including the map, so it must be unique within the index. Consider giving the definition a meaningful name so you can identify its purpose in the future. You may optionally enter a **Description** that provides more information about the definition.

Choose a **Syntax** from the drop-down list to specify the document type.

When you are satisfied, click the **Next** button to continue to the next page which allows you to select a dictionary and message. Alternatively, click the **Previous** button to return to the new definition page (see “Edit map – definition file”).



This page allows you to select a **Dictionary** and **Message** from the drop-down lists (the messages available depend on the dictionary selected).

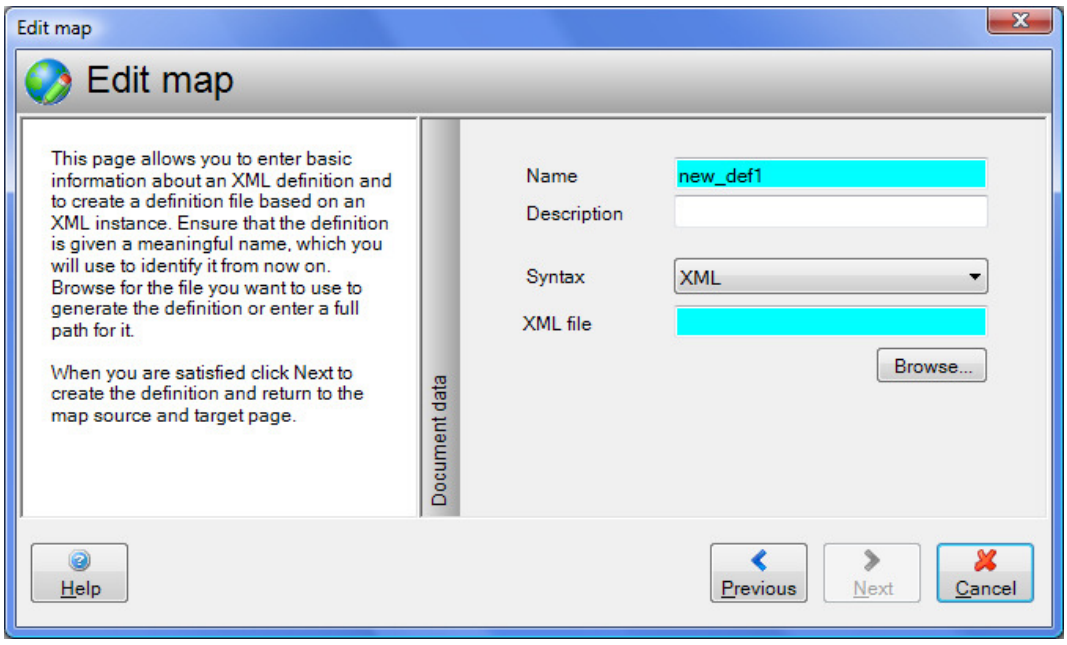
The **Generate comment blocks** option allows you to specify whether the generated definition will contain comments for each record in a preceding block.

The **Sequential cross-references** option allows you to specify whether the element cross-references generated are strictly sequential (for example; RFF00510, RFF00520). The alternative is that they are based on the occurrence number of the segment (for example; RFF020010, RFF020020 for the second instance of RFF in the definition).

When you are satisfied, click the **Next** button to continue, or click the **Previous** button to return to the previous page.

4.3.2.7 Edit map – create definition file from XML file

This page of the edit map wizard is shown when you choose to create a new XML definition file to use with a new document definition you are creating, based on an XML instance.



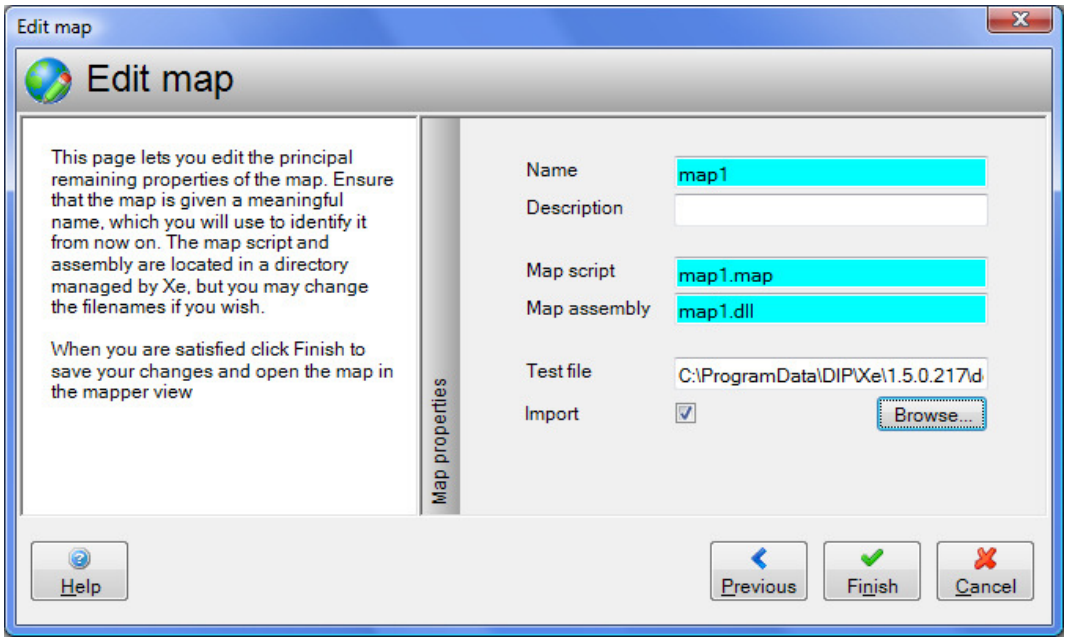
Enter a name for the new definition in the **Name** text box. The name will be used to refer to the definition in other index entities including the map, so it must be unique within the index. Consider giving the definition a meaningful name so you can identify its purpose in the future. You may optionally enter a **Description** that provides more information about the definition.

The **Syntax** drop-down list can only have the value “XML”. Enter the full path of a suitable XML file upon which the definition will be based, or click **Browse** to browse for a file. The Next button will not be enabled until a valid file and definition name are specified.

When you are satisfied, click the **Next** button to continue, or click the **Previous** button to return to the new definition page (see “Edit map – definition file”).

4.3.2.8 Edit map – properties

This is the final page of the edit map wizard, which allows you to specify the remaining data required to created the map.



Enter a name for the map in the **Name** text box. The name will be used to refer to the map in the index, so it must be unique within the index. Consider giving the map a meaningful name so you can identify its purpose in the future. You may optionally enter a **Description** that provides more information about the map.

By default the **Map script** and **Map assembly** filenames are derived from the map name, though you may change them if required. As these files will be managed by Xe in a location that it chooses (alongside the index) a full path is not required.

You may optionally enter the full path of a **Test file** (source file) to use when running the map from the Xe mapper view. If you select a test file you may also choose the **Import** option, which means that Xe will take a copy of the file and keep it locally to the index (this means it will be copied during backups and upgrades when new versions are installed).

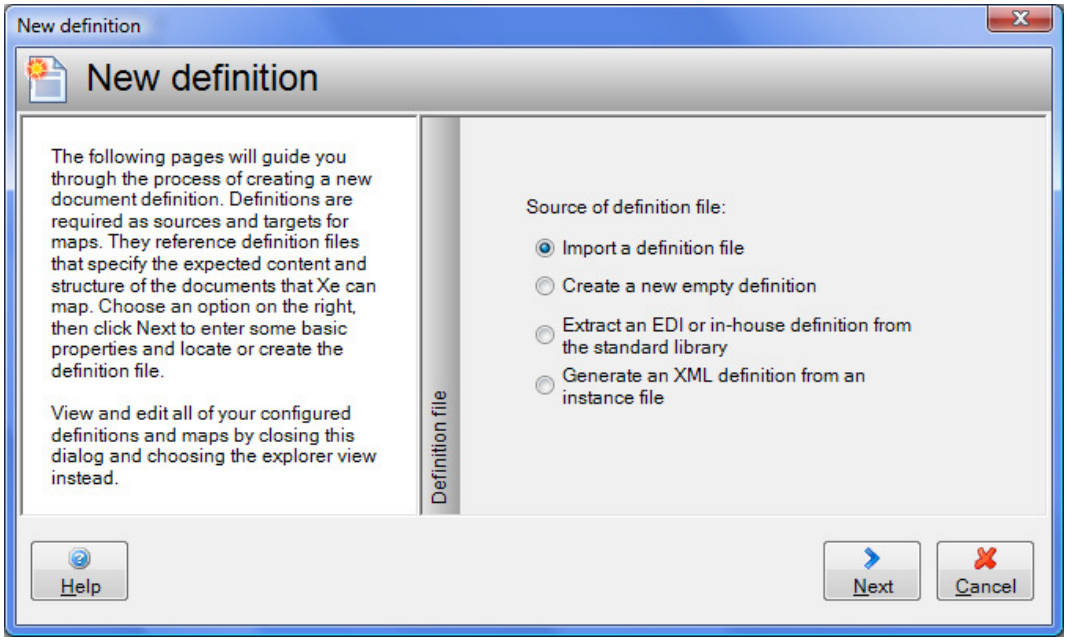
The **Finish** button is enabled only when the mandatory fields have been completed with valid values. If a test file path is entered, the **Finish** button will not be enabled unless the file exists.

When you are satisfied, click the **Finish** button to save changes to the map and open it in the mapper view. Alternatively, click the **Previous** button to return to the new definition page (see “Edit map – definition file”).

4.3.3 New definition wizard

4.3.3.1 New definition – definition file

The new definition wizard is used to create a new definition and associate it with a definition file. The first page of the wizard allows you to specify where the definition file will come from.



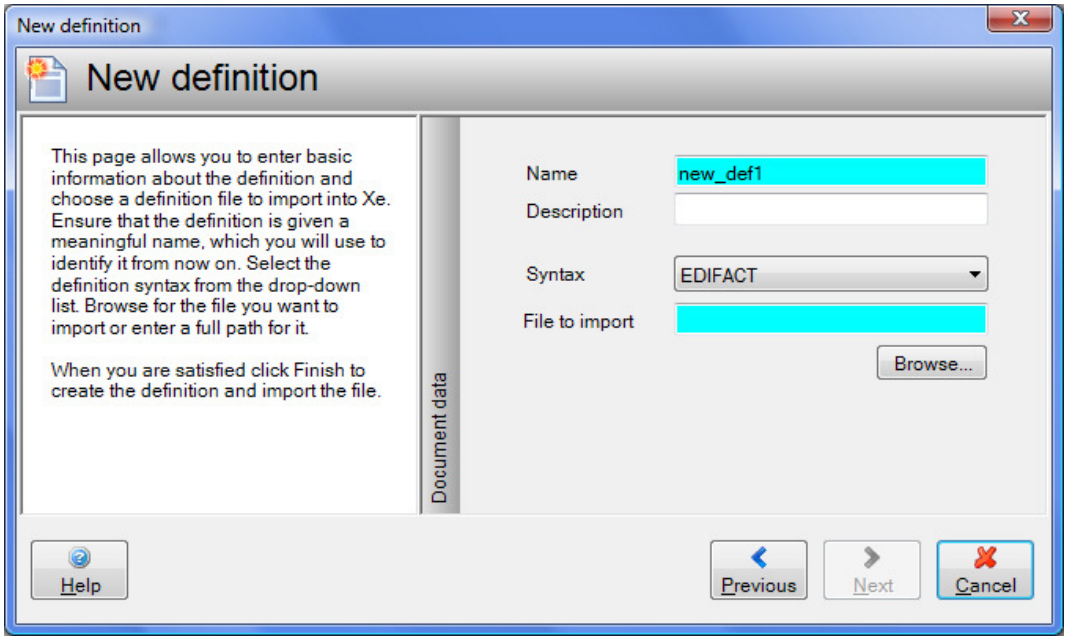
Select one option for the source then click the **Next** button to continue.

The options on this page are as follows:

- Import a definition file – import an existing definition file to use with the new definition (see “New definition – import definition file”).
- Create a new empty definition – create a blank definition using basic information you specify on the next page (see “New definition – new definition file”).
- Extract an EDI or in-house definition from the standard library – select a dictionary and message from the data dictionary and extract an EDI definition or simple in-house definition (see “New definition – create definition file from dictionary”).
- Generate an XML definition from an instance file – select an XML file and create a definition based on it (see “New definition – create definition file from XML file”).

4.3.3.2 New definition – import definition file

This page of the wizard is shown when you choose to import a definition file to use with the new document definition you are creating.



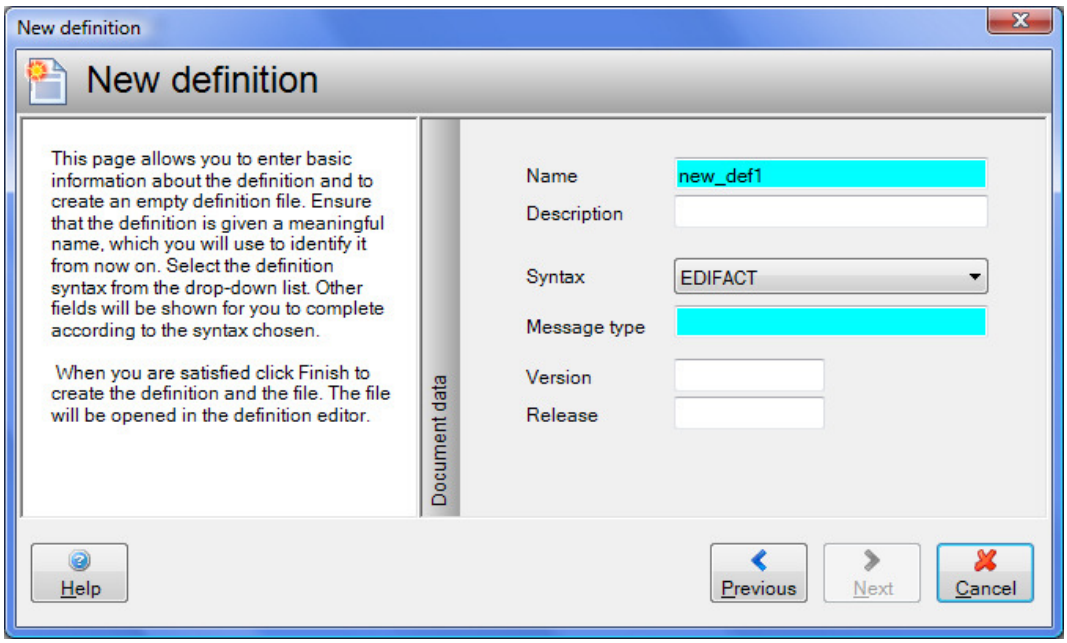
Enter a name for the new definition in the **Name** text box. The name will be used to refer to the definition in other index entities, so it must be unique within the index. Consider giving the definition a meaningful name so you can identify its purpose in the future. You may optionally enter a **Description** that provides more information about the definition.

Choose a **Syntax** from the drop-down list to specify the document type. Enter the full path of a suitable definition file to be imported, or click **Browse** to browse for a file. The **Finish** button will not be enabled until a valid file and definition name are specified.

When you are satisfied, click the **Finish** button to create the definition and open it in the editor, or click the **Previous** button to return to the new definition page (see “New definition – definition file”).

4.3.3.3 New definition – new definition file

This page of the wizard is shown when you choose to create a new empty definition file to use with the new document definition you are creating.



Enter a name for the new definition in the **Name** text box. The name will be used to refer to the definition in other index entities, so it must be unique within the index. Consider giving the definition a meaningful name so you can identify its purpose in the future. You may optionally enter a **Description** that provides more information about the definition.

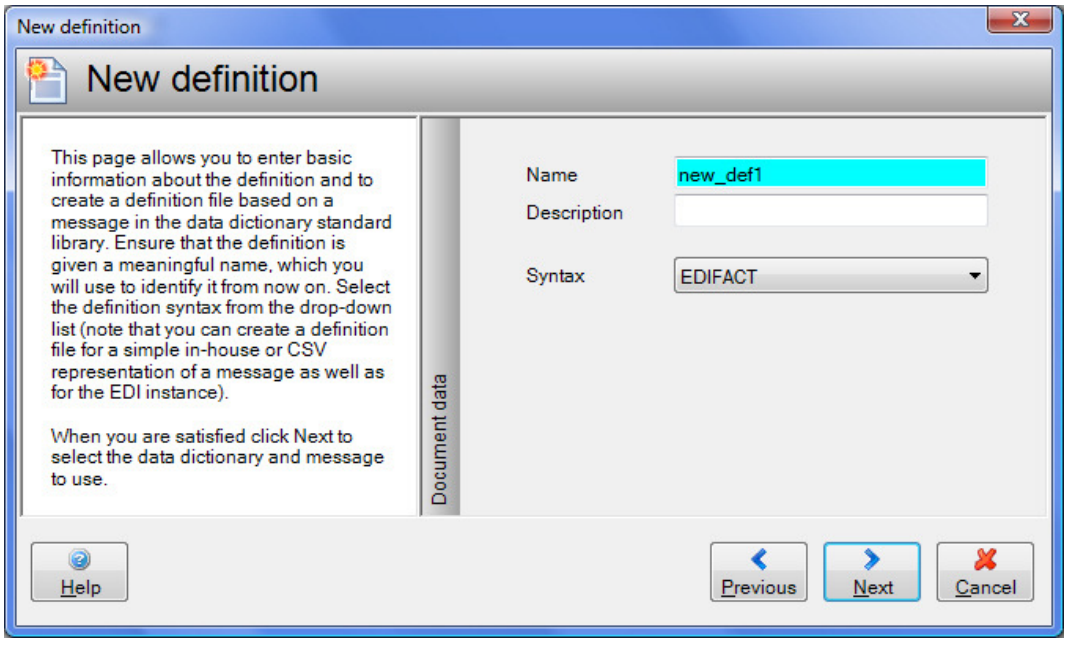
Choose a **Syntax** from the drop-down list to specify the document type. The remaining data you need to enter depends upon the syntax selected:

- For EDI syntaxes (except VDA) you are required to enter a message (or transaction set) name (such as DELFOR, 850 &c), and may optionally enter a version and release.
- For VDA you are required to enter a message name (such as 4905, 4913 &c), and may optionally enter a version.
- For IDOC and in-house definitions (flat, CSV and TRA), select a record format from the drop-down list.
- For XML and report definitions no further data is required.

When you are satisfied, click the **Finish** button to create the definition and open it in the editor, or click the **Previous** button to return to the new definition page (see “New definition – definition file”).

4.3.3.4 New definition – create definition file from dictionary

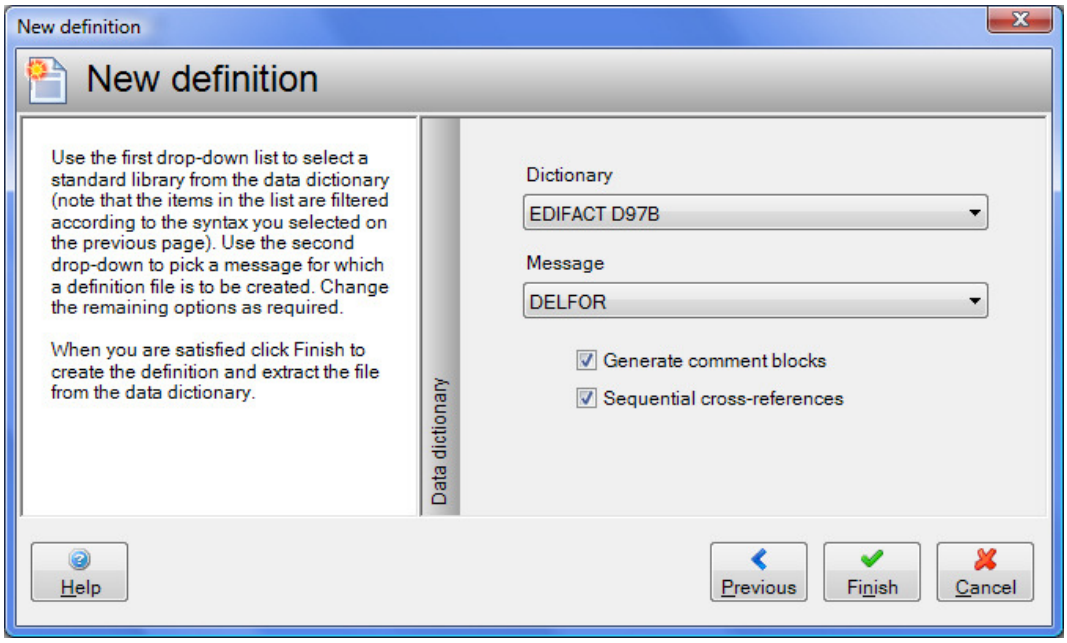
This page of the wizard is shown when you choose to create a new definition file from the data dictionary to use with the new document definition you are creating.



Enter a name for the new definition in the **Name** text box. The name will be used to refer to the definition in other index entities, so it must be unique within the index. Consider giving the definition a meaningful name so you can identify its purpose in the future. You may optionally enter a **Description** that provides more information about the definition.

Choose a **Syntax** from the drop-down list to specify the document type.

When you are satisfied, click the **Next** button to continue to the next page which allows you to select a dictionary and message. Alternatively, click the **Previous** button to return to the new definition page (see “New definition – definition file”).



This page allows you to select a **Dictionary** and **Message** from the drop-down lists (the messages available depend on the dictionary selected).

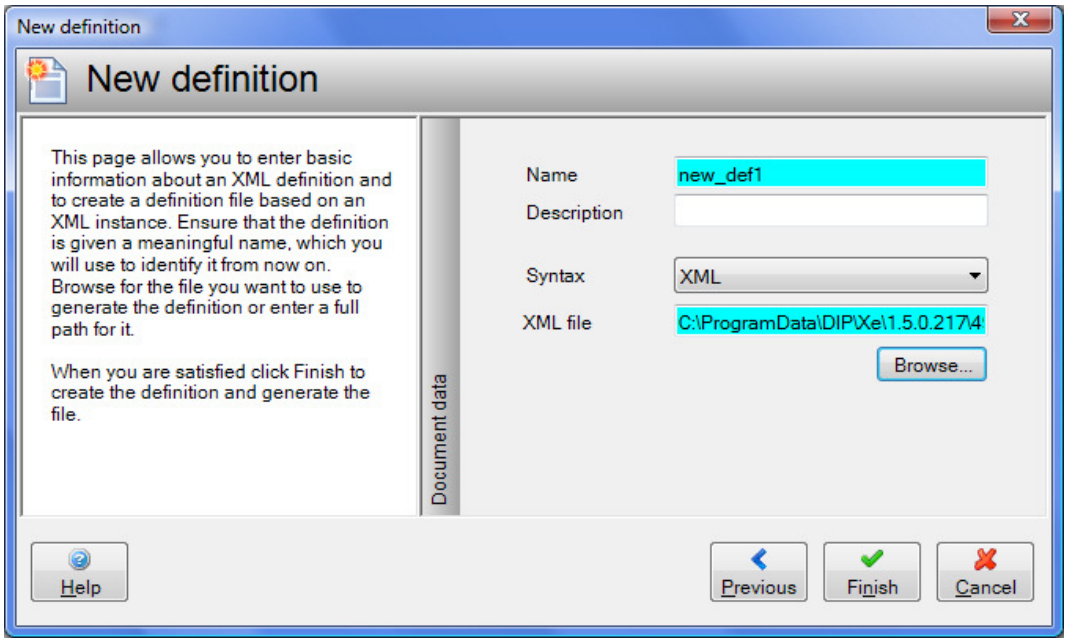
The **Generate comment blocks** option allows you to specify whether the generated definition will contain comments for each record in a preceding block.

The **Sequential cross-references** option allows you to specify whether the element cross-references generated are strictly sequential (for example; RFF00510, RFF00520). The alternative is that they are based on the occurrence number of the segment (for example; RFF020010, RFF020020 for the second instance of RFF in the definition).

When you are satisfied, click the **Finish** button to create the definition and open it in the editor, or click the **Previous** button to return to the previous page.

4.3.3.5 New definition – create definition file from XML file

This page of the wizard is shown when you choose to create a new XML definition file to use with the new document definition you are creating, based on an XML instance.



Enter a name for the new definition in the **Name** text box. The name will be used to refer to the definition in other index entities, so it must be unique within the index. Consider giving the definition a meaningful name so you can identify its purpose in the future. You may optionally enter a **Description** that provides more information about the definition.

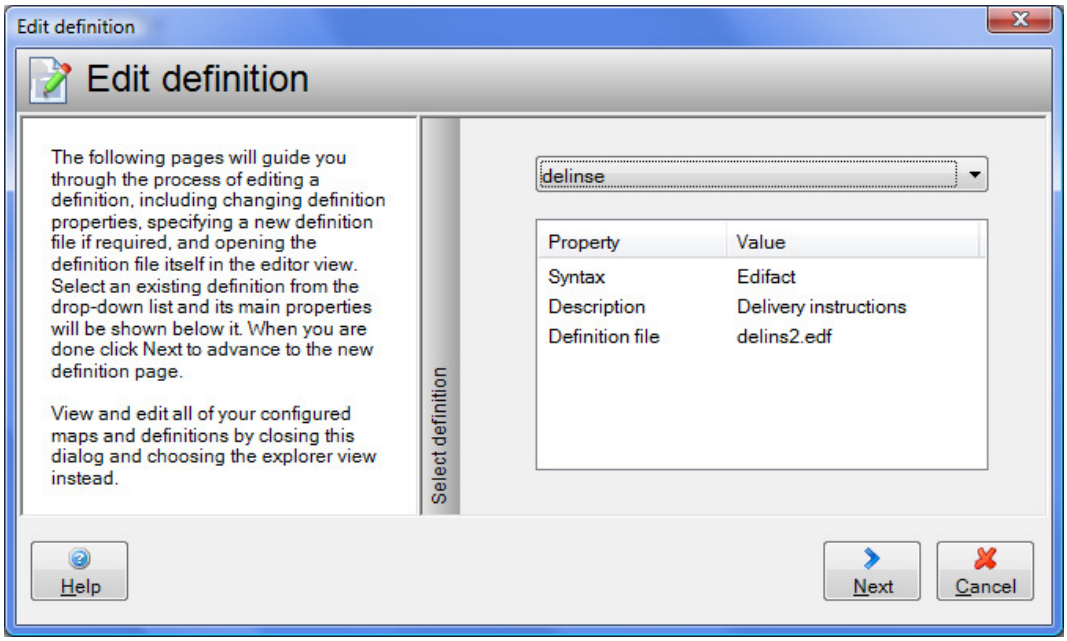
The **Syntax** drop-down list can only have the value “XML”. Enter the full path of a suitable XML file upon which the definition will be based, or click **Browse** to browse for a file. The **Finish** button will not be enabled until a valid file and definition name are specified.

When you are satisfied, click the **Finish** button to create the definition and open it in the editor, or click the **Previous** button to return to the new definition page (see “New definition – definition file”).

4.3.4 Edit definition wizard

4.3.4.1 Edit definition – select definition

The edit definition wizard is used to edit a definition, optionally changing the definition file with which it is associated.

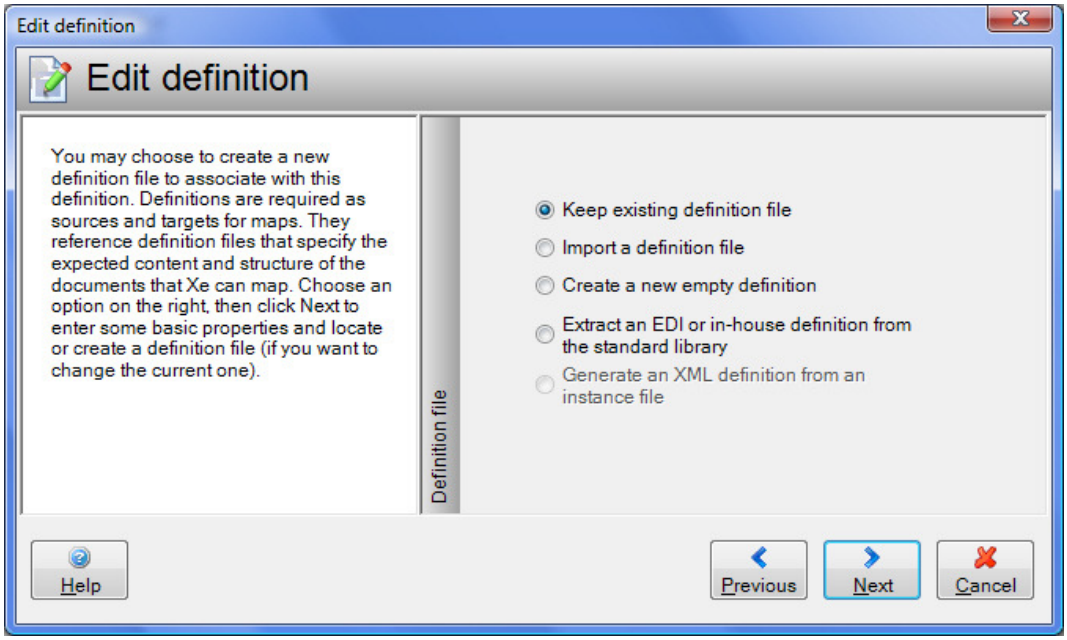


Use the drop-down to select the definition to be edited. Each time you change the selection, the properties of the definition will be shown in the list.

When you are satisfied with the selection, click the **Next** button to continue.

4.3.4.2 Edit definition – definition file

This page of the wizard allows you to specify where the definition file will come from.



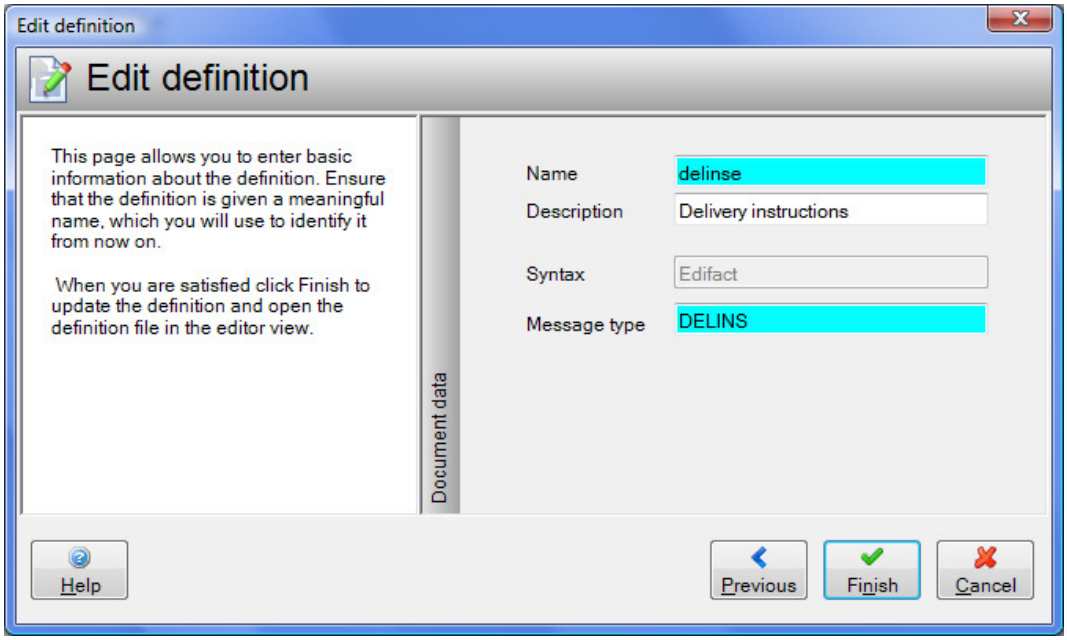
Select one option for the source then click the **Next** button to continue.

The options on this page are as follows:

- Keep existing definition file – use the definition file currently associated with the definition (see “Edit definition – keep definition file”).
- Import a definition file – import an existing definition file to use with the definition (see “Edit definition – import definition file”).
- Create a new empty definition – create a blank definition using basic information you specify on the next page (see “Edit definition – new definition file”).
- Extract an EDI or in-house definition from the standard library – select a dictionary and message from the data dictionary and extract an EDI definition or simple in-house definition (see “Edit definition - create definition file from data dictionary”).
- Generate an XML definition from an instance file – select an XML file and create a definition based on it (see “Edit definition - create definition file from XML file”).

4.3.4.3 Edit definition – keep definition file

This page of the wizard allows you to edit basic data about the definition when you are not changing the definition file associated with it.



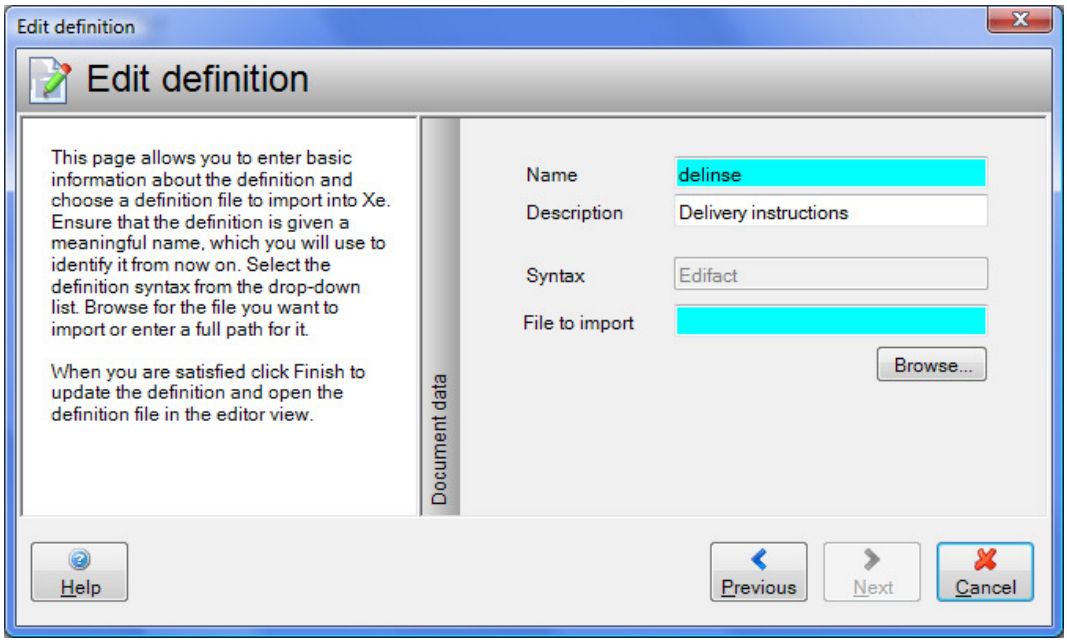
Enter a name for the definition in the **Name** text box. The name will be used to refer to the definition in other index entities, so it must be unique within the index. Consider giving the definition a meaningful name so you can identify its purpose in the future. You may optionally enter a **Description** that provides more information about the definition.

The **Syntax** cannot be changed from this wizard. If you need to change the syntax you can either create a new definition with the correct syntax, or switch to the Explorer view.

When you are satisfied, click the **Finish** button to create the definition and open it in the editor, or click the **Previous** button to return to the definition page (see “Edit definition – definition file”).

4.3.4.4 Edit definition – import definition file

This page of the wizard is shown when you choose to import a definition file to use with the document definition you are editing.



Enter a name for the definition in the **Name** text box. The name will be used to refer to the definition in other index entities, so it must be unique within the

index. Consider giving the definition a meaningful name so you can identify its purpose in the future. You may optionally enter a **Description** that provides more information about the definition.

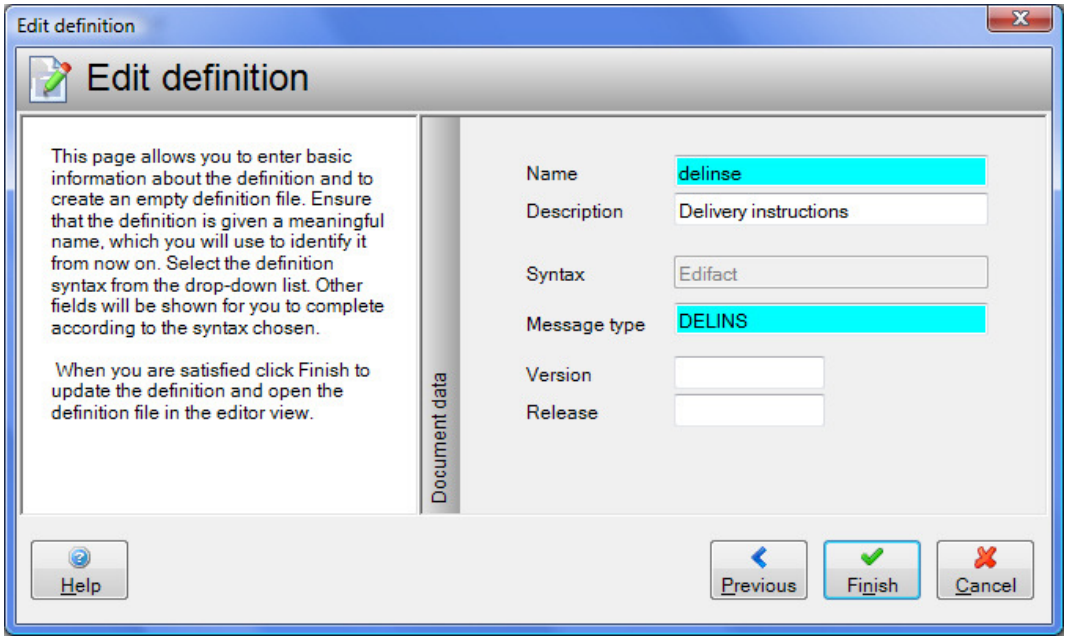
The **Syntax** cannot be changed from this wizard. If you need to change the syntax you can either create a new definition with the correct syntax, or switch to the Explorer view.

Enter the full path of a suitable definition file to be imported, or click **Browse** to browse for a file. The **Finish** button will not be enabled until a valid file and definition name are specified.

When you are satisfied, click the **Finish** button to create the definition and open it in the editor, or click the **Previous** button to return to the definition page (see "Edit definition – definition file").

4.3.4.5 Edit definition – new definition file

This page of the wizard is shown when you choose to create a new empty definition file to use with the document definition you are editing.



Enter a name for the definition in the **Name** text box. The name will be used to refer to the definition in other index entities, so it must be unique within the index. Consider giving the definition a meaningful name so you can identify its purpose in the future. You may optionally enter a **Description** that provides more information about the definition.

The **Syntax** cannot be changed from this wizard. If you need to change the syntax you can either create a new definition with the correct syntax, or switch to the Explorer view. The remaining data you need to enter depends upon the syntax:

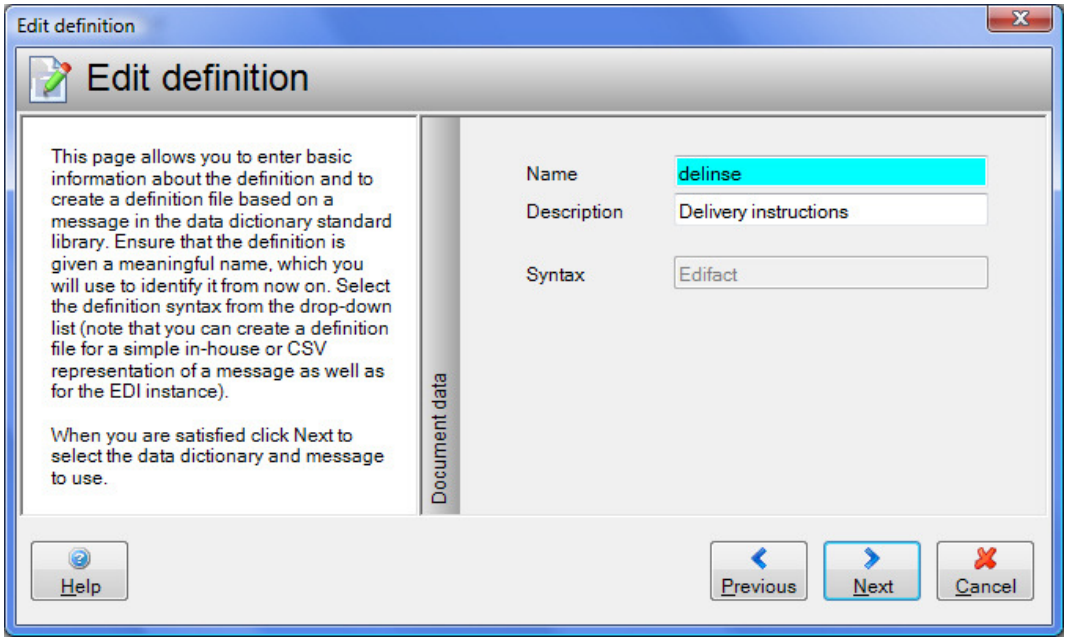
- For EDI syntaxes (except VDA) you are required to enter a message (or transaction set) name (such as DELFOR, 850 &c), and may optionally enter a version and release.
- For VDA you are required to enter a message name (such as 4905, 4913 &c), and may optionally enter a version.
- For IDOC and in-house definitions (flat, CSV and TRA), select a record format from the drop-down list.

- For XML and report definitions no further data is required.

When you are satisfied, click the **Finish** button to create the definition and open it in the editor, or click the **Previous** button to return to the definition page (see “Edit definition – definition file”).

4.3.4.6 Edit definition - create definition file from data dictionary

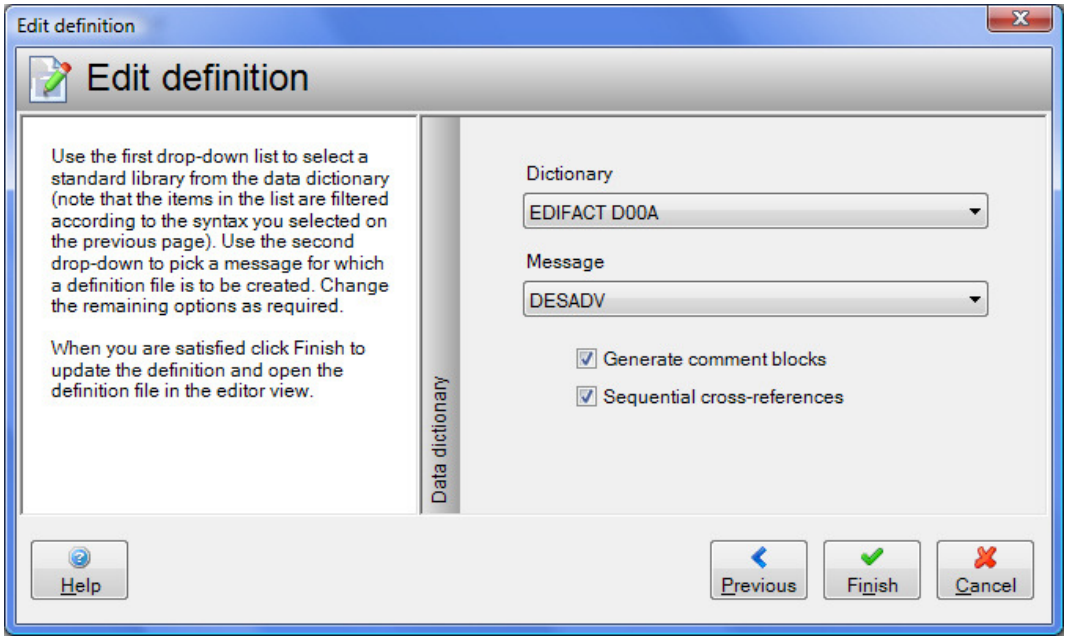
This page of the wizard is shown when you choose to create a new definition file from the data dictionary to use with the document definition you are editing.



Enter a name for the new definition in the **Name** text box. The name will be used to refer to the definition in other index entities, so it must be unique within the index. Consider giving the definition a meaningful name so you can identify its purpose in the future. You may optionally enter a **Description** that provides more information about the definition.

The **Syntax** cannot be changed from this wizard. If you need to change the syntax you can either create a new definition with the correct syntax, or switch to the Explorer view.

When you are satisfied, click the **Next** button to continue to the next page which allows you to select a dictionary and message. Alternatively, click the **Previous** button to return to the definition page (see “Edit definition – definition file”).



This page allows you to select a **Dictionary** and **Message** from the drop-down lists (the messages available depend on the dictionary selected).

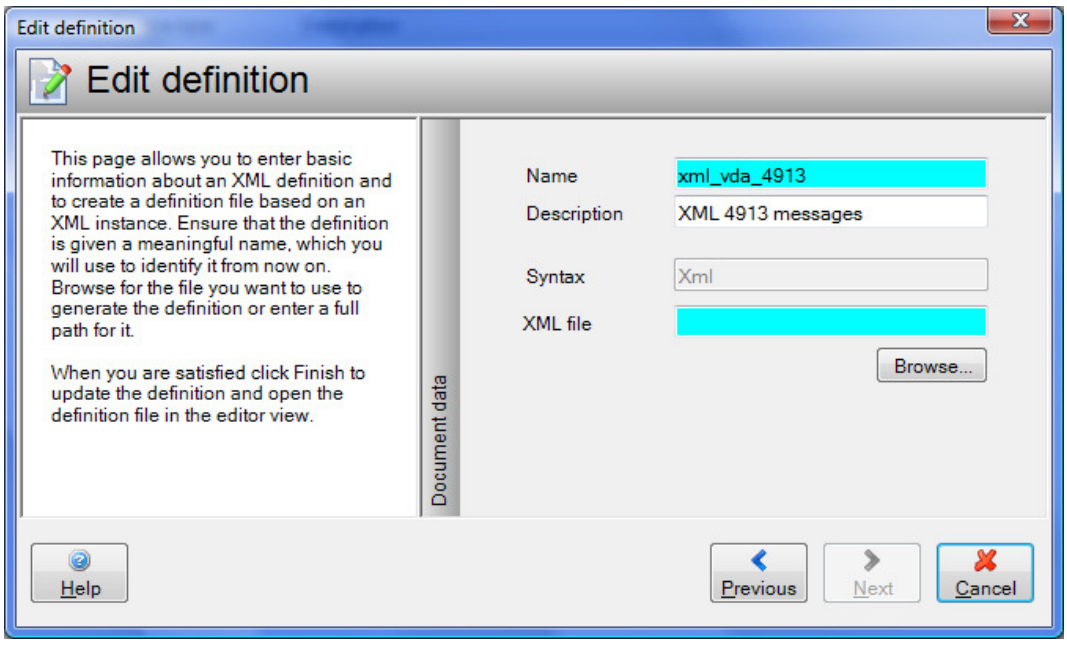
The **Generate comment blocks** option allows you to specify whether the generated definition will contain comments for each record in a preceding block.

The **Sequential cross-references** option allows you to specify whether the element cross-references generated are strictly sequential (for example; RFF00510, RFF00520). The alternative is that they are based on the occurrence number of the segment (for example; RFF020010, RFF020020 for the second instance of RFF in the definition).

When you are satisfied, click the **Finish** button to create the definition and open it in the editor, or click the **Previous** button to return to the previous page.

4.3.4.7 Edit definition - create definition file from XML file

This page of the wizard is shown when you choose to create a new XML definition file to use with the document definition you are editing, based on an XML instance.



Enter a name for the definition in the **Name** text box. The name will be used to refer to the definition in other index entities, so it must be unique within the index. Consider giving the definition a meaningful name so you can identify its purpose in the future. You may optionally enter a **Description** that provides more information about the definition.

The **Syntax** cannot be changed from this wizard. If you need to change the syntax you can either create a new definition with the correct syntax, or switch to the Explorer view.

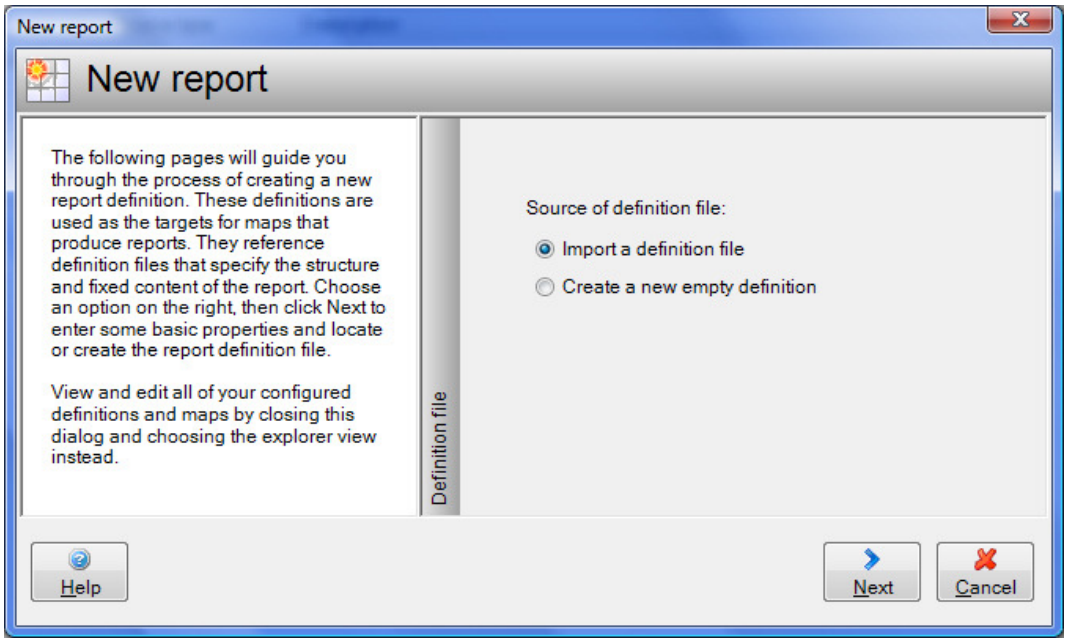
Enter the full path of a suitable XML file upon which the definition will be based, or click **Browse** to browse for a file. The **Finish** button will not be enabled until a valid file and definition name are specified.

When you are satisfied, click the **Finish** button to create the definition and open it in the editor, or click the **Previous** button to return to the definition page (see "Edit definition – definition file").

4.3.5 New report wizard

4.3.5.1 New report – definition file

The new report definition wizard is used to create a new report definition and associate it with a report definition file (RDF). The first page of the wizard allows you to specify where the definition file will come from.



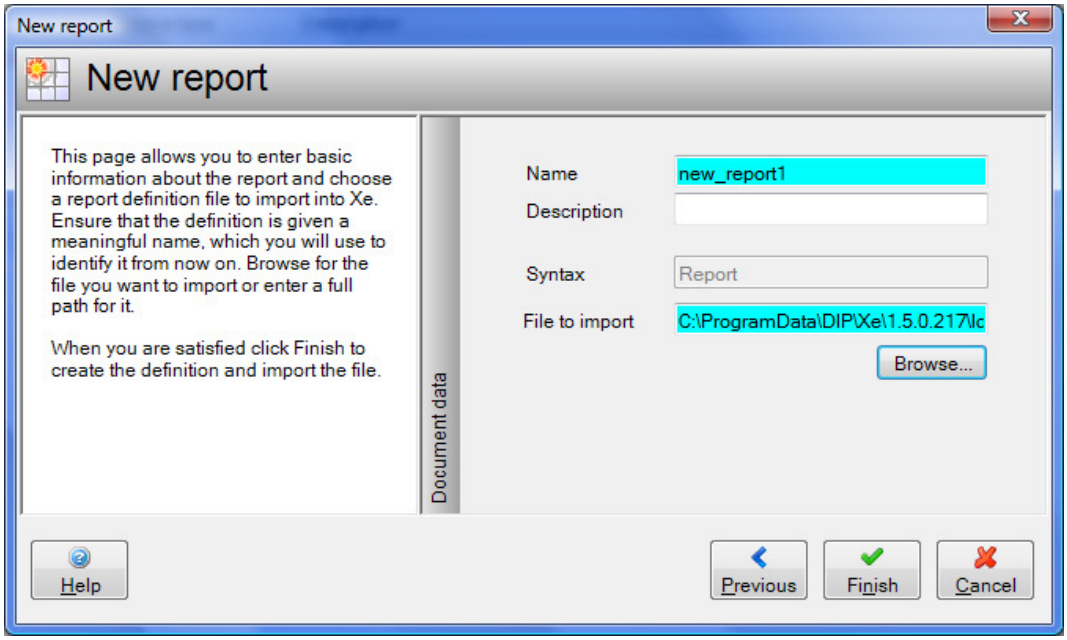
Select one option for the source then click the **Next** button to continue.

The options on this page are as follows:

- Import a definition file – import an existing report definition file to use with the new report definition (see "New report – import report definition file").
- Create a new empty definition – create a blank report definition using basic information you specify on the next page (see "New report - new report definition file").

4.3.5.2 New report – import report definition file

This page of the wizard is shown when you choose to import a report definition file to use with the new report definition you are creating.



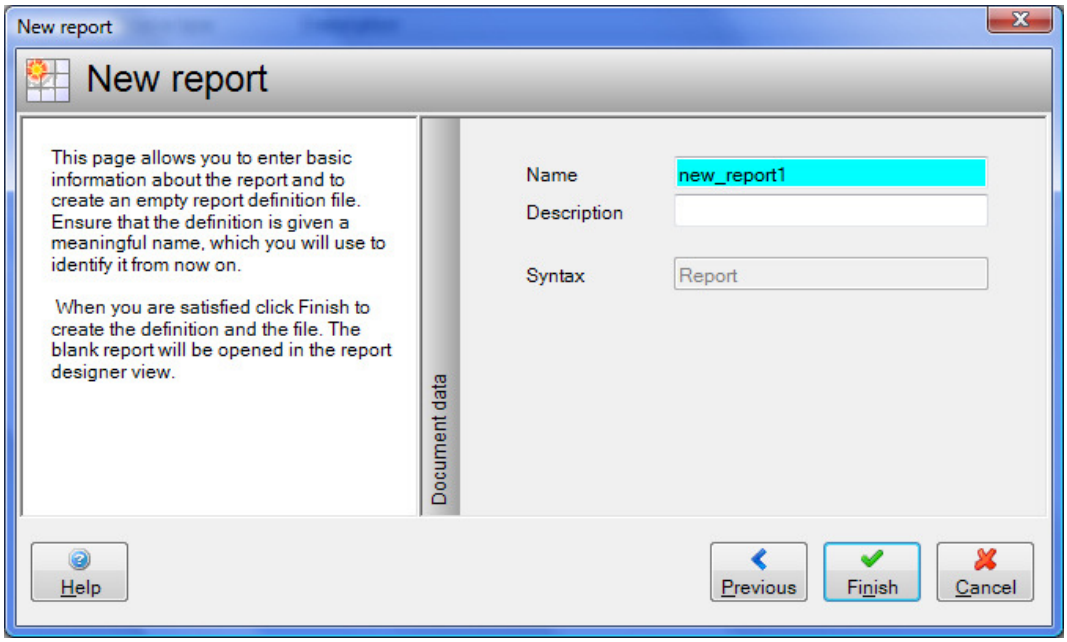
Enter a name for the report definition in the **Name** text box. The name will be used to refer to the definition in other index entities, so it must be unique within the index. Consider giving the definition a meaningful name so you can identify its purpose in the future. You may optionally enter a **Description** that provides more information about the definition.

Enter the full path of a suitable report definition file to be imported, or click **Browse** to browse for a file. The **Finish** button will not be enabled until a valid file and definition name are specified.

When you are satisfied, click the **Finish** button to create the definition and open the report definition in the report designer, or click the **Previous** button to return to the new report definition page (see “New report – definition file”).

4.3.5.3 New report - new report definition file

This page of the wizard is shown when you choose to create a new empty report definition file to use with the new report definition you are creating.



Enter a name for the report definition in the **Name** text box. The name will be used to refer to the definition in other index entities, so it must be unique within

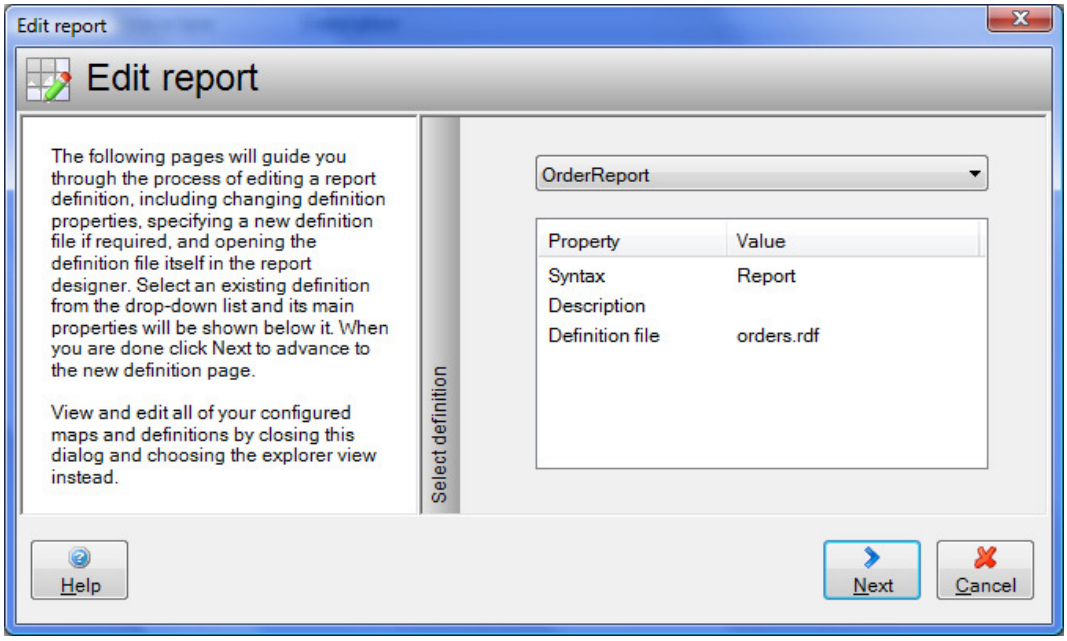
the index. Consider giving the definition a meaningful name so you can identify its purpose in the future. You may optionally enter a **Description** that provides more information about the definition.

When you are satisfied, click the **Finish** button to create the definition and open the report definition in the report designer, or click the **Previous** button to return to the new report definition page (see “New report – definition file”).

4.3.6 Edit report wizard

4.3.6.1 Edit report – select report

The edit report definition wizard is used to edit a report definition, optionally changing the definition file with which it is associated.

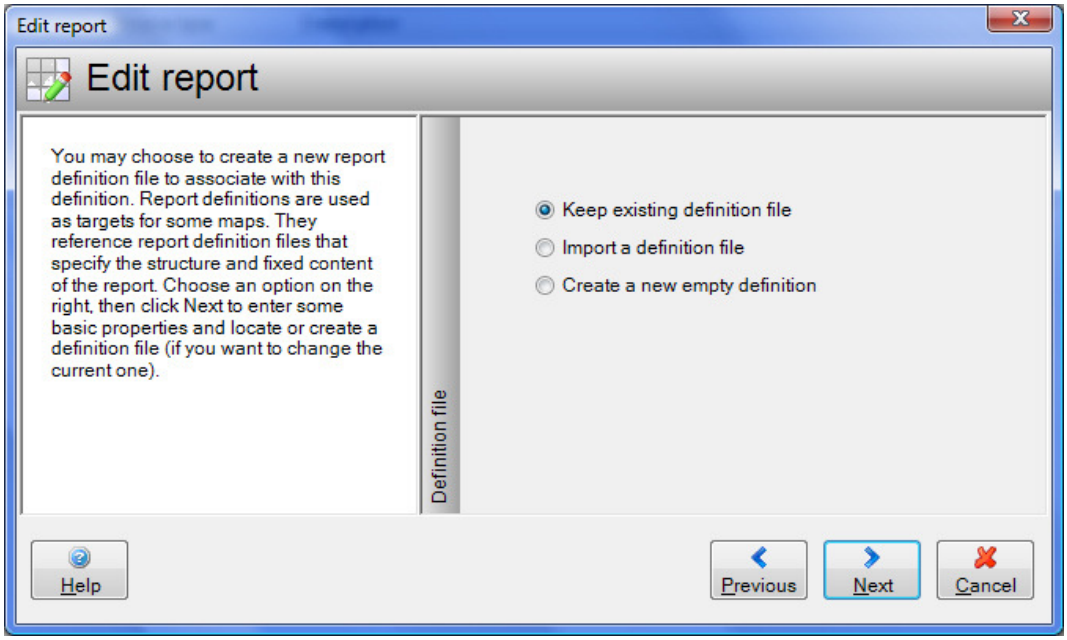


Use the drop-down to select the report definition to be edited. Each time you change the selection, the properties of the report definition will be shown in the list.

When you are satisfied with the selection, click the **Next** button to continue.

4.3.6.2 Edit report – definition file

This page of the wizard allows you to specify where the report definition file will come from.



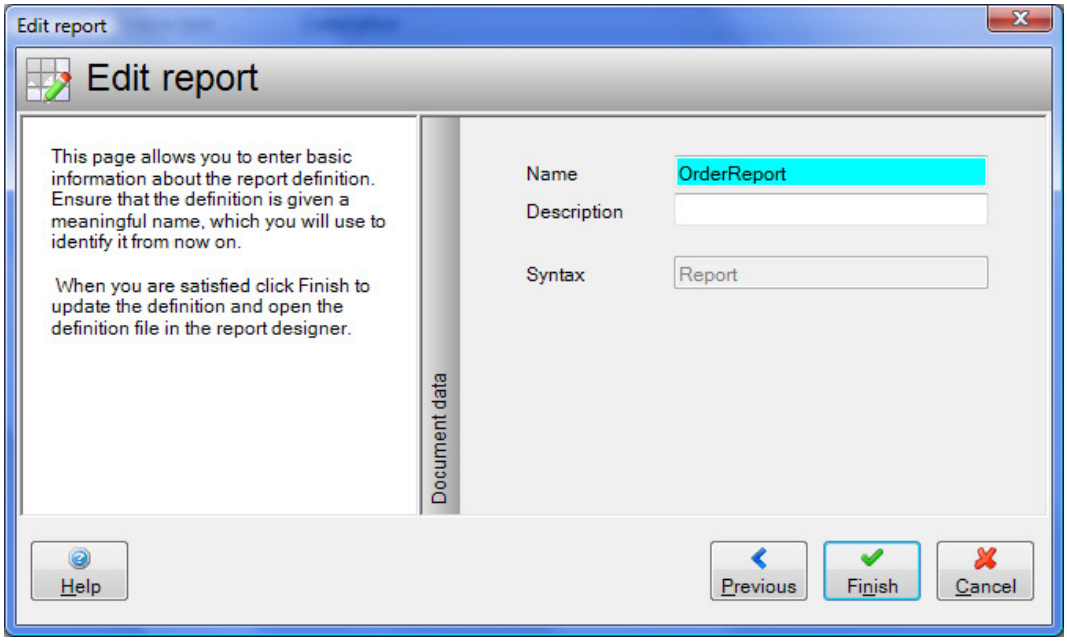
Select one option for the source then click the **Next** button to continue.

The options on this page are as follows:

- Keep existing definition file – use the report definition file currently associated with the definition (see “Edit report – keep report definition file”).
- Import a definition file – import an existing report definition file to use with the definition (see “Edit report – import report definition file”).
- Create a new empty definition – create a blank report definition using basic information you specify on the next page (see “Edit report – new report definition file”).

4.3.6.3 Edit report – keep report definition file

This page of the wizard allows you to edit basic data about the report definition when you are not changing the definition file associated with it.



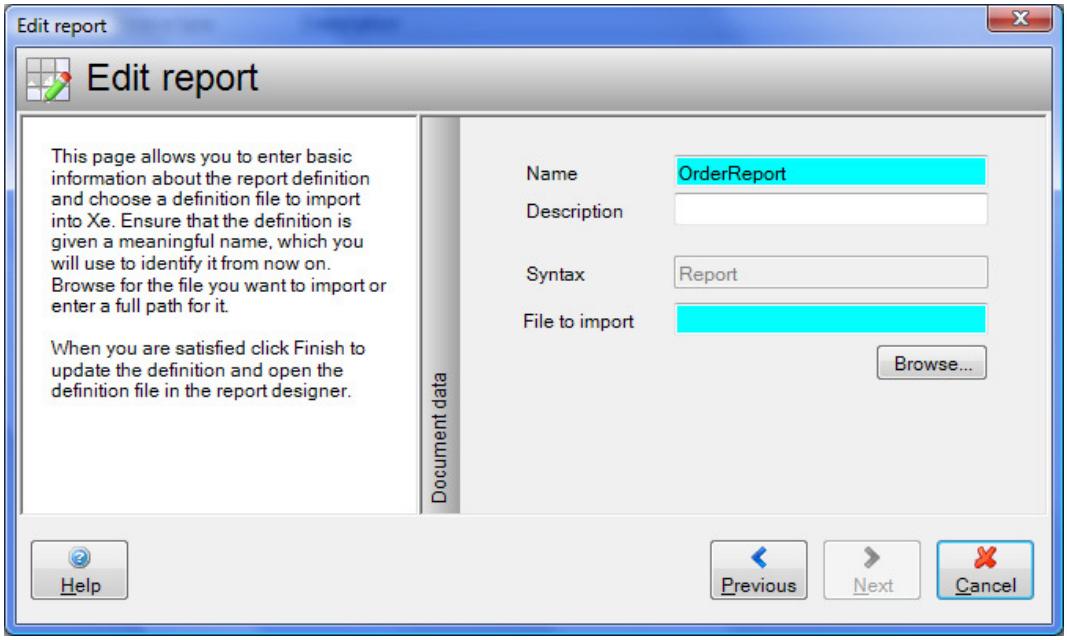
Enter a name for the report definition in the **Name** text box. The name will be used to refer to the definition in other index entities, so it must be unique within

the index. Consider giving the definition a meaningful name so you can identify its purpose in the future. You may optionally enter a **Description** that provides more information about the definition.

When you are satisfied, click the **Finish** button to save changes and open the report in the designer, or click the **Previous** button to return to the definition page (see “Edit report – definition file”).

4.3.6.4 Edit report – import report definition file

This page of the wizard is shown when you choose to import a report definition file to use with the definition you are editing.



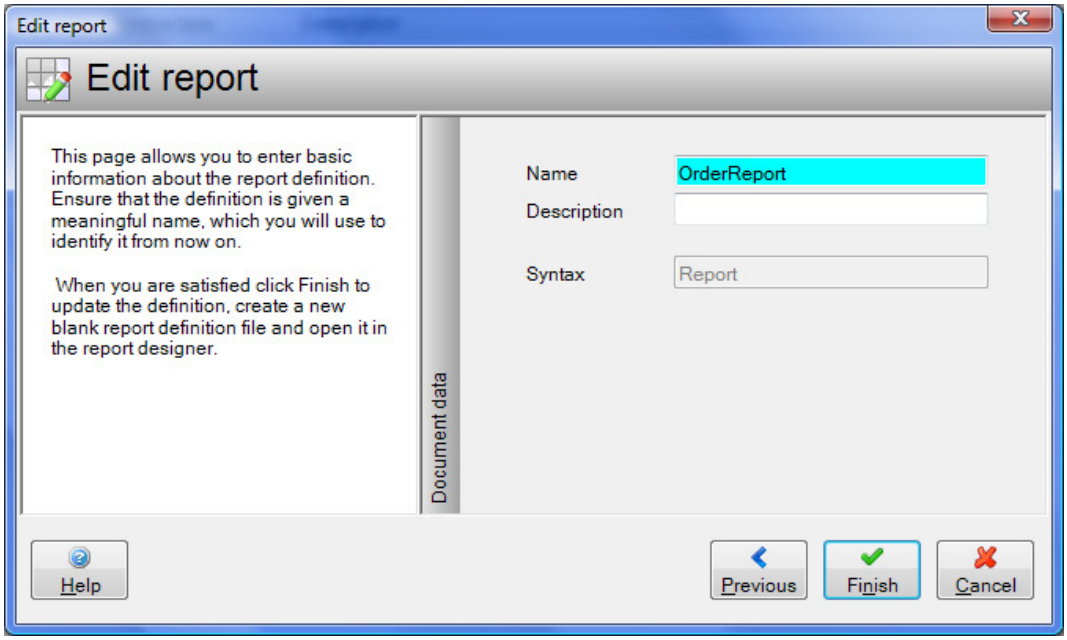
Enter a name for the report definition in the **Name** text box. The name will be used to refer to the definition in other index entities, so it must be unique within the index. Consider giving the definition a meaningful name so you can identify its purpose in the future. You may optionally enter a **Description** that provides more information about the definition.

Enter the full path of a suitable report definition file to be imported, or click **Browse** to browse for a file. The **Finish** button will not be enabled until a valid file and definition name are specified.

When you are satisfied, click the **Finish** button to save changes and open the report in the designer, or click the **Previous** button to return to the definition page (see “Edit report – definition file”).

4.3.6.5 Edit report – new report definition file

This page of the wizard is shown when you choose to create a new empty report definition file to use with the definition you are editing.



Enter a name for the report definition in the **Name** text box. The name will be used to refer to the definition in other index entities, so it must be unique within the index. Consider giving the definition a meaningful name so you can identify its purpose in the future. You may optionally enter a **Description** that provides more information about the definition.

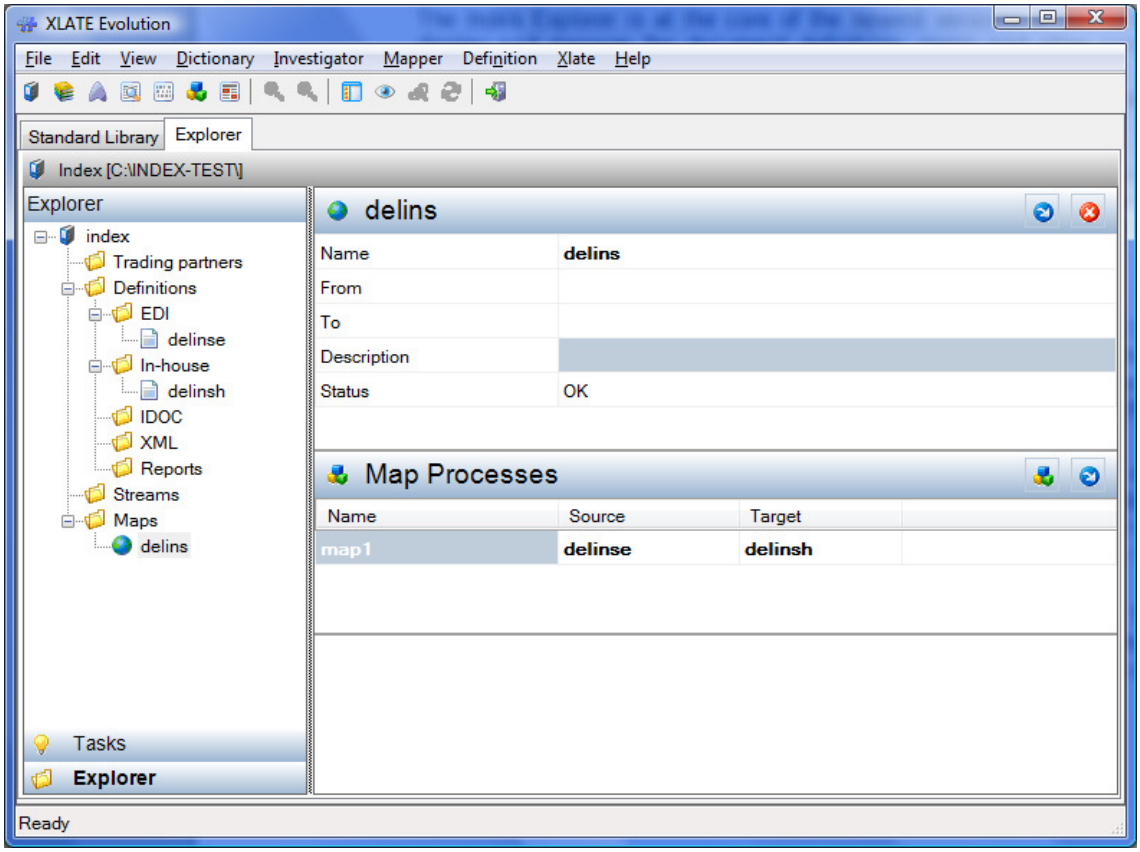
When you are satisfied, click the **Finish** button to save changes and open the report in the designer, or click the **Previous** button to return to the definition page (see “Edit report – definition file”).

4.3.7 View EDI file

Launches the File Investigator. Click the button marked “View an EDI file” to select an EDI file to open. See the section entitled “File Investigator” for more information.

4.4 The Explorer View

The Explorer view of the Index Explorer is show below.



The purpose of the Explorer view is to provide access to all of the index entities defined in the current index. The left of the screen is taken up by a tree view which can be navigated to select entities such as trading partners, definitions and maps. Items in the tree view are grouped into folders representing categories of index entities. If a folder is clicked then the right hand side of the view shows a list of items in the folder. If an entity is clicked in the tree view then the right hand side shows one or more lists of properties relating to the selected entity.

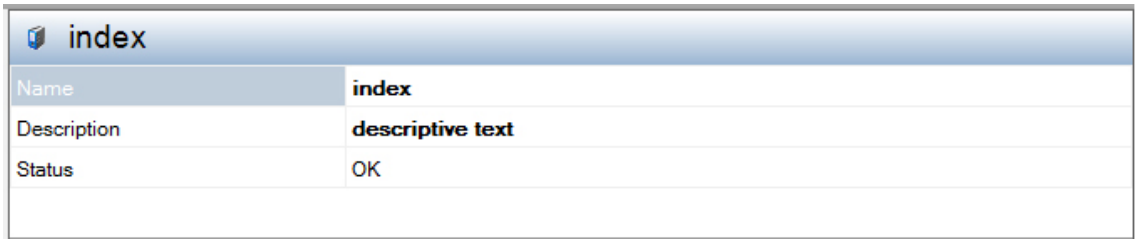
The lists shown in the right hand side of the view consist of a banner followed by the list items. The banner contains a title, which is either a description of the entities contained in the list or the name of the entity being displayed. It may also have buttons on the right hand side (add, edit and delete) that are used to manipulate the entity or list selection. The list can be expanded or collapsed by clicking on the banner.

The following sections describe the lists shown when various items are selected in the tree view and how they are used for editing the related index entities.

4.4.1 Index entity

When the index entity is clicked in the tree view these lists are displayed:

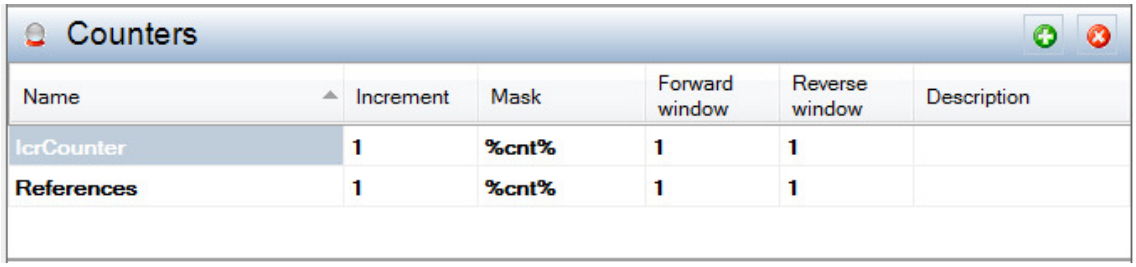
Index Properties



The index properties list contains two editable properties: index name and description. If you are using a single index then name and description are not very important, but if you use (or are likely to use) Xe for managing multiple indexes (for different users or systems, say) then it will help you to identify them more easily if you choose a suitably distinctive name and add descriptive text.

The status field is read-only and is there to indicate whether the index is ready for use. If there are any incomplete index entities or any potential runtime conflicts then the index status will not be OK and one or more messages will be displayed here indicating where the errors are to be found.

Counters



This list displays the counters that are defined in the index. An index counter can be associated with a stream (used to verify and/or create interchange control references) or accessed from code snippets in the mapper.

Click the **Add** button ( ) to add a new counter and then edit its properties directly in the list:

- Name – provide a meaningful name that will be used to identify the counter
- Increment – specify the amount by which the counter is incremented each time
- Mask – type a mask used to specify how the counter value is formatted. For example, ABC%cnt:5% produces counter values ABC00001, ABC00002, &c.
- Forward window – when used for interchange control reference validation by a stream, indicates the maximum value by which the reference is allowed to exceed the current counter value without failing validation.
- Reverse window – when used for interchange control reference validation by a stream, indicates the maximum value by which the reference is allowed to fall short of the current counter value without failing validation.
- Description – enter a meaningful description for the counter.

Click the **Delete** button ( ) to delete the currently selected counter.

Lookup Tables

The list displays the lookup tables that are defined in the index. Lookup tables contain pairs of values and are used during mapping to convert codes from one value (usually the left hand value in the table) to another (usually the right hand

value). Lookup table files are one of the file types now managed by Xe in its directory structure, so if you create a new table it will be created in a location that Xe chooses, and if you want to use an existing table, it will need to be imported into a location that Xe chooses.

Click the **Add** button () to add a new table definition to the index. Edit the list directly to set the table name (by which it will be referenced in the mapper) and description. You will note that the definition is created without a table file and that the status of the index and the table is not OK until a table file is specified.

Click the **Edit** button () to display options for choosing or manipulating the table file associated with the currently selected table definition. The options are:

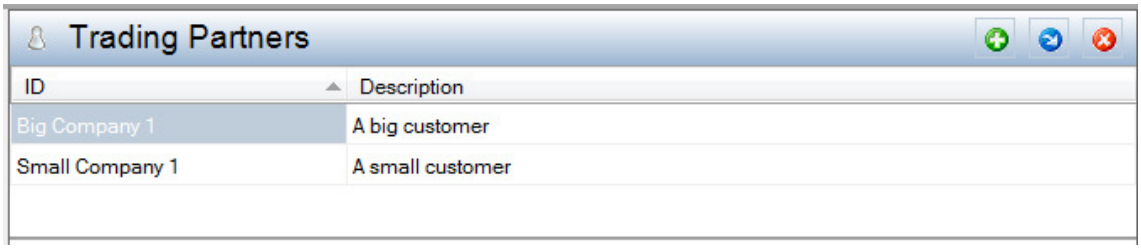
- Edit table – open a dialog containing the table contents so that the values can be edited.
- New table – create a new table file and open the edit dialog. If a file is already specified then the newly created one will supplant it.
- Import table file – import an existing table file by browsing for a file on disk. If a file is already specified then the imported one will supplant it.
- Rename file – change the name of the table file (the location cannot be changed as it is managed by Xe).

More information about the table dialog can be found in the section ”Editing Table Files”.

Click the **Delete** button () to delete the currently selected table definition.

4.4.2 Trading partner folder

When the trading partner folder is clicked in the tree view this list is displayed:



The list shows the trading partners currently defined in the index. The company name and description are shown.

Click the **Add** button () to create a new trading partner definition. A new node will be added to the tree view and the list views will change to allow the new entity to be edited.

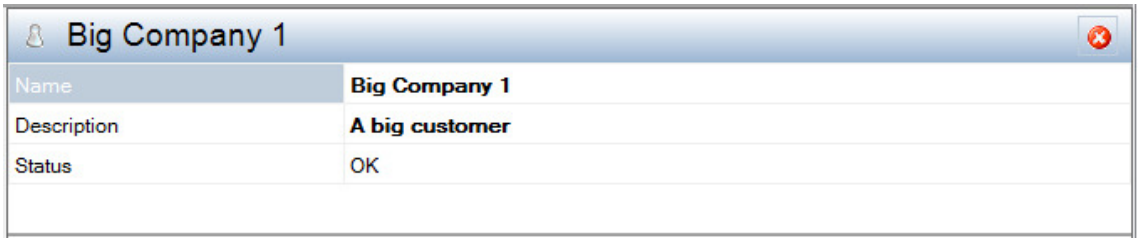
Click the **Edit** button () to edit the currently selected trading partner definition. The corresponding node will be selected in the tree view and the list views will change to allow the entity to be edited.

Click the **Delete** button () to delete the currently selected trading partner.

4.4.3 Trading partner entity

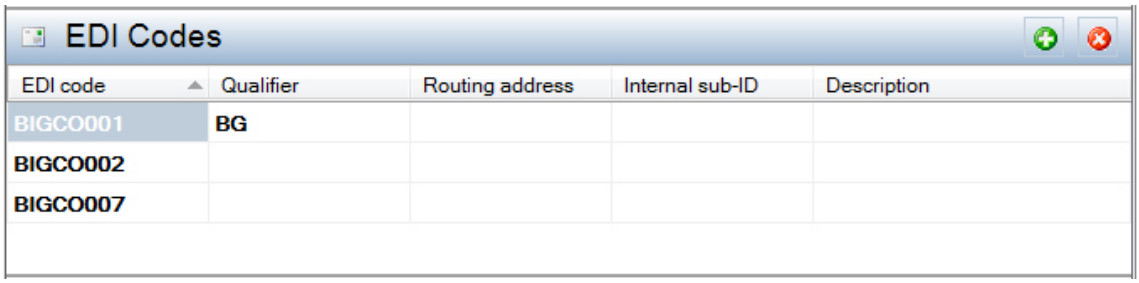
When a trading partner entity is clicked in the tree view these lists are displayed:

Trading Partner Properties



This list displays the properties of the trading partner. You can directly edit the company name and description in the list view. The name is used to reference the company from other parts of the index, so consider giving it a meaningful name.

The status field is read-only and is there to indicate whether the trading partner definition is complete. If there are any problems with the definition then the status will not be OK and one or more messages will be displayed here indicating what the errors are.



The list displays EDI codes defined for this trading partner.

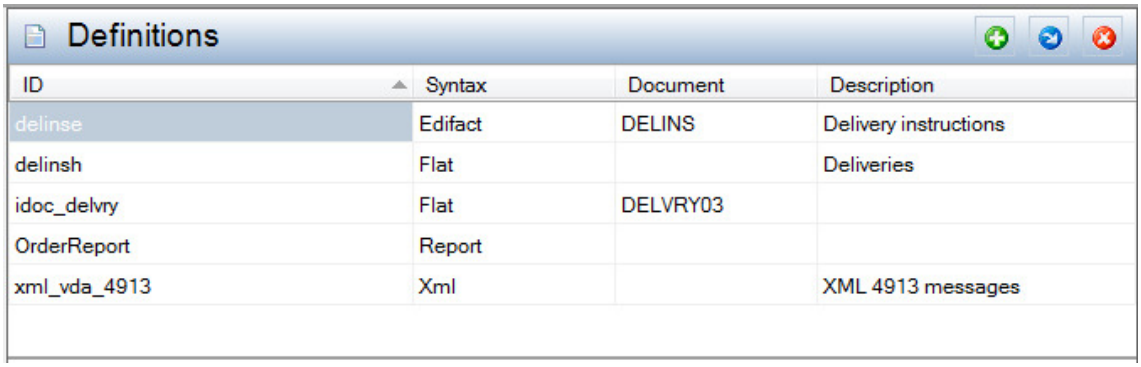
Click the **Add** button ( ) to add a new EDI code.

You can directly edit the values in the list. No line should have exactly the same values in it. If you enter a duplicate code then the status of the trading partner will change to reflect this.

Click the **Delete** button ( ) to delete the currently selected EDI code.

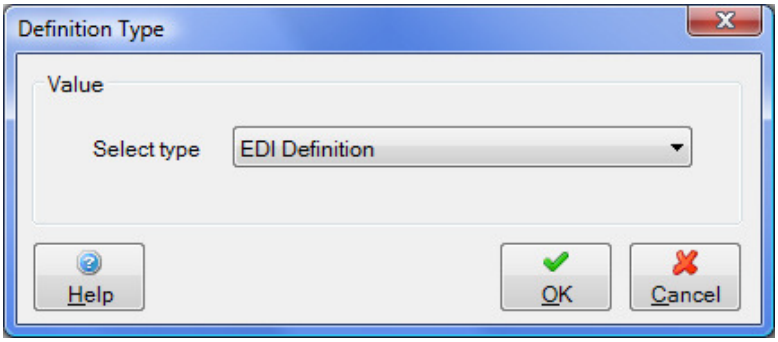
4.4.4 Definitions folder

When the definitions folder is clicked in the tree view this list is displayed:



The list shows the document definitions currently specified in the index. The definition name, syntax and description are shown. In addition, the document type is shown if available (for EDI messages and IDOCs)

Click the **Add** button ( ) to create a new document definition and you will be prompted to choose the definition type with the following dialog:



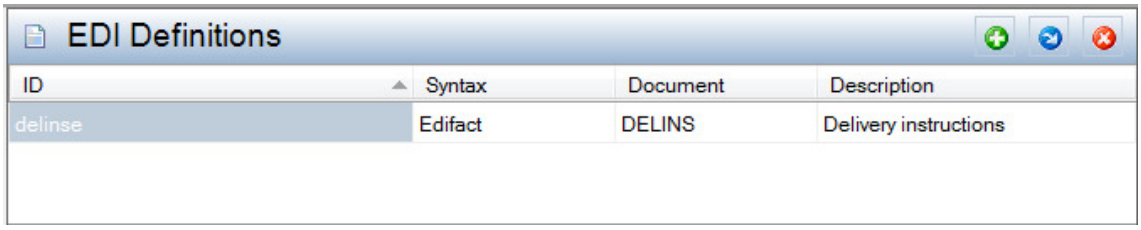
A new node will be added to the tree view and the list views will change to allow the new entity to be edited.

Click the **Edit** button ( ) to edit the currently selected document definition. The corresponding node will be selected in the tree view and the list views will change to allow the entity to be edited.

Click the **Delete** button ( ) to delete the currently selected definition.

4.4.5 EDI definitions folder

When the EDI definitions folder is clicked in the tree view this list is displayed:



The list shows the EDI document definitions currently specified in the index. The definition name, syntax, message (document) type and description are shown.

Click the **Add** button ( ) to create a new EDI definition. A new node will be added to the tree view and the list views will change to allow the new entity to be edited.

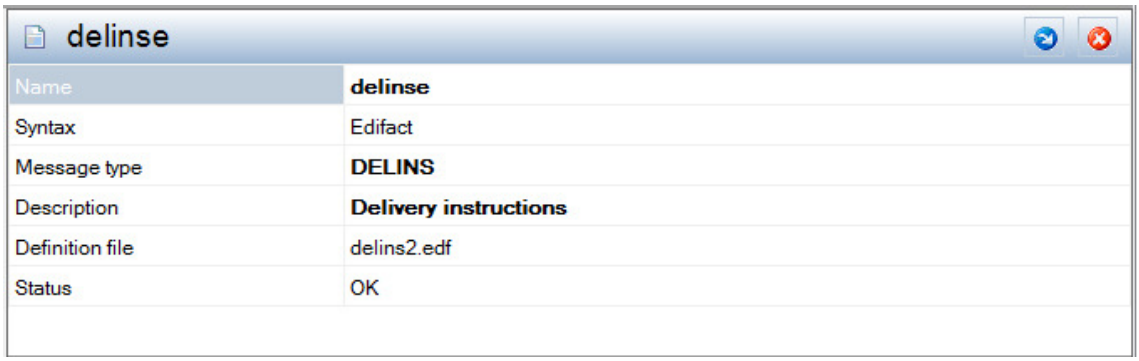
Click the **Edit** button ( ) to edit the currently selected EDI definition. The corresponding node will be selected in the tree view and the list views will change to allow the entity to be edited.

Click the **Delete** button ( ) to delete the currently selected definition.

4.4.6 EDI definition entity

When an EDI definition entity is clicked in the tree view these lists are displayed:

Definition Properties



The list shows the properties of the EDI document definition. The name, syntax, message type and description can be edited directly. The name is used to reference the definition in other parts of the index so you should consider

making it something meaningful. The syntax may be edited by choosing one from a drop-down list, but only when there are no definition fields present.

The status field is read-only and is there to indicate whether the definition is complete. If there are any problems with the definition then the status will not be OK and one or more messages will be displayed here indicating what the errors are.

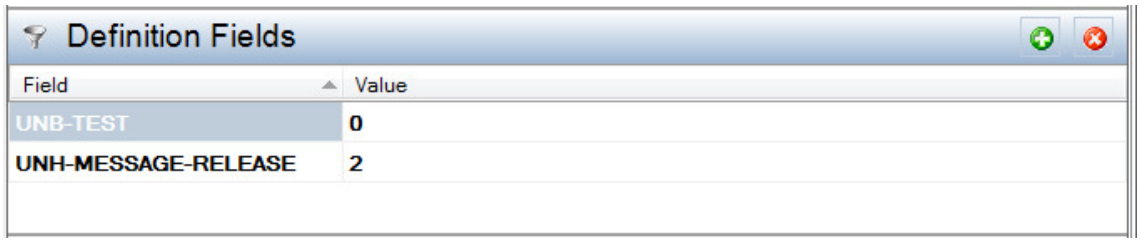
EDI definition files (EDFs) are one of the file types now managed by Xe in its directory structure, so if you create a new definition file it will be created in a location that Xe chooses, and if you want to use an existing definition file, it will need to be imported into a location that Xe chooses.

Click the **Edit** button ( ) to show a menu with options for selecting and manipulating the definition:

- Edit definition – if a definition file has been specified this option opens it in the definition editor.
- Select definition – offers options for specifying the definition file to use:
  - Import definition file – prompts you to browse for an existing definition file to use, which Xe will copy locally.
  - New definition – creates a new EDF; for more details see the section entitled “New EDI Definition”.
  - Create from standard library – creates a new EDF from a definition in the standard library (data dictionary); for more details see the section entitled “New EDI Definition From Data Dictionary”.
  - Use existing definition – prompts you to select a definition already known to Xe; for more details see the section entitled “Select Definition File”.
- Rename file – change the name of the definition file (the location cannot be changed as it is managed by Xe).

Click the **Delete** button ( ) to delete the definition.

Definition Fields



The message type property of the EDI definition is used at runtime to compare with incoming data and determine if there is a match. Definitions can be narrowed down further by specifying one or more definition fields. Each consists of a field identifier, representing a field in a service segment, and a required value. For instance, if a definition includes the first line in the screen shot above, the test field in the UNB segment must have the value “0” if the EDI definition is to match the incoming data.

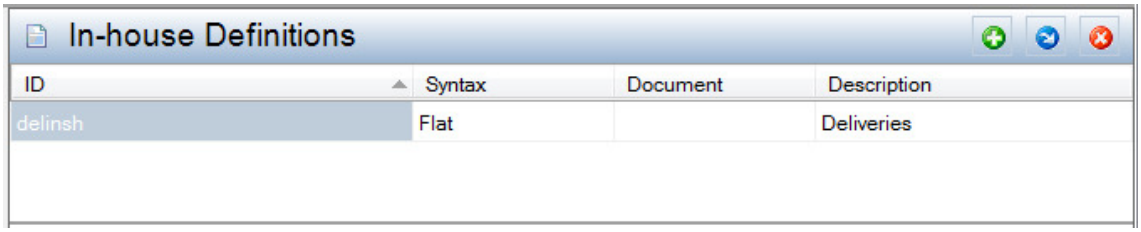
Click the **Add** button ( ) to insert a new definition field, pick the field identifier from the available list and enter the required value.

Click the **Delete** button ( ) to delete the currently selected definition field.

As the fields are specific to each EDI syntax, the syntax cannot be changed if any definition fields are present. To change syntax, all fields must first be removed. New definition fields may then be added that are appropriate to the new syntax.

4.4.7 In-house definitions folder

When the in-house definitions folder is clicked in the tree view this list is displayed:



The list shows the in-house document definitions currently specified in the index. The definition name, syntax and description are shown.

Click the **Add** button ( ) to create a new in-house definition. A new node will be added to the tree view and the list views will change to allow the new entity to be edited.

Click the **Edit** button ( ) to edit the currently selected in-house definition. The corresponding node will be selected in the tree view and the list views will change to allow the entity to be edited.

Click the **Delete** button ( ) to delete the currently selected definition.

4.4.8 In-house definition entity

When an in-house definition entity is clicked in the tree view this list is displayed:

The list shows the properties of the in-house document definition. The name, syntax and description can be edited directly. The name is used to reference the definition in other parts of the index so you should consider making it something meaningful.

The status field is read-only and is there to indicate whether the definition is complete. If there are any problems with the definition then the status will not be OK and one or more messages will be displayed here indicating what the errors are.

In-house definition files (HDFs) are one of the file types now managed by Xe in its directory structure, so if you create a new definition file it will be created in a location that Xe chooses, and if you want to use an existing definition file, it will need to be imported into a location that Xe chooses.

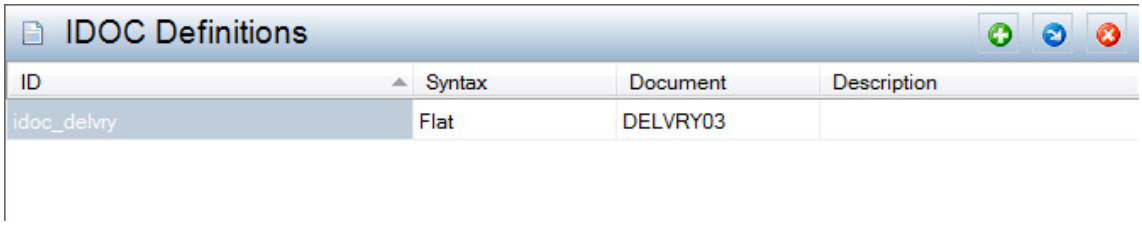
Click the **Edit** button ( ) to show a menu with options for selecting and manipulating the definition:

- Edit definition – if a definition file has been specified this option opens it in the definition editor.
- Select definition – offers options for specifying the definition file to use:
    - Import definition file – prompts you to browse for an existing definition file to use, which Xe will copy locally.
    - New definition – creates a new HDF; for more details see the section entitled “New In-house Definition”.
    - Create from standard library – creates a new HDF from a definition in the standard library (data dictionary); for more details see the section entitled “New In-house Definition From Data Dictionary”.
    - Use existing definition – prompts you to select a definition already known to Xe; for more details see the section entitled “Select Definition File”.
- Rename file – change the name of the definition file (the location cannot be changed as it is managed by Xe).

Click the **Delete** button ( ) to delete the definition.

4.4.9 IDOC definitions folder

When the IDOC definitions folder is clicked in the tree view this list is displayed:



The list shows the IDOC document definitions currently specified in the index. The definition name, syntax, IDOC (document) type and description are shown.

Click the **Add** button ( ) to create a new IDOC definition. A new node will be added to the tree view and the list views will change to allow the new entity to be edited.

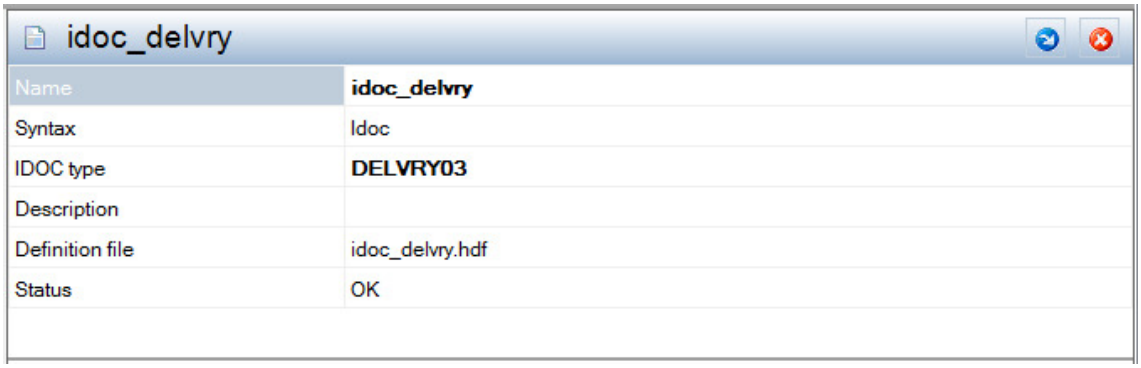
Click the **Edit** button ( ) to edit the currently selected IDOC definition. The corresponding node will be selected in the tree view and the list views will change to allow the entity to be edited.

Click the **Delete** button ( ) to delete the currently selected definition.

4.4.10 IDOC definition entity

When an IDOC definition entity is clicked in the tree view these lists are displayed:

Definition Properties



The list shows the properties of the IDOC definition. The name, IDOC type and description can be edited directly. The name is used to reference the definition in other parts of the index so you should consider making it something meaningful.

The status field is read-only and is there to indicate whether the definition is complete. If there are any problems with the definition then the status will not be OK and one or more messages will be displayed here indicating what the errors are.

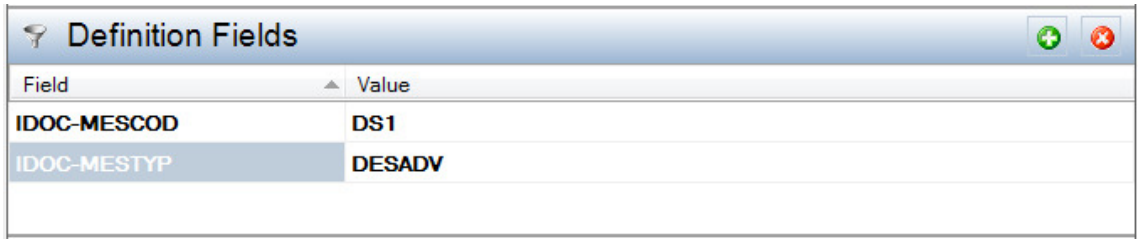
In-house definition files (HDFs) are one of the file types now managed by Xe in its directory structure, so if you create a new definition file it will be created in a location that Xe chooses, and if you want to use an existing definition file, it will need to be imported into a location that Xe chooses.

Click the **Edit** button () to show a menu with options for selecting and manipulating the definition:

- Edit definition – if a definition file has been specified this option opens it in the definition editor.
- Select definition – offers options for specifying the definition file to use:
    - Import definition file – prompts you to browse for an existing definition file to use, which Xe will copy locally.
    - New definition – creates a new HDF; for more details see the section entitled “New In-house Definition”.
    - Use existing definition – prompts you to select a definition already known to Xe; for more details see the section entitled “Select Definition File”.
- Rename file – change the name of the definition file (the location cannot be changed as it is managed by Xe).

Click the **Delete** button () to delete the definition.

Definition Fields



The IDOC type property of the IDOC definition is used at runtime to compare with incoming data and determine if there is a match. Definitions can be narrowed down further by specifying one or more definition fields. Each consists

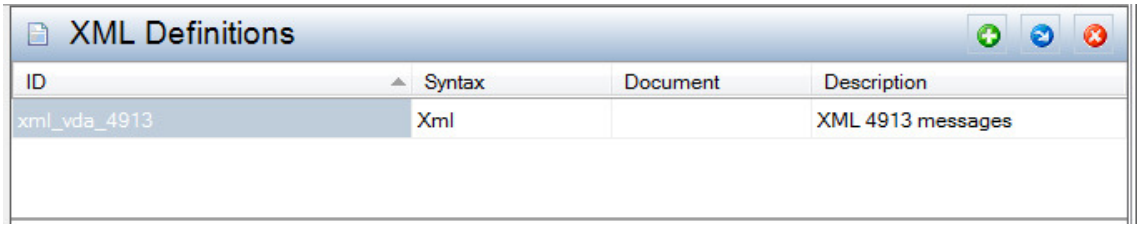
of a field identifier, representing a field in the EDI_DC40 record, and a required value. For instance, if a definition includes the first line in the screen shot above, the message code field in the EDI_DC40 record must have the value “DS1” if the IDOC definition is to match the incoming data.

Click the **Add** button () to insert a new definition field, pick the field identifier from the available list and enter the required value.

Click the **Delete** button () to delete the currently selected definition field.

4.4.11 XML definitions folder

When the XML definitions folder is clicked in the tree view this list is displayed:



The list shows the XML document definitions currently specified in the index. The definition name, syntax and description are shown.

Click the **Add** button () to create a new XML definition. A new node will be added to the tree view and the list views will change to allow the new entity to be edited.

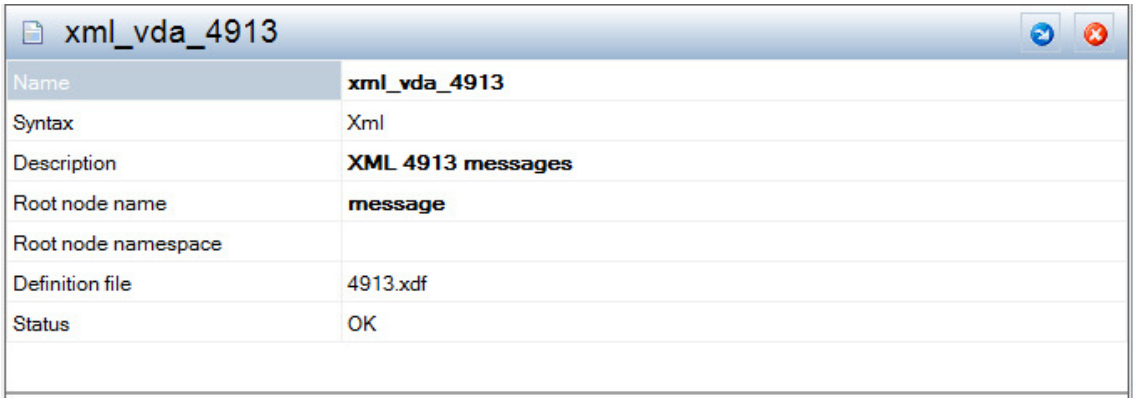
Click the **Edit** button () to edit the currently selected XML definition. The corresponding node will be selected in the tree view and the list views will change to allow the entity to be edited.

Click the **Delete** button () to delete the currently selected definition.

4.4.12 XML definition entity

When an XML definition entity is clicked in the tree view these lists are displayed:

Definition Properties



The list shows the properties of the XML definition. The name, root node name and namespace, and description can be edited directly. The name is used to reference the definition in other parts of the index so you should consider making it something meaningful. The root node name and namespace are compared with source data to identify which definition matches a source file.

The status field is read-only and is there to indicate whether the definition is complete. If there are any problems with the definition then the status will not be OK and one or more messages will be displayed here indicating what the errors are.

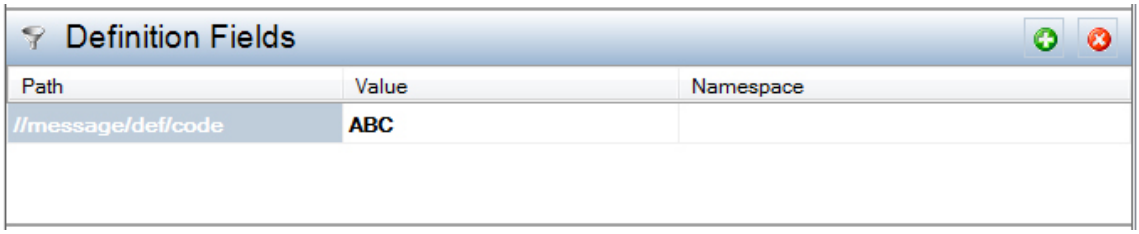
XML definition files (XDFs) are one of the file types now managed by Xe in its directory structure, so if you create a new definition file it will be created in a location that Xe chooses, and if you want to use an existing definition file, it will need to be imported into a location that Xe chooses.

Click the **Edit** button ( ) to show a menu with options for selecting and manipulating the definition:

- Edit definition – if a definition file has been specified this option opens it in the definition editor.
- Select definition – offers options for specifying the definition file to use:
  - Import definition file – prompts you to browse for an existing definition file to use, which Xe will copy locally.
  - New definition – creates a new empty XDF.
  - Create from XML instance – prompts you to browse for an XML file from which the definition will be deduced.
  - Use existing definition – prompts you to select a definition already known to Xe; for more details see the section entitled “Select Definition File”.
- Rename file – change the name of the definition file (the location cannot be changed as it is managed by Xe).

Click the **Delete** button ( ) to delete the definition.

Definition Fields



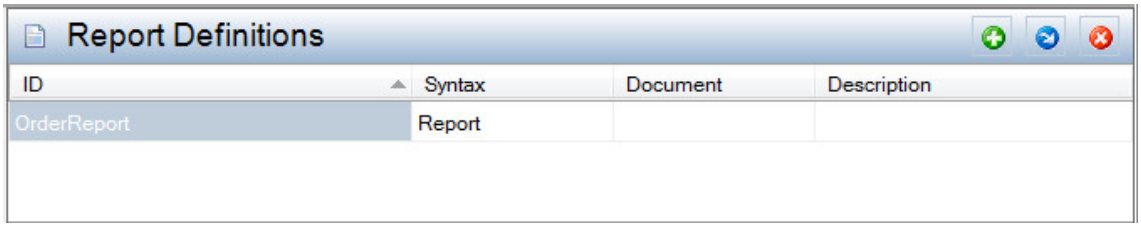
The root node and namespace properties of the XML definition are used at runtime to compare with incoming data and determine if there is a match. Definitions can be narrowed down further by specifying one or more definition fields. Each consists of a full path identifier, representing an element or attribute value in the XML document, and a required value. For instance, if a definition includes the first line in the screen shot above, the code element (child of the def element, child of the root message element) must have the value “ABC” if the XML definition is to match the incoming data.

Click the **Add** button ( ) to insert a new definition field, enter the path of the element or attribute and the required value.

Click the **Delete** button ( ) to delete the currently selected definition field.

4.4.13 Report definitions folder

When the report definitions folder is clicked in the tree view this list is displayed:



The list shows the report definitions currently specified in the index. The definition name, syntax and description are shown.

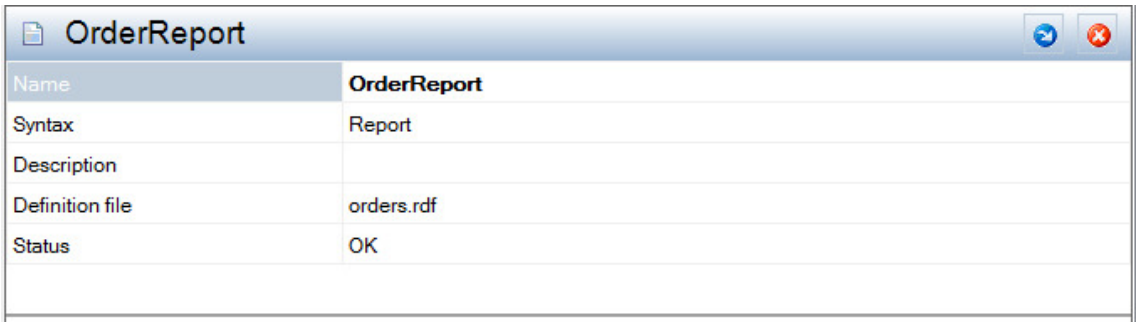
Click the **Add** button () to create a new report definition. A new node will be added to the tree view and the list views will change to allow the new entity to be edited.

Click the **Edit** button () to edit the currently selected report definition. The corresponding node will be selected in the tree view and the list views will change to allow the entity to be edited.

Click the **Delete** button () to delete the currently selected definition.

4.4.14 Report definition entity

When a report definition entity is clicked in the tree view this list is displayed:



The list shows the properties of the report definition. The name and description can be edited directly. The name is used to reference the definition in other parts of the index so you should consider making it something meaningful.

The status field is read-only and is there to indicate whether the definition is complete. If there are any problems with the definition then the status will not be OK and one or more messages will be displayed here indicating what the errors are.

Report definition files (RDFs) are one of the file types now managed by Xe in its directory structure, so if you create a new definition file it will be created in a location that Xe chooses, and if you want to use an existing definition file, it will need to be imported into a location that Xe chooses.

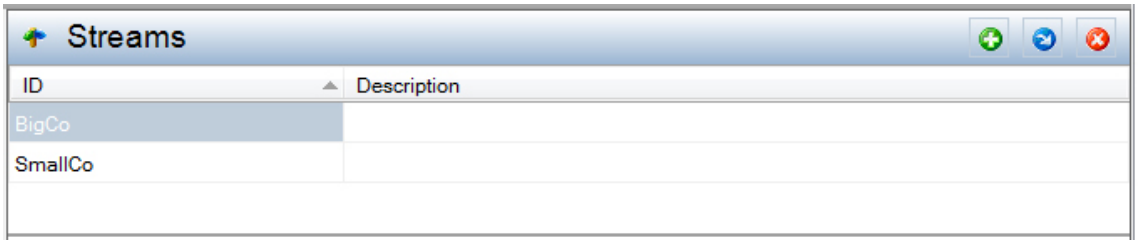
Click the **Edit** button () to show a menu with options for selecting and manipulating the definition:

- Edit definition – if a definition file has been specified this option opens it in the report designer.
- Select definition – offers options for specifying the definition file to use:
    - Import definition file – prompts you to browse for an existing definition file to use, which Xe will copy locally.
    - New definition – creates a new empty RDF and opens it in the report designer.
    - Use existing definition – prompts you to select a definition already known to Xe; for more details see the section entitled “Select Definition File”.
- Rename file – change the name of the definition file (the location cannot be changed as it is managed by Xe).

Click the **Delete** button () to delete the definition.

4.4.15 Streams folder

When the streams folder is clicked in the tree view this list is displayed:



The list shows the streams currently specified in the index. The stream name and description are shown.

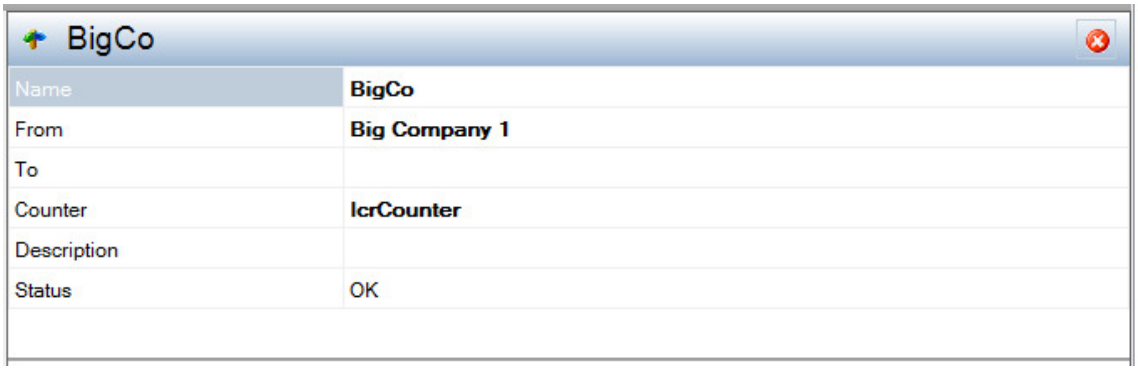
Click the **Add** button () to create a new stream. A new node will be added to the tree view and the list views will change to allow the new entity to be edited.

Click the **Edit** button () to edit the currently selected stream. The corresponding node will be selected in the tree view and the list views will change to allow the entity to be edited.

Click the **Delete** button () to delete the currently selected stream.

4.4.16 Stream entity

When a stream definition entity is clicked in the tree view this list is displayed:



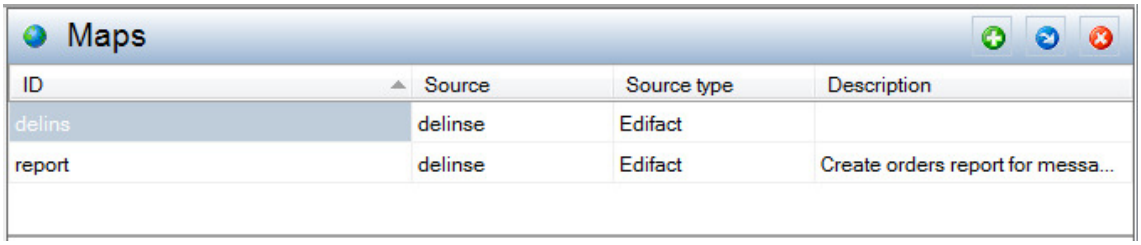
The list shows the properties of the stream. The name and description can be edited directly. The from and to fields are editable by drop-down list, which are used to pick a sender and receiver company already defined in the index. The counter field is also editable by drop-down; select any counter already defined in the index.

The status field is read-only and is there to indicate whether the stream definition is complete. If there are any problems with the definition then the status will not be OK and one or more messages will be displayed here indicating what the errors are.

Click the **Delete** button () to delete the definition.

4.4.17 Maps folder

When the maps folder is clicked in the tree view this list is displayed:



The list shows the maps currently specified in the index. The map name, source definition and syntax, and map description are shown. At this level in the index editor a map is equivalent to a transformation entity in the index file. One or more map processes may be defined within a transformation, as previously described.

Click the **Add** button ( ) to create a new map. A new node will be added to the tree view and the list views will change to allow the new entity to be edited.

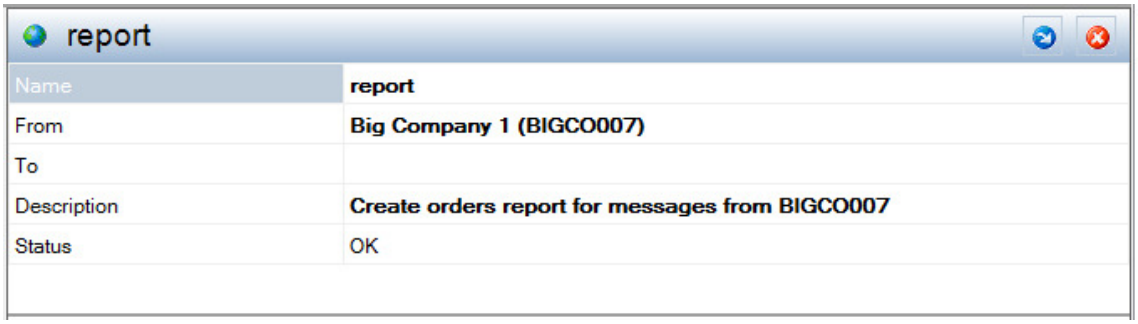
Click the **Edit** button ( ) to edit the currently selected map. The corresponding node will be selected in the tree view and the list views will change to allow the entity to be edited.

Click the **Delete** button ( ) to delete the currently selected map.

4.4.18 Map entity

When a map entity is clicked in the tree view these lists are displayed:

Map Properties



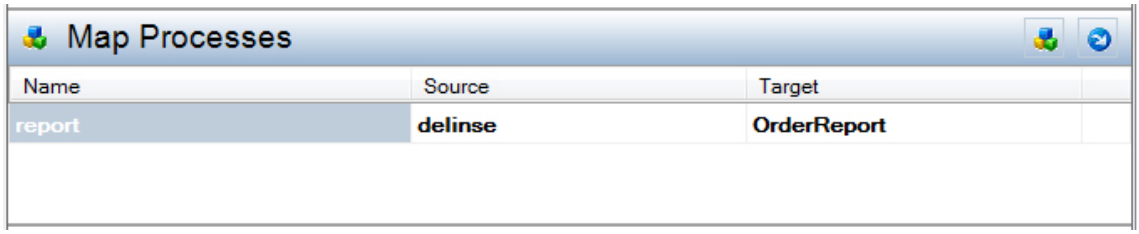
The list shows the properties of the map (transformation). The name and description can be edited directly. The name is used to reference the definition in other parts of the index so you should consider making it something meaningful. The sender and receiver company or EDI code ('from' and to 'fields') can be selected using drop-down lists.

The status field is read-only and is there to indicate whether the map definition is complete. If there are any problems with the map then the status will not be OK and one or more messages will be displayed here indicating what the errors are.

Click the **Edit** button ( ) to show a dialog for editing additional properties of the transformation (see "Edit transformation").

Click the **Delete** button ( ) to delete the definition.

Map Processes

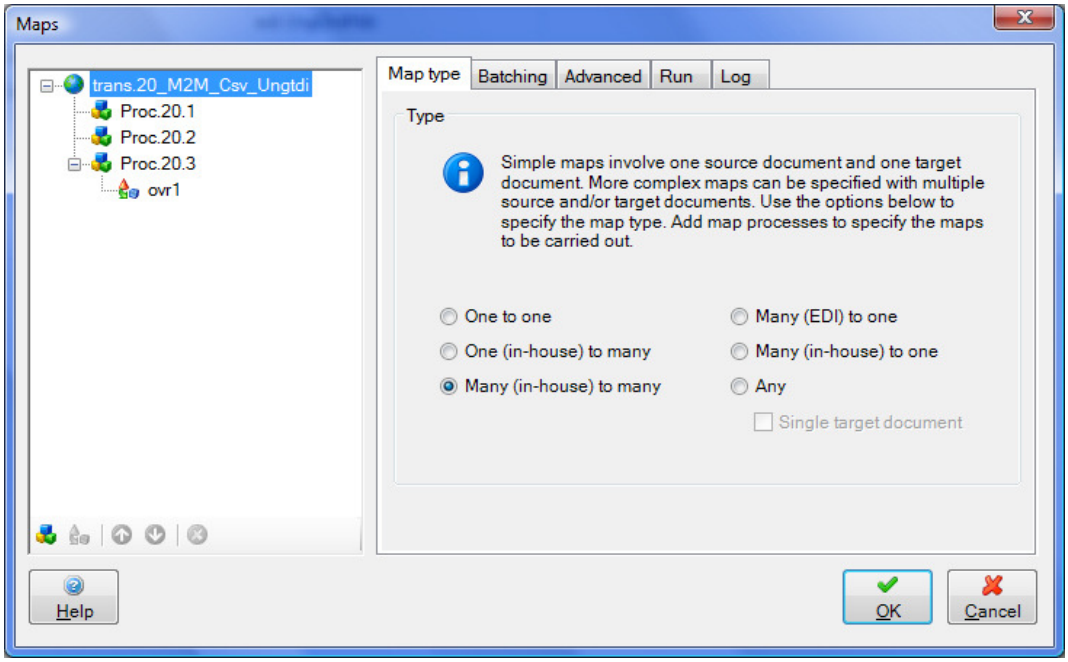


The list shows the map processes associated with the map (transformation). There is always at least one process. If there is only one process then the source and target document definitions can be edited using drop-down lists. More map processes can be added, and processes edited when there are more than one, using an edit dialog. Click the **Edit** button ( ) to show the dialog (see "Edit transformation").

Click the **Open mapper** button ( ) to edit the currently selected map process in the mapper view.

4.4.19 Edit transformation

The following dialog is used to edit advanced properties of transformations and also to configure multiple map processes (and/or overrides) to be used in a transformation (for more information about the relationship between a transformation, process and override see the section entitled “Index structure”).



The dialog is in two sections:

- The left had side consists of a tree view showing the transformation and other entities.
- The right hand side consists of a number of tabs displaying properties according to what is selected in the tree view. Each tab is described in its own section below.

The tree view on the left-hand side shows:

- The transformation currently being edited (as the root node)
- The processes defined for that transformation (as child nodes of the root node)
- The overrides defined for each process (as child nodes of the process nodes)

(For more information about the relationship between a transformation, process and override see the section entitled “Index structure”).

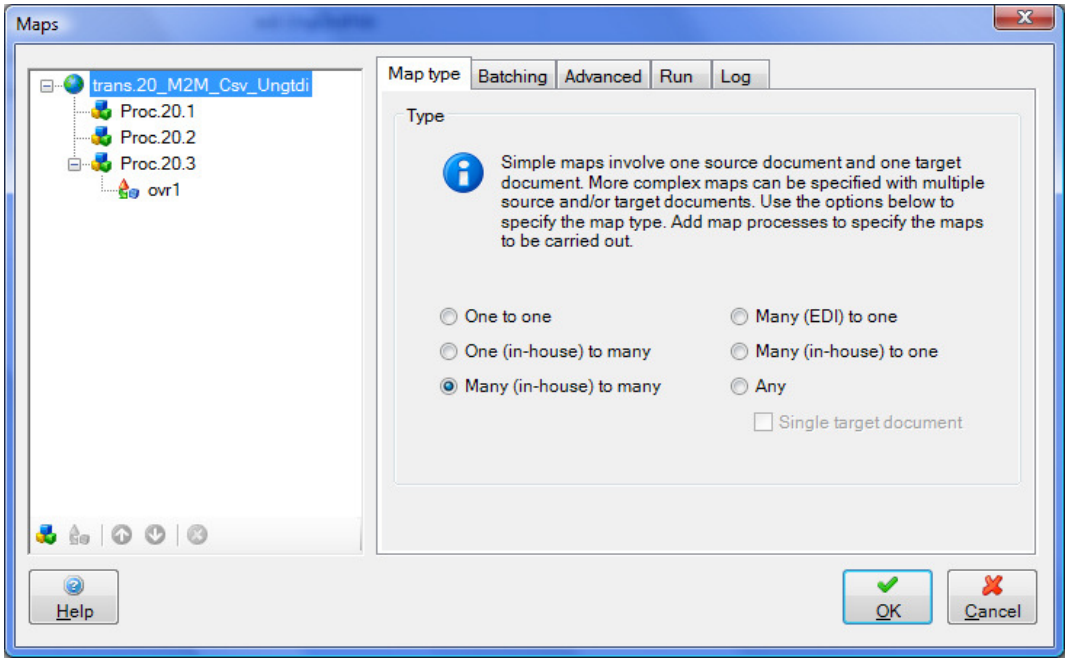
Map processes and overrides can be added, moved and deleted using the buttons below the tree view:

- The new process button ( ) can be used to add a new map process to the transformation.
- The new override button ( ) can be used to add a new override to the selected map process.
- The up button ( ) can be used to move the selected process or override up in the tree if it has preceding sibling nodes.

- The down button ( ) can be used to move the selected process or override down in the tree if it has following sibling nodes.
- The delete button ( ) can be used to remove the selected process or override.

4.4.19.1 Edit transformation – map type

The **Map type** property page is available when the transformation is selected in the dialog's tree view.

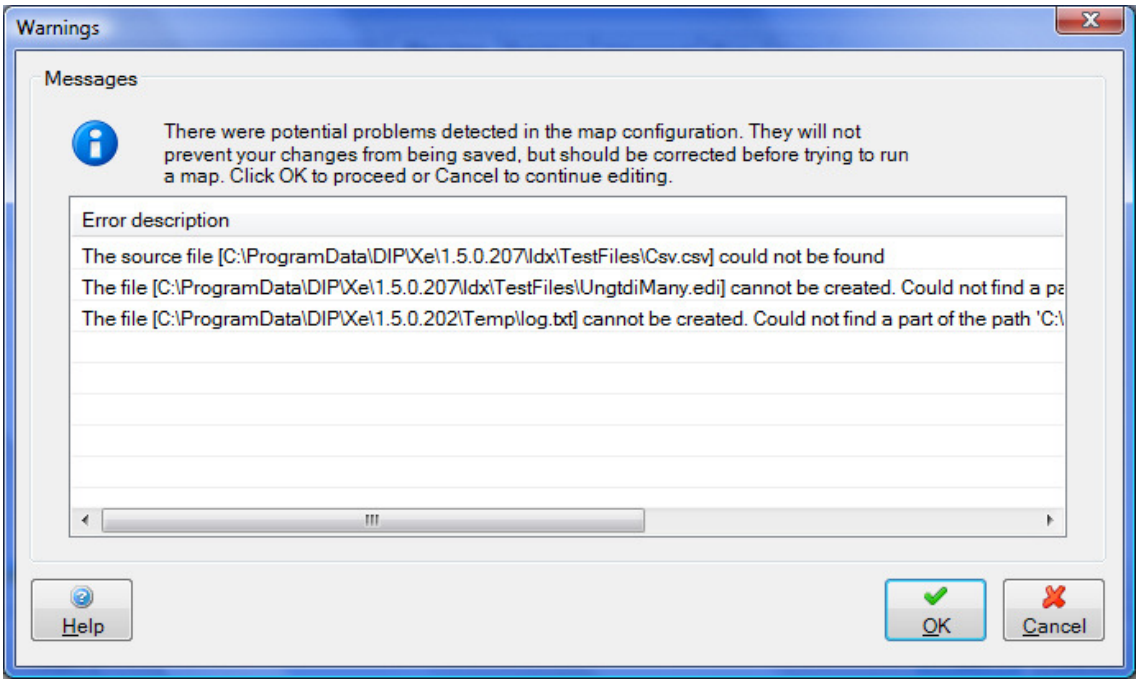


This page displays map type options. The map type determines which combinations of source and target document are permitted. The available types are:

- **One to one** – the transformation represents a single map process from one source to one target document. Overrides are permitted if the source syntax type in-house. Typically a single one-to-one transformation would be specified to process a whole source file containing in-house, XML or IDOC data. If the input file contains EDI messages, the transformation will b executed each time the relevant EDI message is encountered, and other transformations may be performed to map different EDI messages in the same file.
- **Many (EDI) to one** – the transformation takes data from multiple EDI documents and produces one target document. Typically, this type of transformation would be used to map from a Tradacoms file containing many messages (for instance, ORDHDR [1], ORDERS [*], ORDTLR [1]) to a single target. One map process is defined for each EDI message.
- **One (in-house) to many** – the transformation takes data from a single in-house document and produces multiple output documents. Typically, this type of transformation would be used to map from an in-house file to a Tradacoms file to containing many messages (for instance, ORDHDR [1], ORDERS [*], ORDTLR [1]). One map process is defined for each EDI message. There is only one in-house definition (one HDF) and Xe knows when to start a new map by looking for a given record specified against the process (the trigger record) in the source document. A process is therefore uniquely identified with a trigger record.

- **Many (in-house) to one** – the transformation takes data from multiple in-house documents in a single file and produces one target document. In this case, there are multiple HDFs defined for the source, each represents a part of the source file (an in-house ‘document’). Xe knows when to load a new HDF and start a new map by looking for a given record specified against the process (the trigger record) in the source document. A process is therefore uniquely identified with a trigger record.
- **Many (in-house) to many** – the transformation takes data from multiple in-house documents in a single file and produces multiple target documents (one for each source document). In this case, there are multiple HDFs defined for the source, each represents a part of the source file (an in-house ‘document’). Xe knows when to load a new HDF and start a new map by looking for a given record specified against the process (the trigger record) in the source document. A process is therefore uniquely identified with a trigger record.
- **Any** – the combination of source and target documents is not checked so you can include different combinations of maps not covered in the foregoing. For example, an XML document contains the data for a Tradacoms file: choose the **Any** option and specify separate map processes for the header, body and trailer messages. When the **Any** option is selected the **Single target document** check box is enabled. If checkd this causes all of the output to be written to the same document.

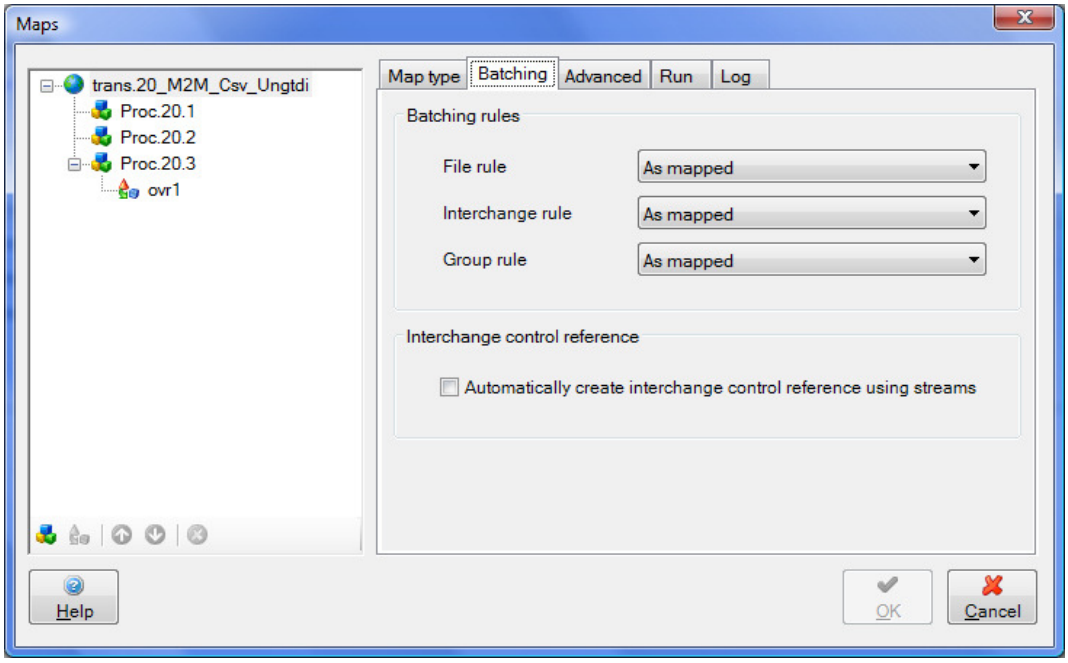
One map (transformation) type must be selected; the default is ‘one to one’. The configured processes and overrides will be evaluated against the selected type when the dialog is dismissed and you will be warned if there are problems, using the following dialog:



For more details about using this dialog to specify the map processes in a transformation see the section entitled “Edit transformation”.

4.4.19.2 Edit transformation – batching

The **Batching** property page is available when the transformation is selected in the dialog’s tree view.



The batching feature can be used to determine how output documents are organised in files. Typically it will be used to specify how EDI messages are organised into interchanges and how interchanges are organized into files. This is specified using three levels of rules:

- **File rule** – specifies when Xe will create a new output file.
- **Interchange rule** – specifies when Xe will create a new EDI interchange.
- **Group rule** – specifies when Xe will create a new EDI functional group.

If all three rules have their default value ('as mapped') then the maps are allowed to determine the batching. For instance, a new interchange is created whenever the relevant segment (UNB, ISA, STX &c) is mapped.

Select non-default values using the drop-down lists to alter the output. For instance:

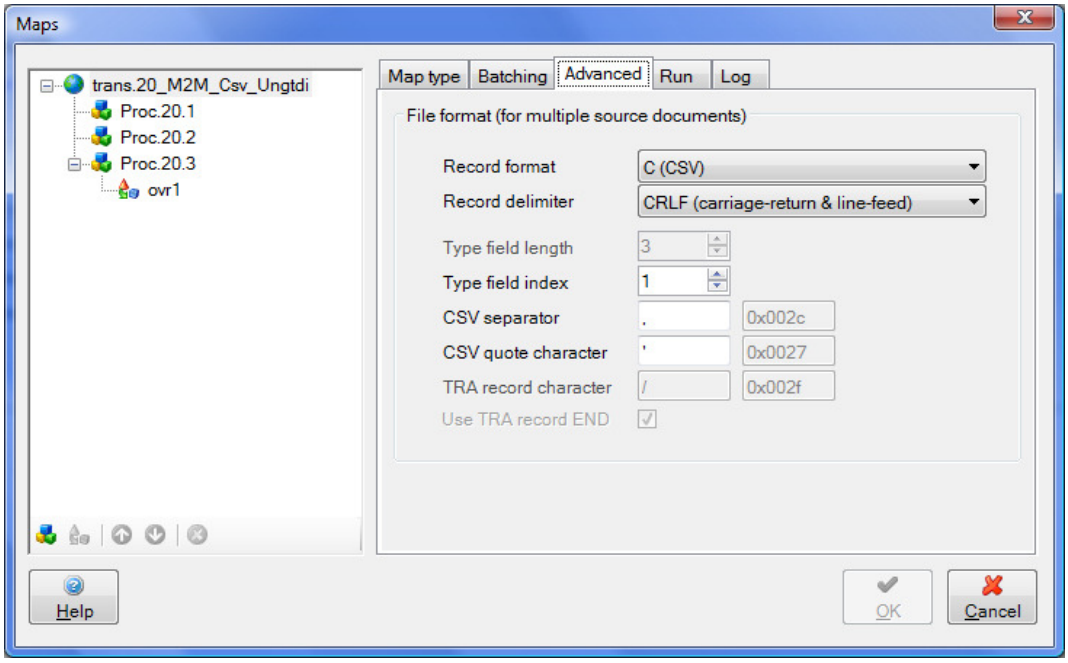
- A **File rule** of 'one per interchange' will cause Xe to output a new file for each interchange it writes.
- An **Interchange rule** of 'automatic' will cause Xe to output as few interchanges as possible by grouping suitable messages into existing interchanges rather than writing new ones.
- A **Group rule** of 'one per message type' will cause Xe to output only one group for each message type written (all the instances of each message type will be in the same functional group).

This property page can also be used to specify whether interchange control references will automatically be created (overwriting any mapped value) using streams defined in the index. This option should only be checked if a suitable stream has been created.

For more details about using this dialog to specify the map processes in a transformation see the section entitled "Edit transformation".

4.4.19.3 Edit transformation – advanced

The **Advanced** property page is available when the transformation is selected in the dialog's tree view.



When using a map type of 'many (in-house) to one' or 'many (in-house) to many' the source file contains multiple in-house 'documents' each identified to Xe by a trigger record defined against a map process. In order to read in-house files which may have multiple document types in them. Xe needs to be able to look for trigger records when it first starts reading the file. To do so, it needs some information about the format of the file. This property page is used to set the following details, used by Xe to read the in-house file (depending on the in-house syntax in use):

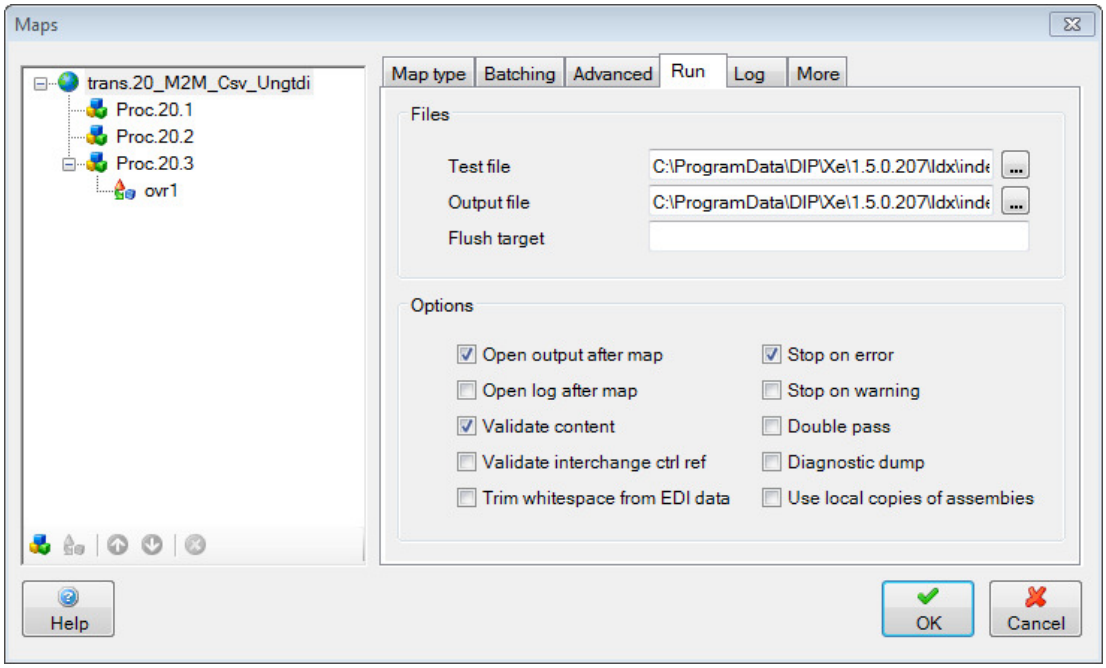
- **Record format** – specify the format of records in the input file. The options are:
  - A – ASCII format (records delimited by CRLF, CR or LF).
  - F – fixed length records.
  - L – variable length records (lengths specified in each record definition in the HDF).
  - C – CSV file.
  - T1 – TRA format with minimal validation of the input file structure.
  - T2 – TRA format with full validation of the input file structure.
- **Delimiter** – for delimited record formats (A and C) specifies the delimiter:
  - **CRLF** – carriage-return and line-feed (0x0d 0x0a).
  - **CR** – carriage-return (0x0d).
  - **LF** – line-feed (0x0a).
- **Type start/index** – specifies the default record type field start position (or index for CSV files).
- **Type length** – specifies the default record type field length.
- **CSV separator** – the character used to separate fields in a CSV file (usually ',' or ';').
- **CSV quote** - the quote character used for quoted fields in a CSV file (usually ' or ").

- **TRA record character** – the character used to indicate the start of a record in a TRA format file (usually '/', as in '/REC-NAME')
- **TRA record END** – flag indicating whether the record end indicator is used in a TRA format file (usually '/END')

For more details about using this dialog to specify the map processes in a transformation see the section entitled "Edit transformation".

4.4.19.4 Edit transformation – run

The **Run** property page is available when the transformation is selected in the dialog's tree view.



These options are used to support running the map from the GUI. The same values can be set in the configuration dialog in the mapper. These values apply to the transformation as a whole, so if, for instance, the test file is changed in the mapper view in respect of one map process, the change will apply to all processes in the transformation. The page is used to set the following details:

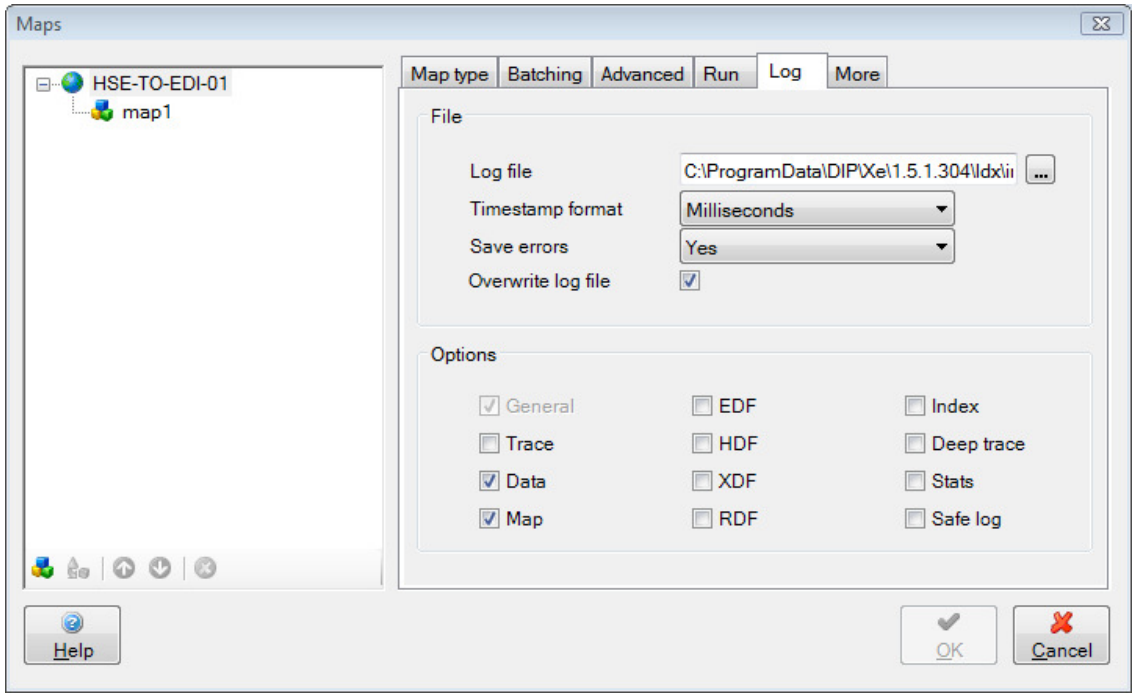
- **Test file** – specify the test (source) file to be used when running a map from the GUI (you will be given the option to import the file, which means that it will be copied locally and managed by Xe henceforth).
- **Output file** – specify the output file location to use when running a map from the GUI.
- **Flush target** – indicate that Xe should perform a partial write of the target document each time it reaches the given node in the target definition (defined by cross-reference for EDF/HDF or full path for XDF). This option will be used only where it is necessary to flush parts of a very large output file to avoid memory problems during mapping.
- **Open output after map** – indicates whether the output file(s) will be opened after the map is run in the GUI.
- **Open log after map** - indicates whether the log will be opened after the map is run in the GUI.
- **Validate content** – indicates whether Xe should validate content and formatting against the definition file.

- **Validate interchange control ref** – indicates whether Xe should attempt to find a stream to use for validating the interchange control reference on an incoming EDI message. This option should be checked only if a suitable stream has been defined in the index.
- **Trim whitespace from EDI data** – indicates whether Xe should trim any whitespace padding from the end of EDI data elements.
- **Stop on error** – indicates whether Xe should stop map processing when a recoverable mapper error is issued.
- **Stop on warning** - indicates whether Xe should stop map processing when a mapper warning is issued.
- **Double pass** – indicates whether Xe should perform a double pass by processing the output from a map as the input of a second map. This option should be checked only if the index is configured for both maps (that is, a transformation is defined that is triggered by the output of the first map).
- **Diagnostic dump** – indicates whether Xe should create files in the same directory as the output file with dumps of the contents of the source and target DOMs before each map is performed.
- **Use local copies of assemblies** – indicates whether Xe should copy map assemblies to a temporary location in its data directory before running the map. If you are working with a remote index that is generating map assemblies on a network drive, you may experience problems if your map requires certain security permissions (because you are using database functions or code snippets that try to write to disk files, for instance). These can be overcome by using local copies of the assemblies.

For more details about using this dialog to specify the map processes in a transformation see the section entitled “Edit transformation”.

4.4.19.5 Edit transformation – log

The **Log** property page is available when the transformation is selected in the dialog’s tree view.



These options are used to support running the map from the GUI. They specify what is to be written to the log and where it is to be created. The same values can be set in the configuration dialog in the mapper. These values apply to the transformation as a whole, so if, for instance, the log file is changed in the mapper view in respect of one map process, the change will apply to all processes in the transformation. The page is used to set the following details:

- **Log file** – specify the log file location to use when running a map from the GUI.
- **Timestamp format** – specify the format of the timestamp that appears at the left hand side of each log line.
- **Save errors** – specify whether to extract any validation errors that are logged and save them for display after a map is run.
- **Overwrite log file** – indicates whether the log file should be overwritten each time the map is run.
- **Log types** – the following log types can be configured:
    - **General** – the highest level of log messages; always written if logging is turned on.
    - **Trace** – log more detailed general messages.
    - **Data** – log each segment/record read from the source and each element/field loaded into/read from the DOM.
    - **Map** – log each data item that is mapped.
    - **EDF** – log detailed messages when loading an EDF.
    - **HDF** – log detailed messages when loading an HDF.
    - **XDF** – log detailed messages when loading an XDF.
    - **RDF** – log detailed messages when loading an RDF.
    - **Index** – log detailed messages when loading a runtime index.
    - **Deep trace** – log very detailed diagnostic messages relating to the reading of data from the source DOM and writing of values to the

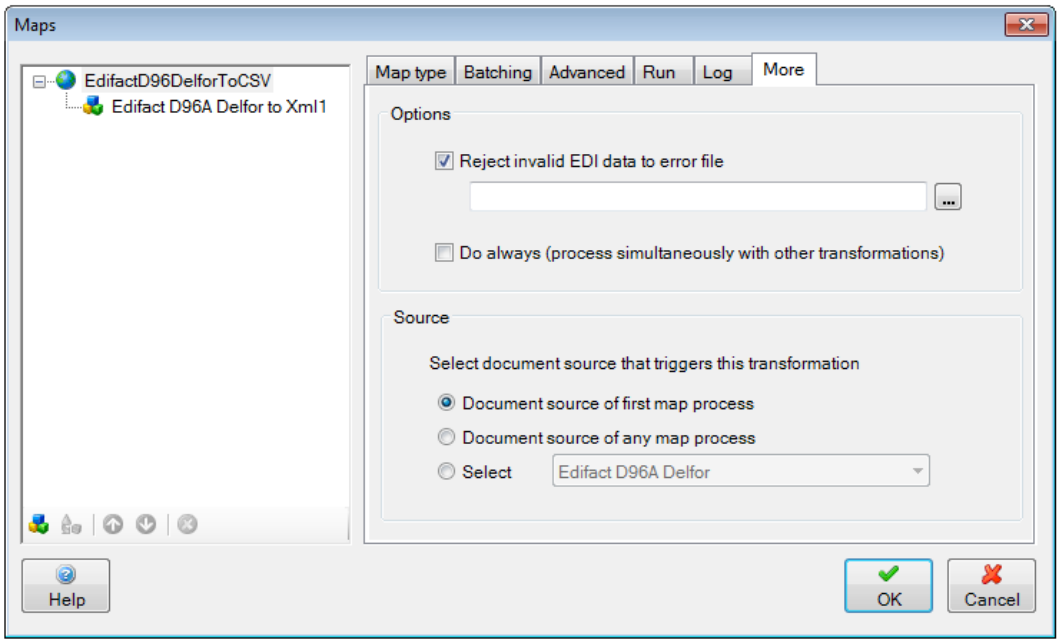
target DOM during mapping (note that deep trace logging slows performance of the mapper dramatically).

- **Stats** – write performance statistics at the end of the log.
- **Safe log** – log file is opened and closed each time a message is written so messages are not lost if there is a fatal error (note that safe logging slows performance of the mapper dramatically).

For more details about using this dialog to specify the map processes in a transformation see the section entitled “Edit transformation”.

4.4.19.6 Edit transformation – more

The **More** property page is available when the transformation is selected in the dialog’s tree view.



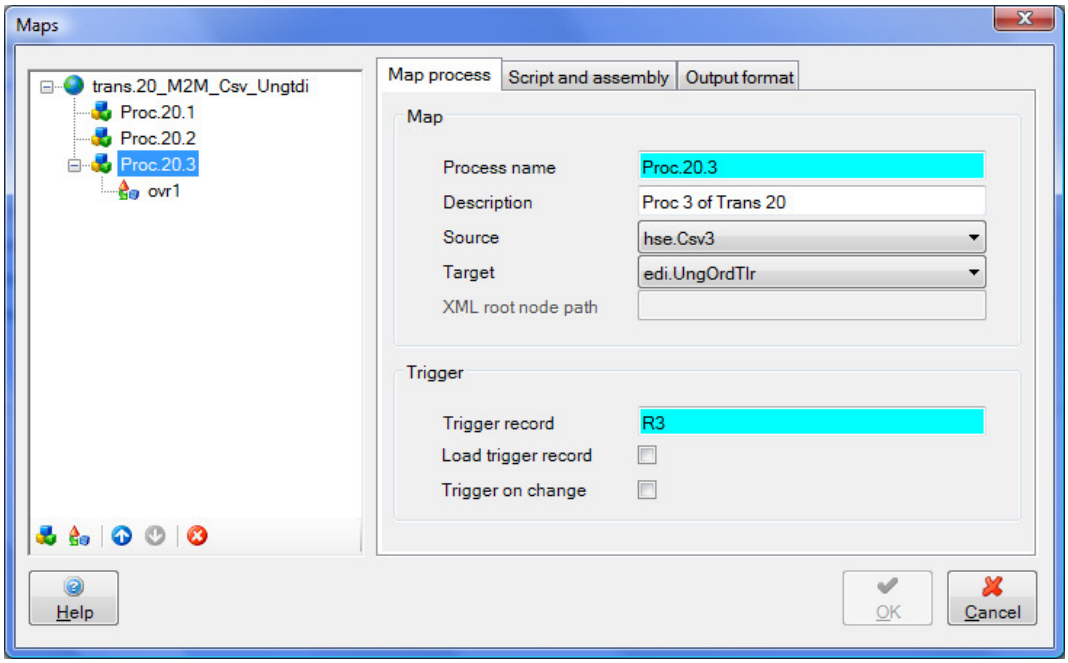
These options are used to support running the map from the GUI. The same values can be set in the configuration dialog in the mapper. These values apply to the transformation as a whole, so if, for instance, the test file is changed in the mapper view in respect of one map process, the change will apply to all processes in the transformation. The page is used to set the following details:

- **Reject invalid EDI data to error file** – specify the log file location to log invalid EDI data to this file.
- **Do always (process simultaneously with other transactions)** – do always and process simultaneously with other transformations.
- **Select document source that triggers this transformation**
  - **Document source of first map process –** this transformation will be triggered based on the match to the document source of first map process.

- **Document source of any map process –** this transformation will be triggered based on the match of any document source from any map processes in this transformation.
- **Select –** this transformation will be triggered based on the document source selected by the user.

4.4.19.7 Edit transformation – map process

The **Map process** property page is available when a process is selected in the dialog's tree view.



The map process page is used to edit the principal details of a single process. The following details are shown:

- **Process name** – choose a name for the process that will distinguish it from others in the same transformation.
- **Description** – optionally include a description with more details about the process.
- **Source** – select the source document definition from the drop-down list.
- **Target** – select the target document definition from the drop-down list.
- **XML root node path** – where the source document is XML you may optionally specify a path to a node in the source that is to be treated as the root node in the mapper. For instance, an XML document has a root node named 'doc', and the root node has multiple child nodes named 'msg'. Enter the path '//doc/msg' in this field and the map will be performed once for each instance of the 'msg' node, treating that node as the root.
- **Trigger record** – if the map specifies multiple processes and the source document type is in-house then a unique trigger record must be provided for each process. The trigger record indicates the record type in the in-house document that triggers the map represented by the process.
- **Load trigger record** – where trigger records are being used as above and there are multiple in-house 'documents' (each process has a

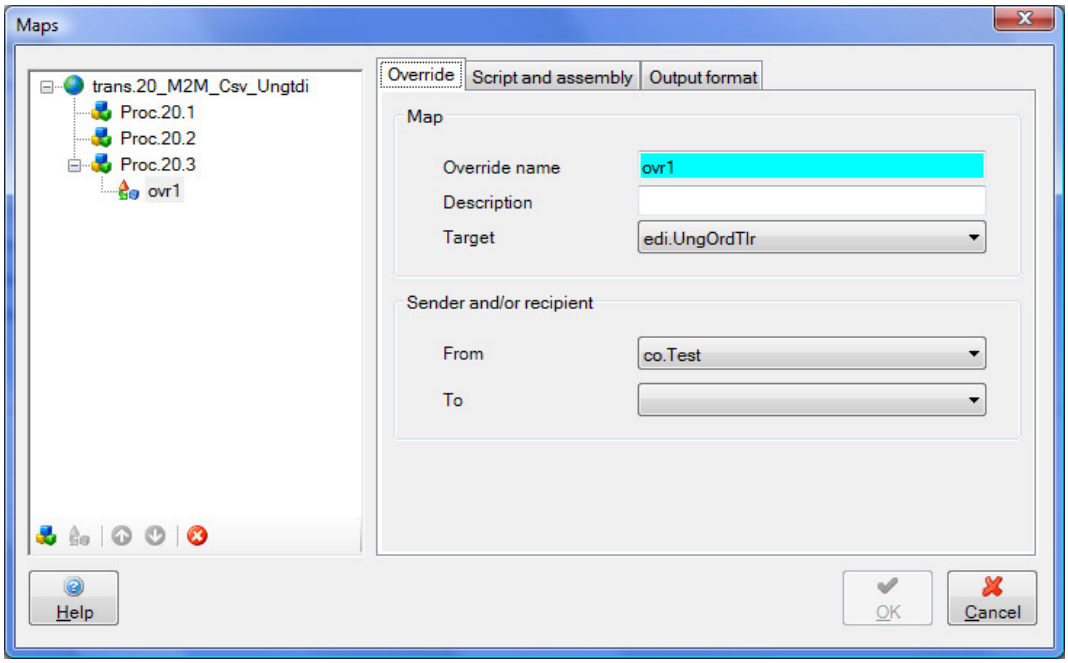
different source, so there are multiple HDFs defining the source file), this flag indicates that the trigger record for the process is to be loaded into the source DOM for the document represented by the current trigger record **and** the DOM for the previous trigger (if there is one).

- **Trigger on change** – where trigger records are being used as above, indicates whether the process should be triggered by each occurrence of the record, or just by distinct occurrences.

For more details about using this dialog to specify the map processes in a transformation see the section entitled “Edit transformation”.

4.4.19.8 Edit transformation – override

The **Override** property page is available when a map process override is selected in the dialog’s tree view.



The override page is used to edit the principal details of a map process override. The following details are shown:

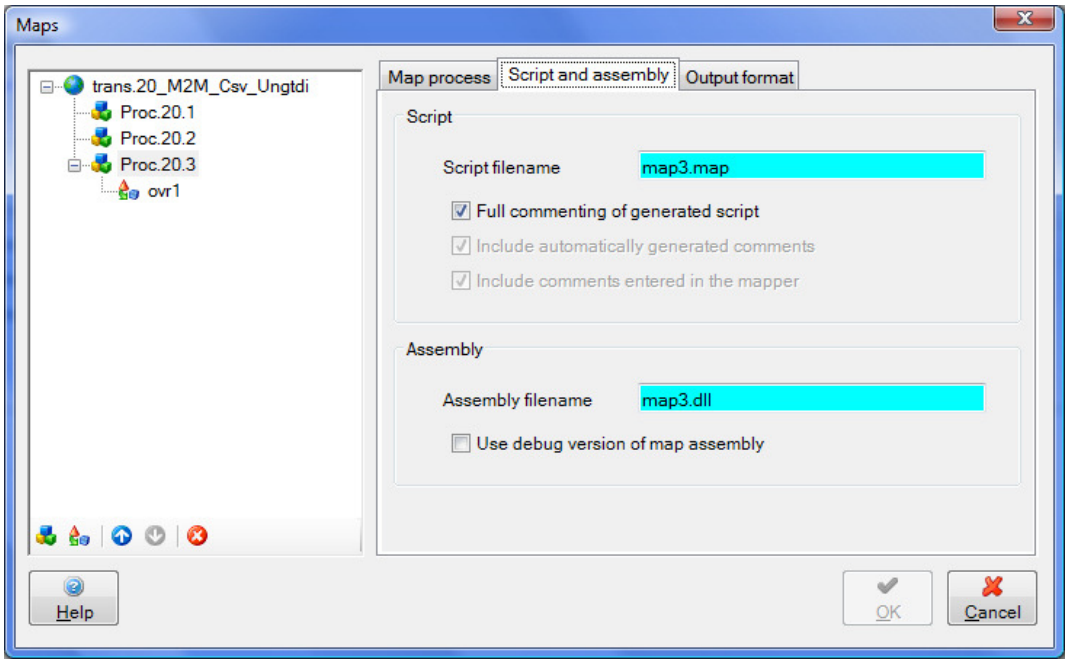
- **Override name** – choose a name for the override that will distinguish it from others in the same transformation.
- **Description** – optionally include a description with more details about the override.
- **Target** – select the target document definition from the drop-down list (the source is the same as for the overridden process).
- **From** – select the sender company or EDI code from the drop-down list. The override map will be performed if the values in the source file indicated by the ORG record in the HDF match the EDI code information selected for this field.
- **To** – select the recipient company or EDI code from the drop-down list. The override map will be performed if the values in the source file indicated by the DST record in the HDF match the EDI code information selected for this field.

Note that at least one of the **From** and **To** fields must be specified.

For more details about using this dialog to specify the map processes in a transformation see the section entitled “Edit transformation”.

4.4.19.9 Edit transformation – script and assembly

The **Script and assembly** property page is available when a map process or override is selected in the dialog’s tree view.



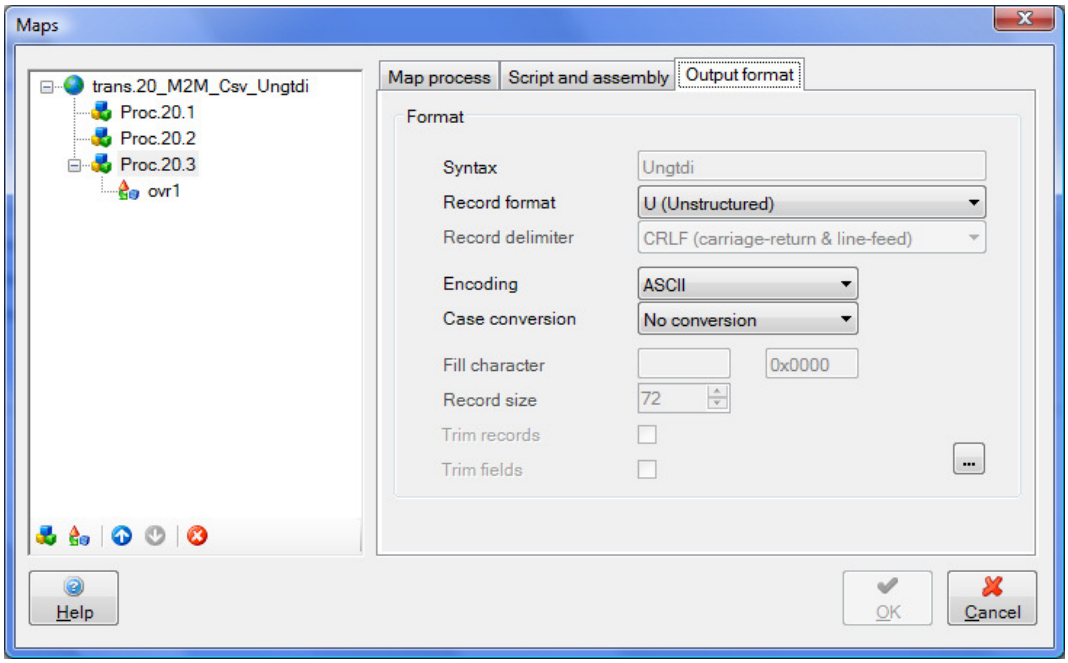
This page is used to specify map script and map assembly options for both map processes and overrides. The following details are shown:

- **Script filename** – specify the filename of the map script (remember that the script file is managed by Xe in a location it chooses so you cannot specify a full path).
- **Commenting of generated script** – indicates what comments will be included in the map script when generated by the GUI. These options are available:
  - **Full commenting** – means a comment banner is written for each loop statement and map statement, including an automatically generated comment and any comment you enter against the map connection in the GUI.
  - **Automatic commenting** – means a comment banner is written for each loop statement and map statement, including automatically generated comments.
  - **Comments entered in the mapper** – means a comment banner is written in respect of comments you enter against map connections in the GUI.
- **Assembly filename** – specify the filename of the map assembly (remember that the assembly file is managed by Xe in a location it chooses so you cannot specify a full path).
- **Use debug version of map assembly** – indicates that the map assembly should be generated in a debug version. The debug version includes extra logging indicating the map script statement being performed at each stage of the mapping process.

For more details about using this dialog to specify the map processes in a transformation see the section entitled “Edit transformation”.

4.4.19.10 Edit transformation – output format

The **output format** property page is available when a map process or override is selected in the dialog’s tree view.



This page is used to specify output formatting options for both map processes and overrides. The following details are shown:

- **Syntax** – a read-only field given by the source document type, it conditions which of the following options are available.
- **Record format** – indicates the format in which data is to be written. Varies according to syntax:
    - **EDI syntaxes (excluding VDA)**:
        - **U (unstructured)** – no formatting is applied.
        - **F (fixed length)** – segments are a fixed length with no delimiters.
        - **A1 (delimited)** – segments are delimited (by CRLF, CR or LF).
        - **A2 (delimited, max len)** - segments are delimited (by CRLF, CR or LF), a maximum length is specified and segments are delimited at this length.
        - **A3 (delimited, max len & fill)** – segments are delimited (by CRLF, CR or LF), a maximum length is specified and segments are delimited at this length and padded to this length.
    - **VDA**
        - **F (fixed length)** – segments are a fixed length with no delimiters.
        - **A1 (delimited)** – segments are delimited (by CRLF, CR or LF).

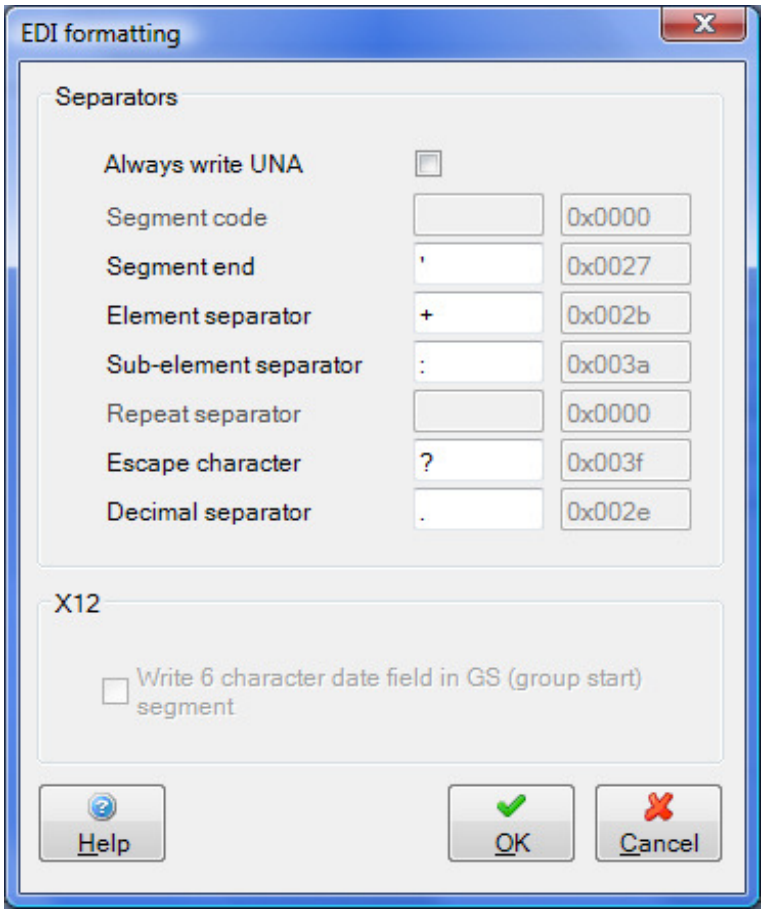
- **Flat**
  - **A (ASCII)** – records are delimited (by CRLF, CR or LF).
  - **F (fixed length)** – records are a fixed length with no delimiters.
  - **L (variable length)** – records are of variable length (specified in the HDF) with no delimiters.
- **CSV**
  - **C (CSV)** – records are character separated values and are delimited (by CRLF, CR or LF).
- **TRA**
  - **T (TRA)** – records are TRA format.
- **IDOC**
  - **A (ASCII)** – records are delimited (by CRLF, CR or LF).
  - **F (fixed length)** – records are a fixed length with no delimiters.
- **XML**
  - **U (unstructured)** – no formatting is applied.
  - **A (delimited)** – elements are delimited (by CRLF, CR or LF) and indented according to hierarchy.
- **Report**
  - **None** – there are no formatting options where the output type is 'report'.

- **Record delimiter** – specifies the delimiter where the record format is delimited (carriage-return & line-feed or carriage-return or line-feed).
- **Encoding** – specifies the output file encoding (ASCII/Latin-1, EBCDIC, Unicode, Unicode big-endian, UTF-8 or UTF-7).
- **Case conversion** – specifies the case conversion to perform on output (to upper, to lower, or no conversion).
- **Fill character** – specifies the fill character to use where the record format is A3 (max record length with fill).
- **Record size** – specifies the record length where the record format is F (fixed), A2 (max record length) or A3 (max record length with fill).
- **Trim records** – indicates whether whitespace should be trimmed from the end of records in delimited flat files
- **Trim fields** – indicates whether empty fields in delimited flat files should be trimmed from the end of records (differs from the previous options in that whitespace at the end of the last field that has some content will not be trimmed).

For more details about using this dialog to specify the map processes in a transformation see the section entitled "Edit transformation".

4.4.19.11 EDI formatting

In addition to the standard output formatting options available (see “Edit transformation – output format”) there are additional options available for EDI target documents. Use the button marked “**…**” on the output formatting page of the transformation dialog to show the options:



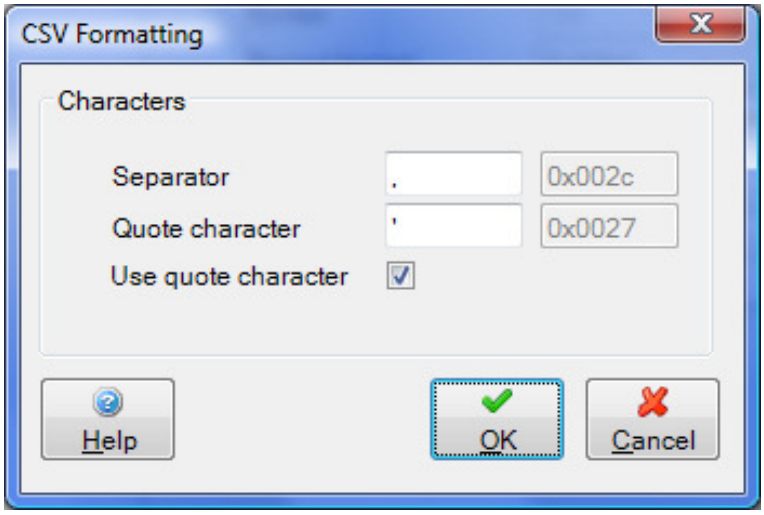
The options available are:

- **Always write UNA (or SCH)** – indicates whether the separator segment should be written even if standard separators are being used (if unchecked then the UNA or SCH will be written only if at least one non-standard separator is used).
- **Segment code** – specifies the segment code separator.
- **Segment end** – specifies the segment end separator.
- **Element separator** – specifies the element separator.
- **Sub-element separator** – specifies the sub-element separator.
- **Repeat separator** – specifies the repeat separator.
- **Escape separator** – specifies the escape separator.
- **Decimal separator** – specifies the decimal separator.
- **Write 6 character date field** – indicates that the date field written to the X12 GS segment should be 6 characters (YYMMDD) instead of the standard 8 (CCYYMMDD).

Note that only those options relevant to the EDI syntax of the target document are enabled.

4.4.19.12 CSV formatting

In addition to the standard output formatting options available (see “Edit transformation – output format”) there are additional options available for CSV target documents. Use the button marked “**…**” on the output formatting page of the transformation dialog to show the options:

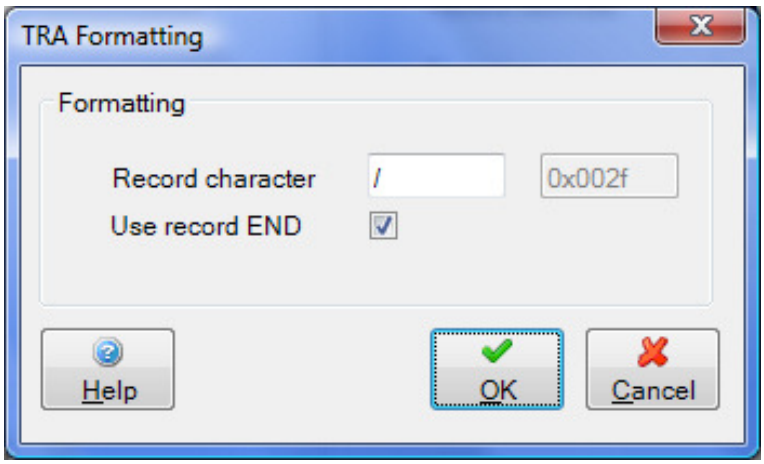


The options available are:

- **Separator** – specifies the CSV separator character to use.
- **Quote character** – specifies the CSV quote character to use.
- **Use quote character** – indicates whether fields in the output should be surrounded by instances of the quote character.

4.4.19.13 TRA formatting

In addition to the standard output formatting options available (see “Edit transformation – output format”) there are additional options available for TRA target documents. Use the button marked “**…**” on the output formatting page of the transformation dialog to show the options:



The options available are:

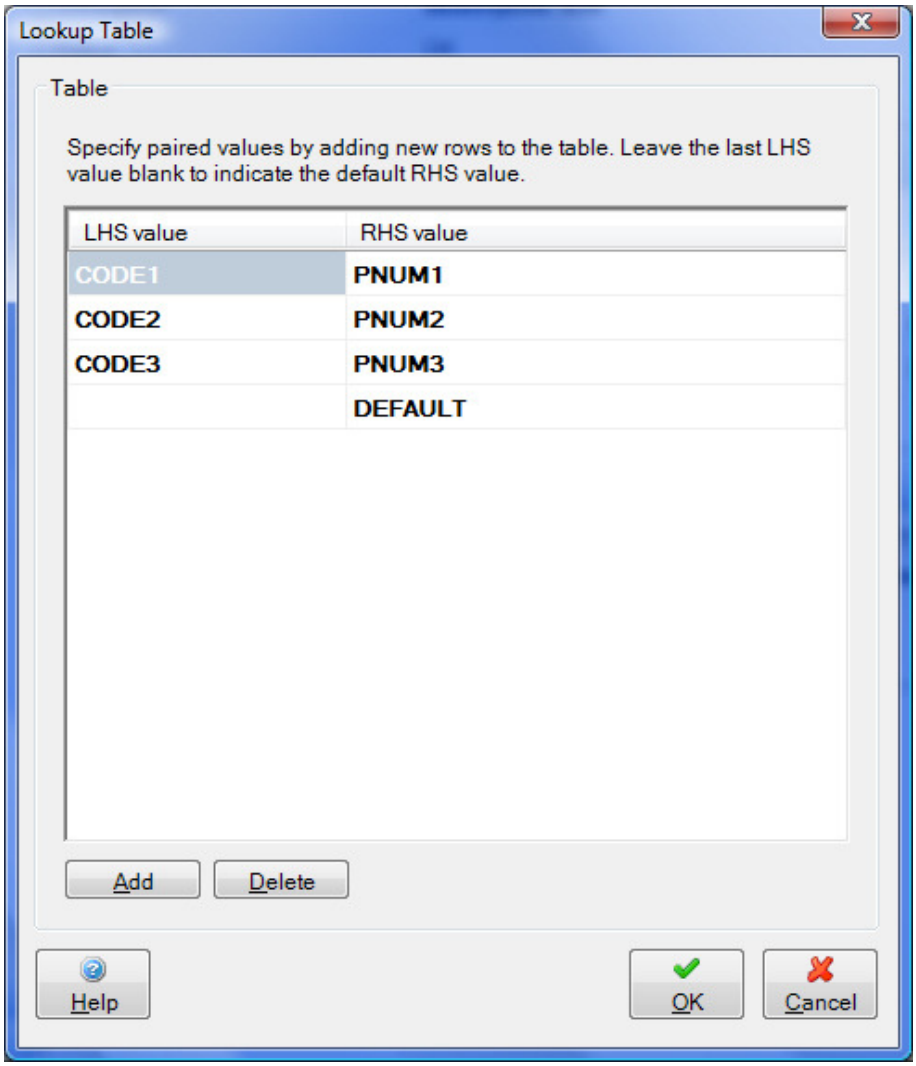
- **Record character** – specify the character that is used to indicate a record name (usually ‘/’ as in ‘/REC-NAME’).
- **Use record END** – Indicate whether the record end indicator is to be written (‘/END’ if the record character is ‘/’).

4.5 Index Editor Dialogs

This section explains the use of various dialogs used in the Index Explorer

4.5.1 Editing Table Files

This dialog is shown when a new table file is created, or when you choose to edit an existing file.



A valid table file consists of pairs of values and is used for lookups; usually the right-hand side value is looked up, keyed by the left-hand side, but the reverse is possible as well. A default lookup value can be specified by leaving the last value for the key blank (for example, if looking up from left to right then the last left-hand side value is left blank to indicate that the corresponding right-hand side value is the default).

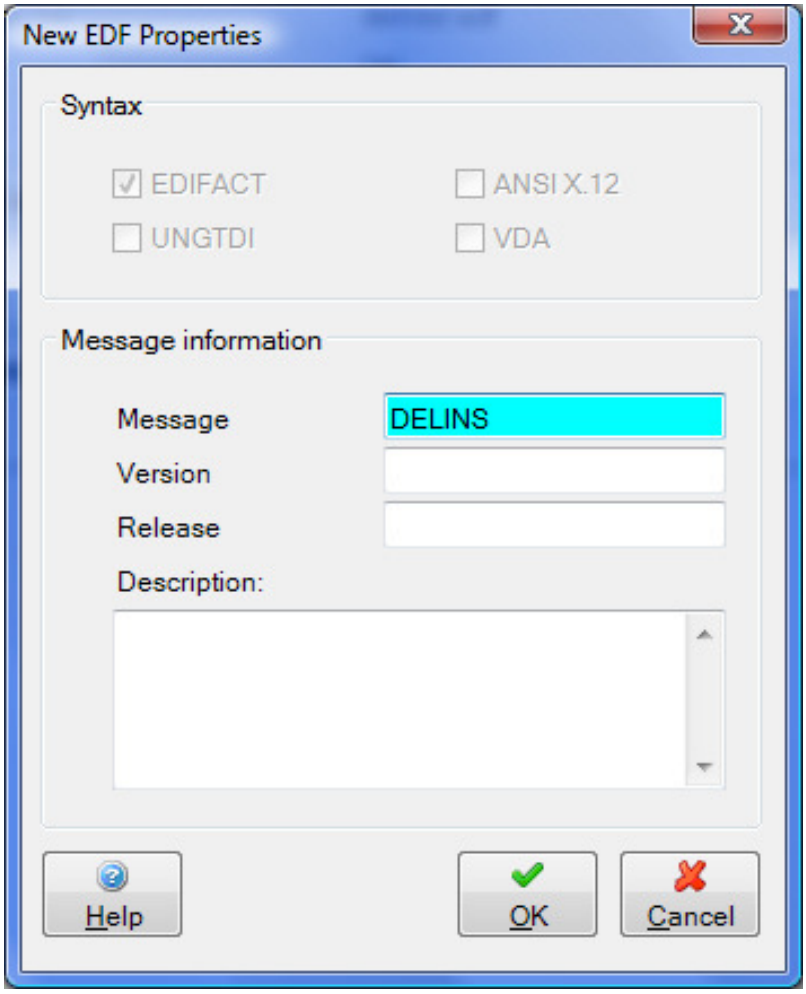
Click the **Add** button to insert a row.

Click the **Delete** button to remove a row.

For rapid keying of values, use the shortcut key **Alt + A** to insert a new row and the left-hand value will be selected for editing automatically. Enter the required value, then press **Tab** to begin editing the right-hand value.

4.5.2 New EDI Definition

This dialog is shown when a new EDI Definition File (EDF) is to be created.

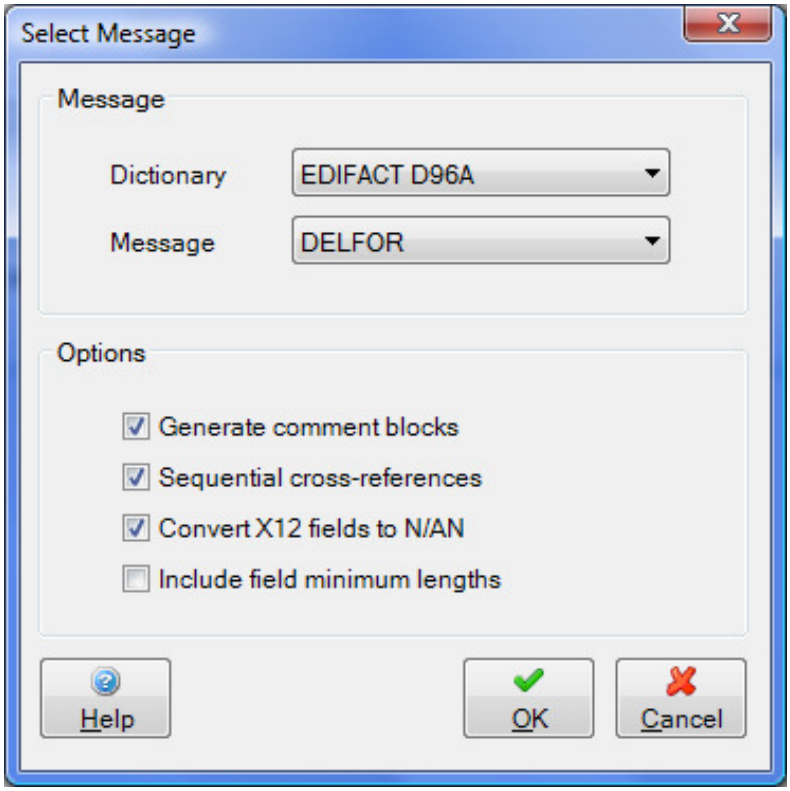


The **Message** (or transaction set) name, must be supplied (for example, DELFOR, 850, 4915). The **Version**, **Release** and **Description** fields are optional and can be filled in to provide more information about the message being defined, if required. These fields are all editable within the definition editor control once the EDF has been created.

When you are satisfied click **OK** to proceed and create the definition, or click **Cancel** to quite without creating an EDF.

4.5.3 New EDI Definition From Data Dictionary

This dialog is shown when a new EDI Definition File (EDF) is to be created from a definition in the data dictionary.



Use the drop-down list to select a **Dictionary** from which the definition is to be extracted. Use the **Message** drop-down to select the message to extract (the list is populated according to what messages are available in the selected dictionary).

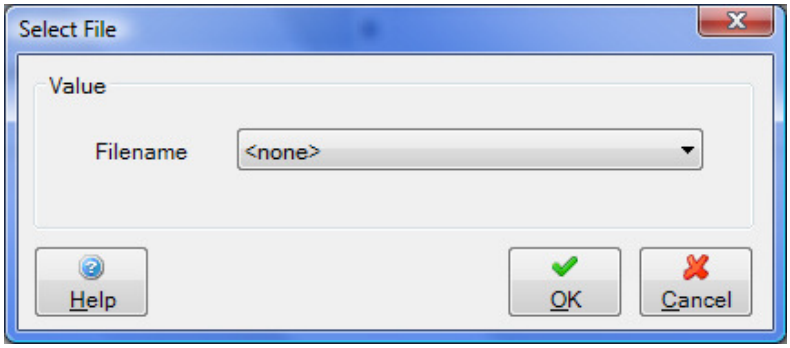
The following options are available, which affect the format of the generated EDF:

- The **Generate comment blocks** option allows you to specify whether the generated definition will contain comments for each record in a preceding block.
- The **Sequential cross-references** option allows you to specify whether the element cross-references generated are strictly sequential (for example; RFF00510, RFF00520). The alternative is that they are based on the occurrence number of the segment (for example; RFF020010, RFF020020 for the second instance of RFF in the definition).
- The **Convert X12 fields to N/AN** option allows you to specify that ID, DT and TM fields in the data dictionary are converted to A or AN fields in the EDF.
- The **Include field minimum lengths** option allows you to specify whether minimum field lengths are included in the EDF if present in the data dictionary definition.

When you are satisfied click **OK** to proceed and create the definition, or click **Cancel** to quite without creating an EDF.

4.5.4 Select Definition File

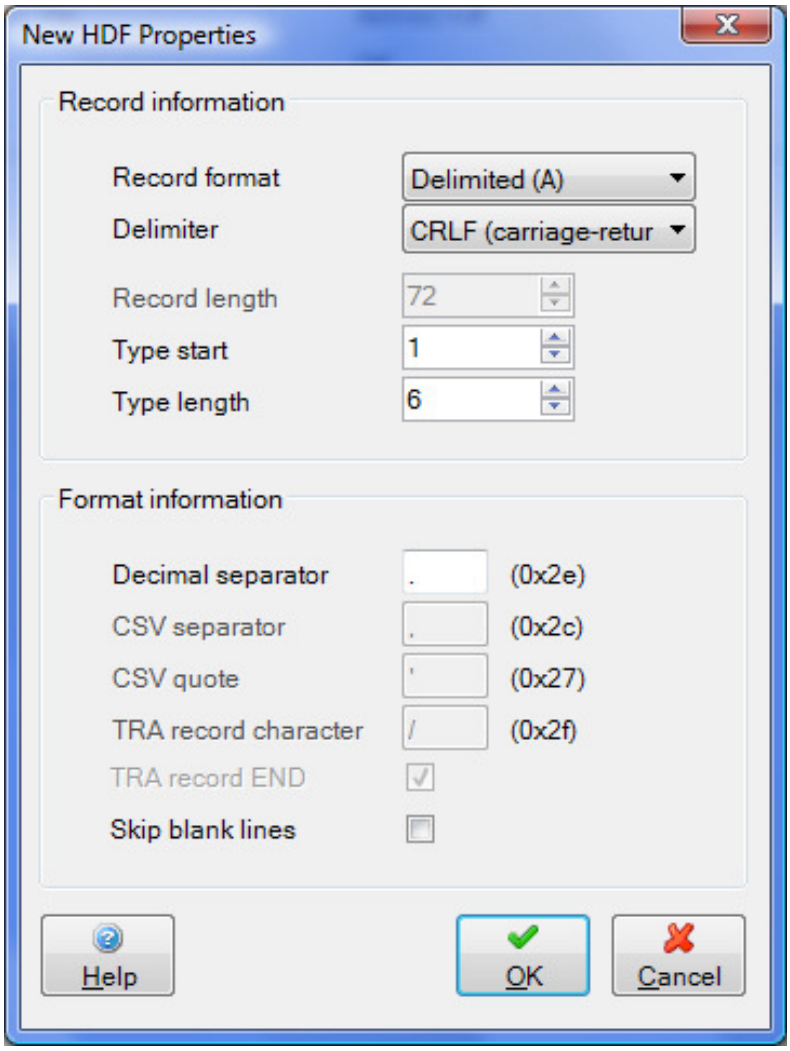
This dialog is shown to allow you to select an existing definition file already used by an index entity in Xe



Select a **Filename** from the drop-down list and click **OK** to proceed, or click **Cancel** to close the dialog without selecting a file.

4.5.5 New In-house Definition

This dialog is shown when a new In-House Definition File (HDF) is to be created.



The following options may be available (depending on the syntax and record format specified):

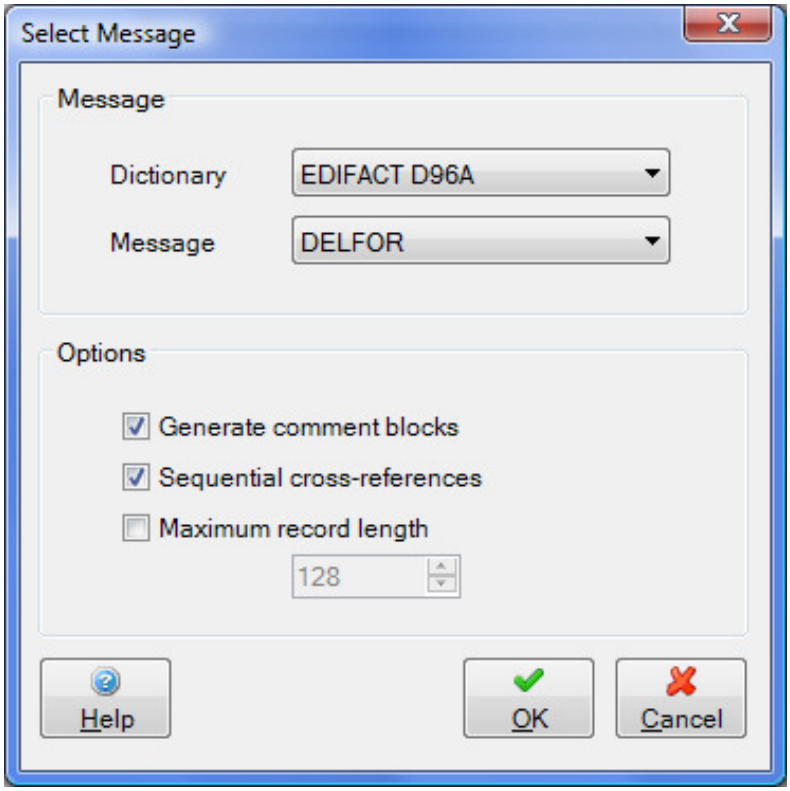
- **Record format** – specify the format of records in the input file. The options are:
  - A – ASCII format (records delimited by CRLF, CR or LF).
  - F – fixed length records.
  - L – variable length records (lengths specified in each record definition in the HDF).

    - C – CSV file.
    - T1 – TRA format with minimal validation of the input file structure.
    - T2 – TRA format with full validation of the input file structure.
- **Delimiter** – for delimited record formats (A and C) specifies the delimiter:
    - **CRLF** – carriage-return and line-feed (0x0d 0x0a).
    - **CR** – carriage-return (0x0d).
    - **LF** – line-feed (0x0a).
- **Record length** – for fixed length records specifies the constant record length.
- **Type start** – specifies the default record type field start position.
- **Type length** – specifies the default record type field length.
- **Decimal separator** – the default decimal separator.
- **CSV separator** – the character used to separate fields in a CSV file (usually ‘,’ or ‘;’).
- **CSV quote** - the quote character used for quoted fields in a CSV file (usually ‘ or “).
- **TRA record character** – the character used to indicate the start of a record in a TRA format file (usually ‘/’, as in ‘/REC-NAME’)
- **TRA record END** – flag indicating whether the record end indicator is used in a TRA format file (usually ‘/END’)
- **Skip blank lines** – flag indicating whether blank lines in the input file can be ignored.

When you are satisfied click **OK** to proceed and create the definition, or click **Cancel** to quite without creating an HDF.

4.5.6 New In-house Definition From Data Dictionary

This dialog is shown when a new In-House Definition File (HDF) is to be created from a definition in the data dictionary.



Use the drop-down list to select a **Dictionary** from which the definition is to be extracted. Use the **Message** drop-down to select the message to extract (the list is populated according to what messages are available in the selected dictionary).

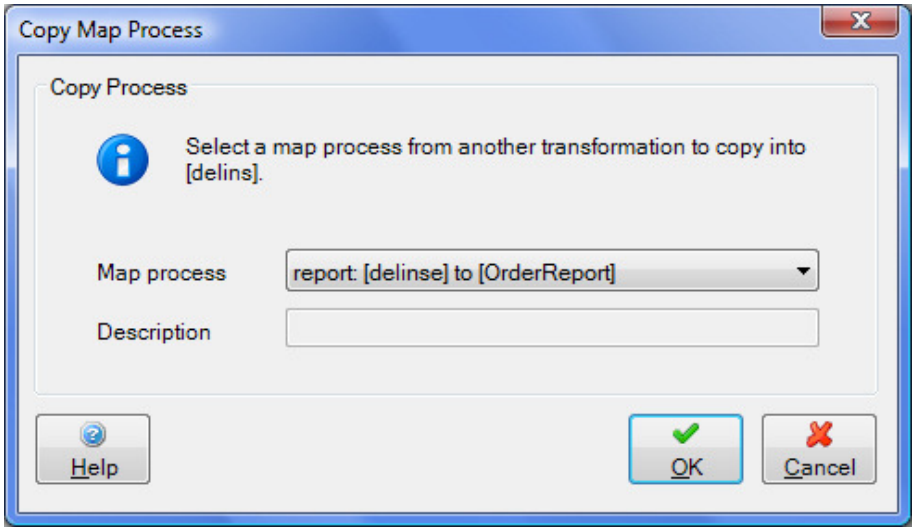
The following options are available, which affect the format of the generated EDF:

- The **Generate comment blocks** option allows you to specify whether the generated definition will contain comments for each record in a preceding block.
- The **Sequential cross-references** option allows you to specify whether the field cross-references generated are strictly sequential (for example; RFF00510, RFF00520). The alternative is that they are based on the occurrence number of the record (for example; RFF020010, RFF020020 for the second instance of RFF in the definition).
- The **Maximum record length** option allows you to specify that the maximum length of an HDF record should be no more than that specified.

When you are satisfied click **OK** to proceed and create the definition, or click **Cancel** to quite without creating an HDF.

4.5.7 Copy Process

This dialog is used to copy a process from transformation to another:



To display the dialog, use the context menu accessed by right-clicking on a Map node in the Index Explorer tree view, and choose **Copy Process**. Use the **Map process** drop-down to select a process defined in another map (transformation) and click **OK** to copy it to the map (transformation) selected in the tree view.

4.6 Import, Export, Upgrade, Backup

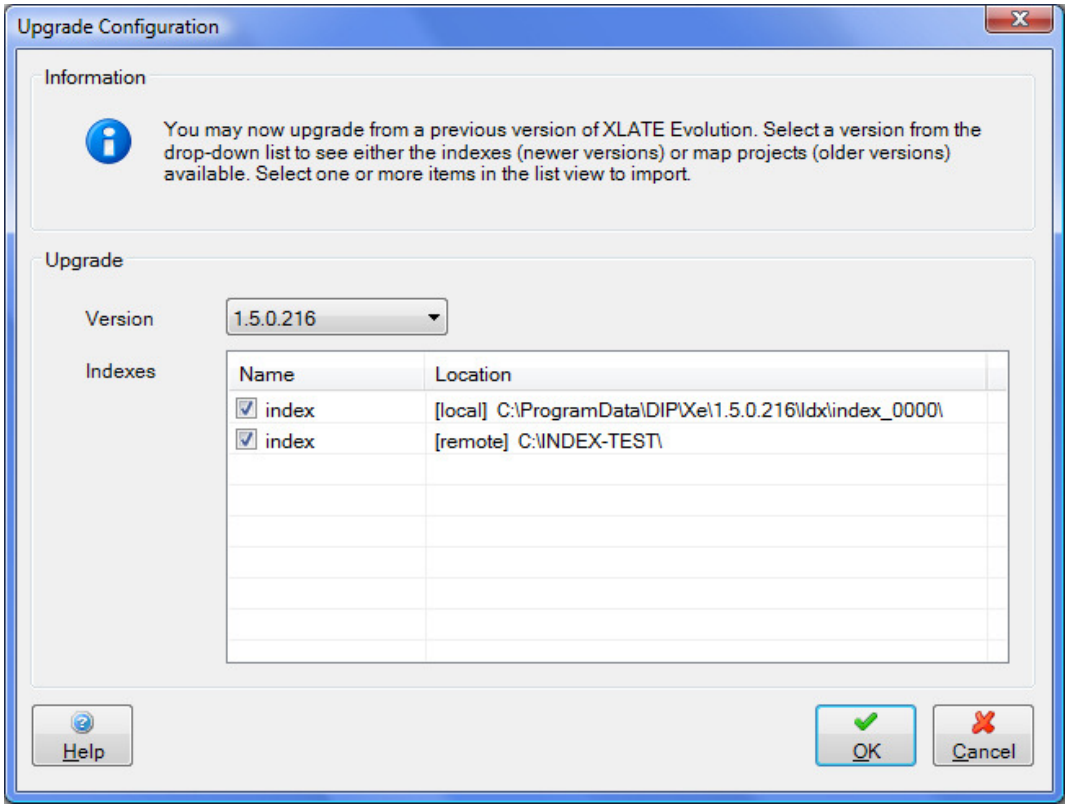
The foregoing sections described the Xe index and how it is created and manipulated in the GUI. One of the key features of the index is that all the necessary associated files (definitions, maps, tables &c) are managed by Xe. This means that you don't need to be concerned with keeping track of these files. However, it also means that the files required to perform a map (including the index itself) are hidden.

For this reason, mechanisms have been provided to support import and export of whole indexes or index entities, and these are discussed in the following sections.

In addition, there is an upgrade mechanism available when upgrading from one version of Xe to a newer one, and backup functionality for making automatic backups of a whole index, also discussed below.

4.6.1 Upgrade

When a newly installed version of Xe is run for the first time you will be offered the option to upgrade indexes from a previous version. The following dialog is shown:



Use the **Version** drop-down to select a previously installed version of Xe. All installed versions that can be found and verified as valid will be listed. When you select a version the list box below shows all of the indexes configured against that version. Use the check boxes in the list view to select the indexes to be upgraded, then click **OK**. The behaviour on upgrading differs according to whether the index is 'local' or 'remote':

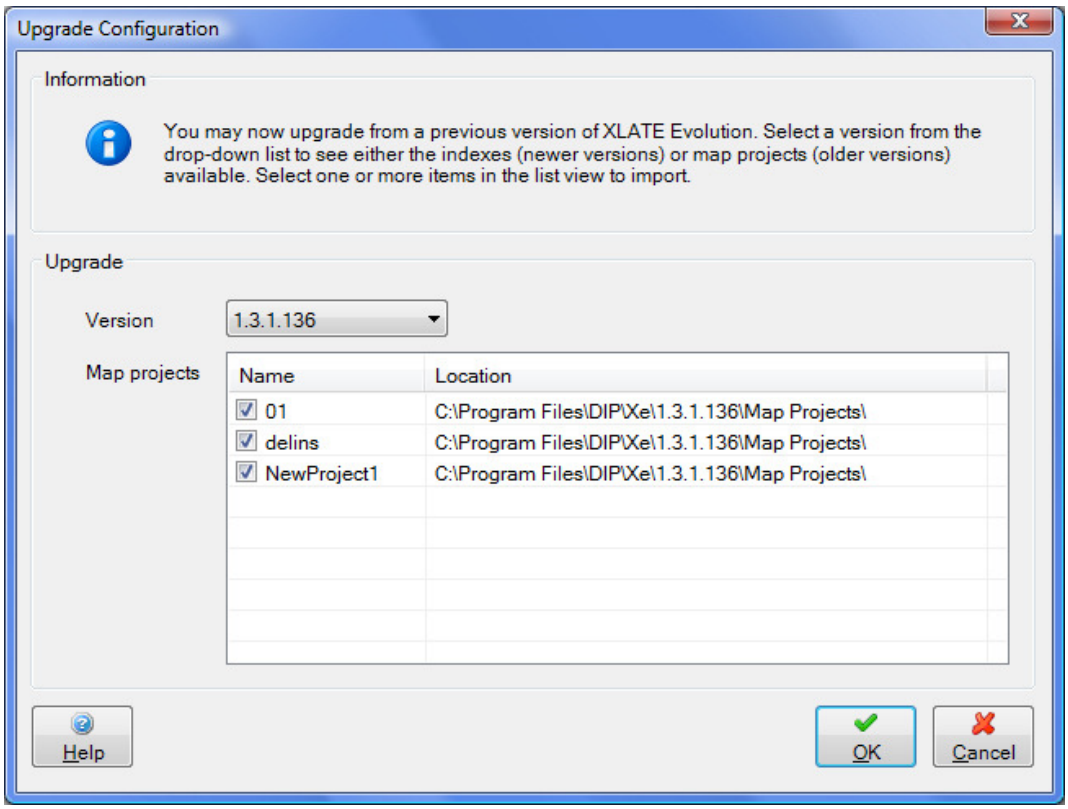
- A local index is one that was created in the data directory belonging to the selected version of Xe. If you choose to upgrade from it then a new local index is created in the data directory of the new version of Xe and all of the supporting files (definitions, maps tables &c) are copied across. If test files have been imported to a local index then these will be copied as well.
- A remote index is one that was created in a directory in some other location not related to the Xe data directory. Typically remote indexes are created on shared drives so they can be accessed by multiple users. If you choose to upgrade from a remote index then the index and associated files are not copied. Instead, a reference to the index is created in the new version of Xe.

If you **Cancel** then the dialog will be dismissed without upgrading and the upgrade will not be offered again. However, you may use the general import mechanism to import single indexes at a later stage.

After the upgrade dialog has been shown, if there is a single index configured for the new version of Xe then it will be opened in the Index Explorer. If there are multiple indexes (or none at all) then a dialog is shown for you to create or select one to use (see "Selecting An Index").

Older versions of Xe without the Index Explorer managed maps using individual map projects. There was no single index tying them together, and the associated files could be located anywhere. The upgrade procedure for these versions is therefore different. When an older version (earlier than 1.5.0.200) is

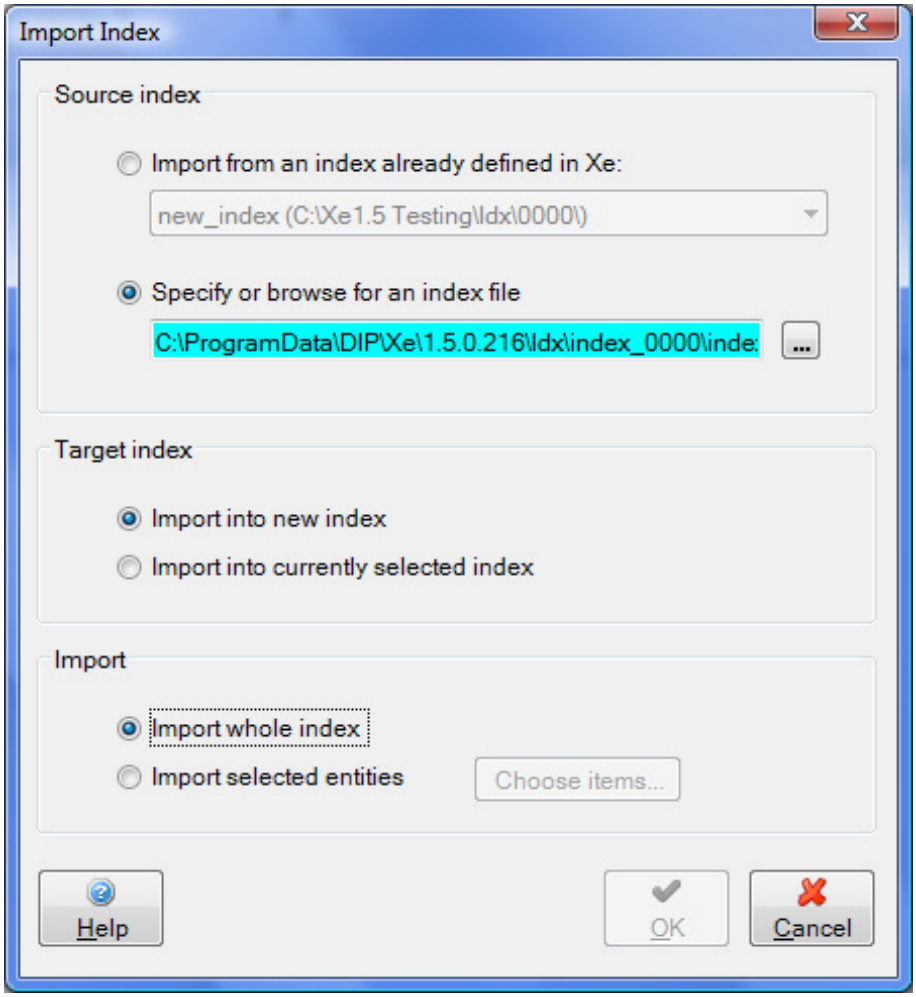
selected in the dialog, the view changes to show map projects configured against that version instead:



To perform the upgrade from this version, use the check boxes in the list view to select projects that will be upgraded, then click **OK**. Xe will create a new local index and import each project you have selected. During the import a new transformation is created in the index for each map project and all the associated files are copied locally.

4.6.2 Import Index

There is a flexible import mechanism in the Index Explorer that allows you to copy all or part of an Xe master index. Index entities can be copied into an existing index, or into a new local index. For the import option use the context menu accessed by right-clicking on the Index root node in the Index Explorer tree view and choosing **Import**. The following dialog is displayed:



The first section allows you to pick the **Source index** from which entities will be imported. You can choose to:

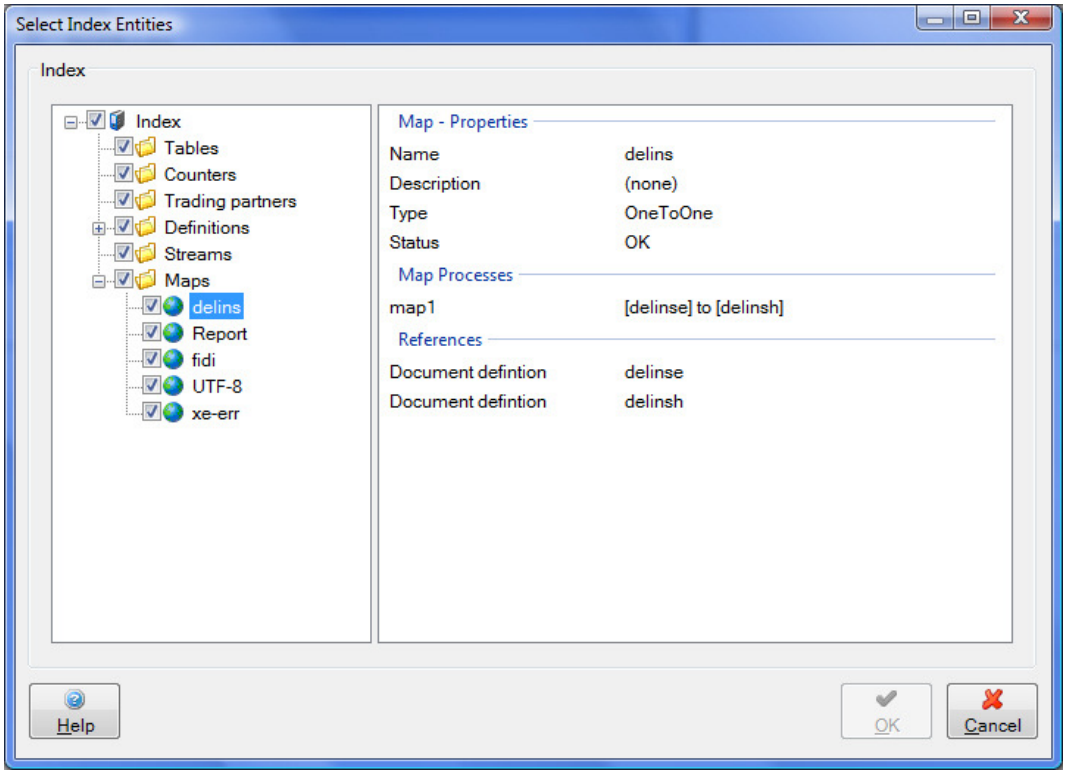
- **Import from an existing index already defined in Xe**. If there are multiple indexes defined in Xe then this option allows you to move entities into the current index from another one.
- **Specify or browse for an index file**. This option allows you to pick an index file to be imported from any location. Note that only master indexes created by Xe can be imported in this way. If you select a file that is not a valid Xe master index (including an index you have created manually) you will be warned accordingly.

The second section allows you to pick the **Target index** to which entities will be imported. You can choose to:

- **Import into new index**. A new master index is created as the target for the import.
- **Import into currently selected index**. The index currently open in the Index Explorer is used as the target for the import.

The third section allows you to pick which entities to **Import**. You can choose to:

- **Import whole index**. All entities will be imported.
- **Import selected entities**. The **Choose items** button will be enabled. Click this to show a dialog used for selecting index entities:



The dialog shows the contents of the source index in a tree view on the left hand side. Check or uncheck boxes in the tree view to indicate which entities are to be imported (by default, everything is selected).

When you click on a folder in the tree view the list view on the right hand side shows a list of entities in that category (for example, all maps, all counters &c).

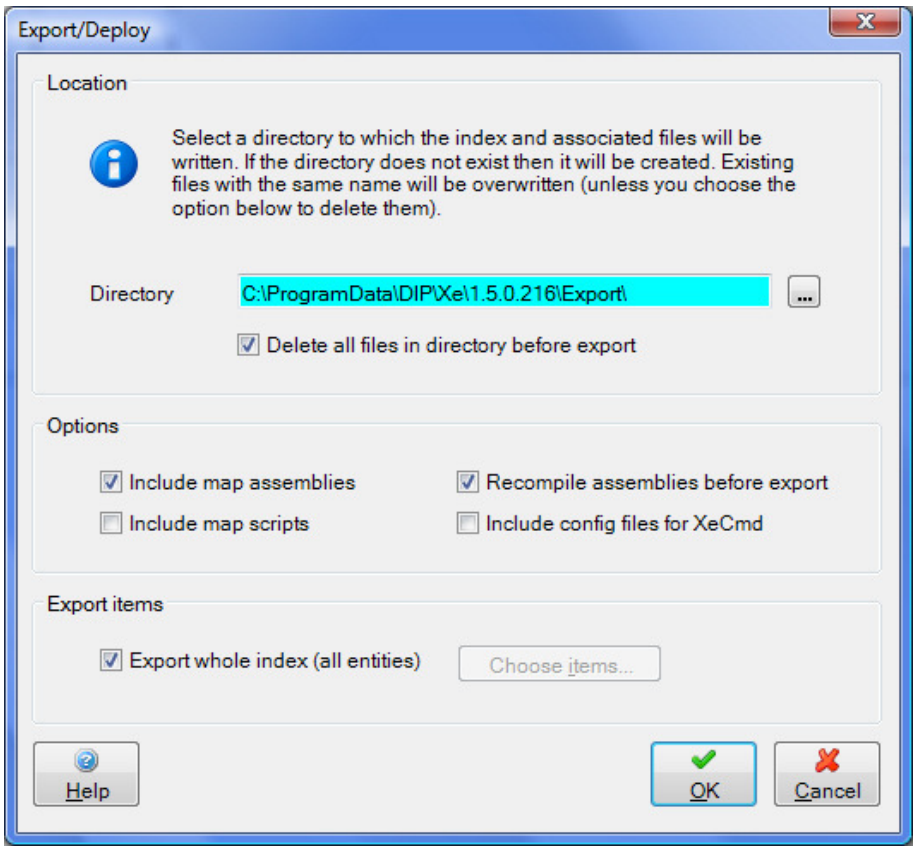
When you click on an individual entity the list view shows its properties. These include the other entities that it references. You should make sure that you do not try to import an entity without importing other entities that it depends upon. For instance, in the example above, the document definitions named ‘delinse’ and ‘delinsh’ are required by the selected map.

When you are happy with the selections made, click **OK** to return to the import dialog. Alternatively, click **Cancel** to dismiss the dialog without saving changes.

4.6.3 Export (deploy)

Xe maps are typically developed using the Xe mapper GUI and then deployed to a different environment. Usually the production environment is Data Interchange’s business integration and communication server, ODEX Enterprise. However, Xe can also be used in other ways through the command line version ‘XeCmd’.

In either case various files need to be deployed to the production environment, including definitions, map assemblies and the index itself. For the export mechanism use the context menu accessed by right-clicking on the Index root node in the Index Explorer tree view and choose **Export/deploy**. The following dialog is displayed:

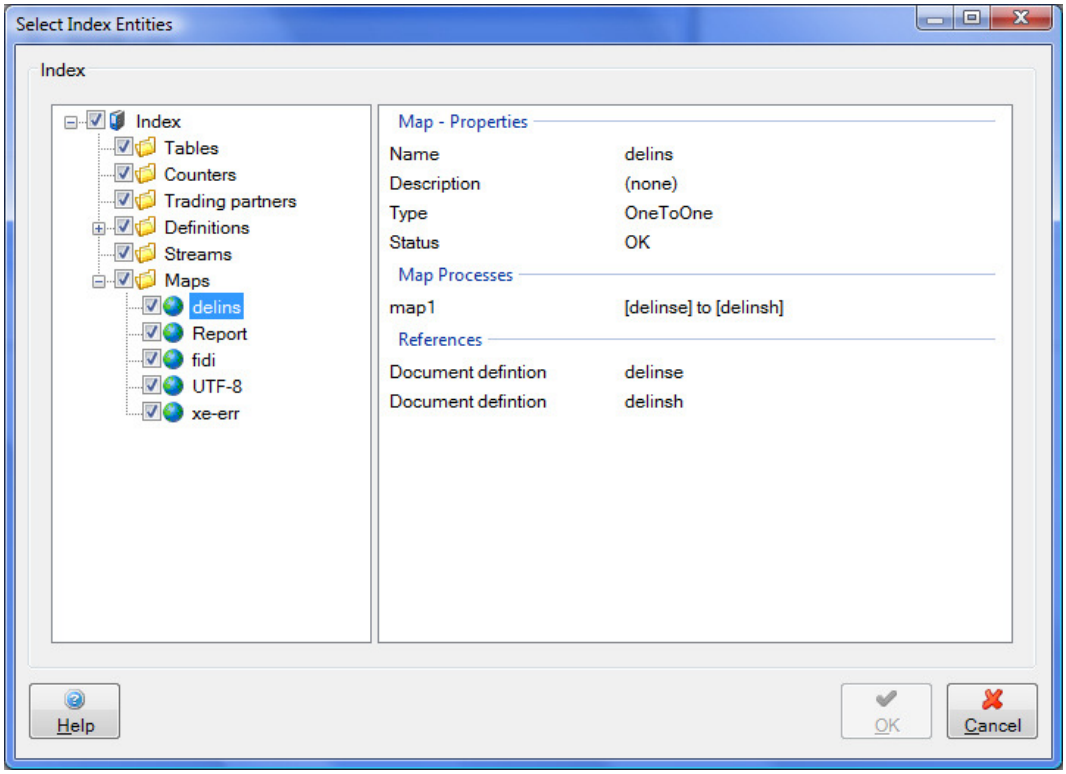


The first section allows you to specify an export **Location**. This is simply a **Directory** into which the required files will be copied. Enter the directory path or click on the browse button (**...**) to browse for a directory. Check the box to indicate that the directory should be emptied first, if required.

The second section contains the following **Options**:

- **Include map assemblies**. Indicates whether the map assemblies are included in the export. Assemblies must be included unless they are to be created from the scripts externally using XeCmd.
- **Recompile assemblies before export**. Indicates whether all exported assemblies should be compiled first.
- **Include map scripts**. Indicates whether map scripts should be exported. Map scripts are not required for mapping once the map assemblies have been created, so do not need to be exported unless the assemblies are to be created from the scripts externally using XeCmd.
- **Include config files for XeCmd**. Indicate whether to export config files that can be used to run the maps under XeCmd (note that the config files produced will include full paths of files in the export directory, which will need editing if the files are moved).

The third section allows you to specify what is exported. If the check box is selected then the whole index and all supporting files are exported. If not, then the **Choose items** button is enabled. Click the button to show a dialog used for selecting index entities:



The dialog shows the contents of the current index in a tree view on the left hand side. Check or uncheck boxes in the tree view to indicate which entities are to be exported (by default, everything is selected).

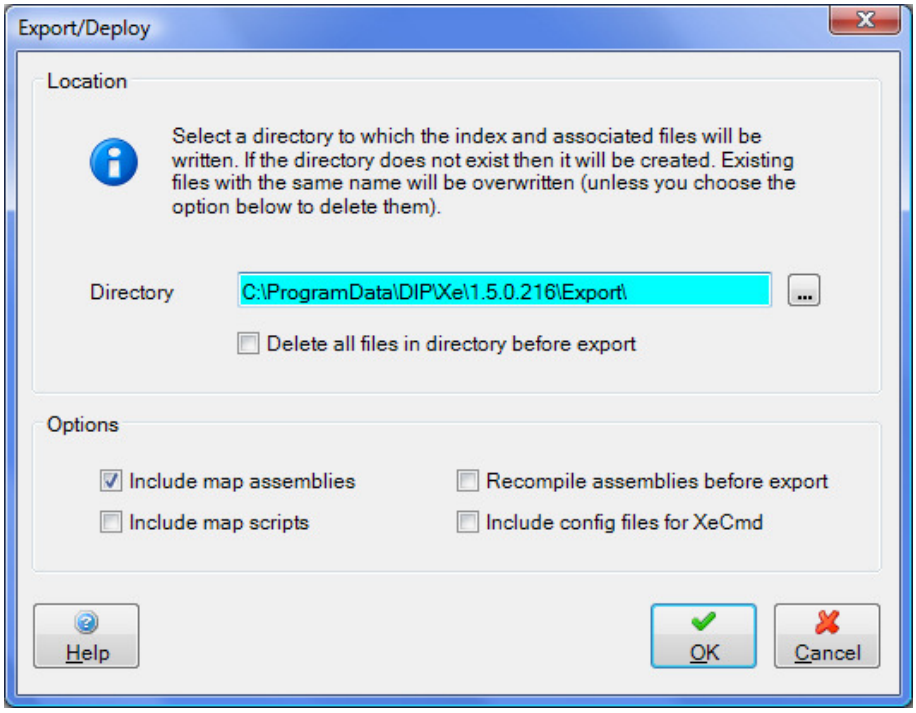
When you click on a folder in the tree view the list view on the right hand side shows a list of entities in that category (for example, all maps, all counters &c).

When you click on an individual entity the list view shows its properties. These include the other entities that it references. You should make sure that you do not try to export an entity without exporting other entities that it depends upon. For instance, in the example above, the document definitions named ‘delinse’ and ‘delinsh’ are required by the selected map.

When you are happy with the selections made, click **OK** to return to the export dialog. Alternatively, click **Cancel** to dismiss the dialog without saving changes.

4.6.4 Export map

You can choose to export files for a single map (transformation) instead of the whole index. For the export map mechanism use the context menu accessed by right-clicking on the required map node in the Index Explorer tree view and choose **Export map**. The following dialog is displayed:



The first section allows you to specify an export **Location**. This is simply a **Directory** into which the required files will be copied. Enter the directory path or click on the browse button (**...**) to browse for a directory. Check the box to indicate that the directory should be emptied first, if required.

The second section contains the following **Options**:

- **Include map assemblies**. Indicates whether the map assemblies are included in the export. Assemblies must be included unless they are to be created from the scripts externally using XeCmd.
- **Recompile assemblies before export**. Indicates whether all exported assemblies should be compiled first.
- **Include map scripts**. Indicates whether map scripts should be exported. Map scripts are not required for mapping once the map assemblies have been created, so do not need to be exported unless the assemblies are to be created from the scripts externally using XeCmd.
- **Include config files for XeCmd**. Indicate whether to export a config file that can be used to run the map under XeCmd (note that the config file produced will include full paths of files in the export directory, which will need editing if the files are moved).

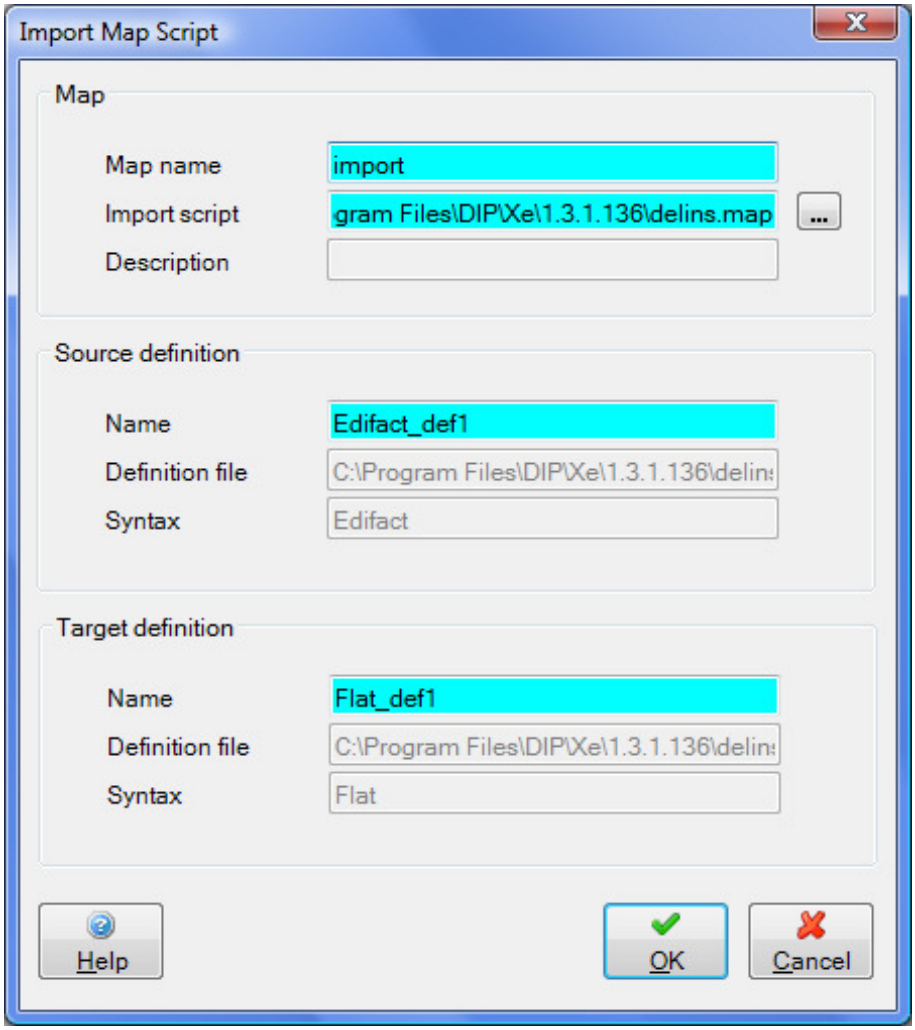
4.6.5 Backup and restore

You can choose to back up an index, and all of the associated files that are managed by Xe, at any time using the manual backup mechanism. Use the context menu accessed by right-clicking on the Index root node in the Index Explorer tree view and choose **Backup**. You will be prompted to browse for a directory where the backup will be created. **If the directory chosen is not empty then all files and sub-directories will be removed before the backup is performed**. You will be prompted to confirm the delete operation first.

To restore an index from a backup use the context menu accessed by right-clicking on the Index root node in the Index Explorer tree view and choose **Restore**. You will be prompted to browse for an index file. After you have selected a file, the index will be restored by creating a new local master index and copying the backup to it.

4.6.6 New map from script

You can choose to create a new map from an existing script at any time using the manual import mechanism. Use the context menu accessed by right-clicking on the Maps folder in the Index Explorer tree view and choose **New map from script**. The following dialog will be displayed:



The **Map name** text box allow you to enter a name for the new map. This must be unique within the current index.

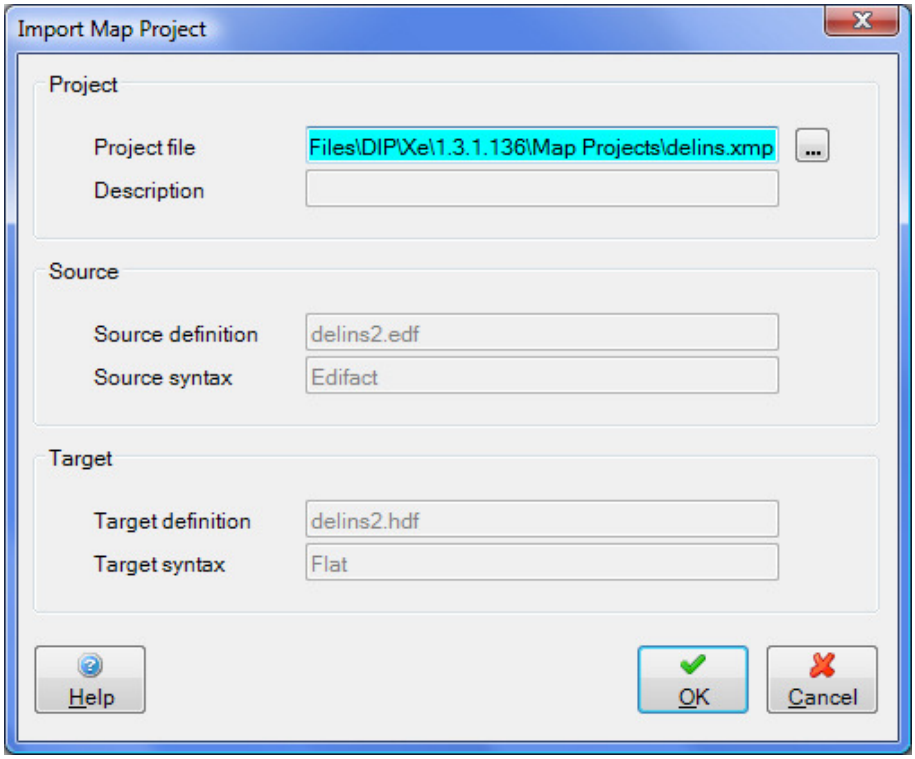
Enter the full path of an Xe map script in the **Import script** box, or click the browse button (**...**) to browse for a script file. When a valid script file has been specified the source and target definition file and syntax text boxes will be populated in the dialog by reading them from the script.

When the import is performed, Xe will create new document definitions in the index for the source and target. Enter a **Name** for each definition if required, or allow them to default.

When you are satisfied, click **OK** to import the script. Xe will create the necessary definitions and a new transformation with a single map process.

4.6.7 New map from project

You can choose to create a new map from a map project at any time using the manual import mechanism. Use the context menu accessed by right-clicking on the Maps folder in the Index Explorer tree view and choose **New map from project**. The following dialog will be displayed:

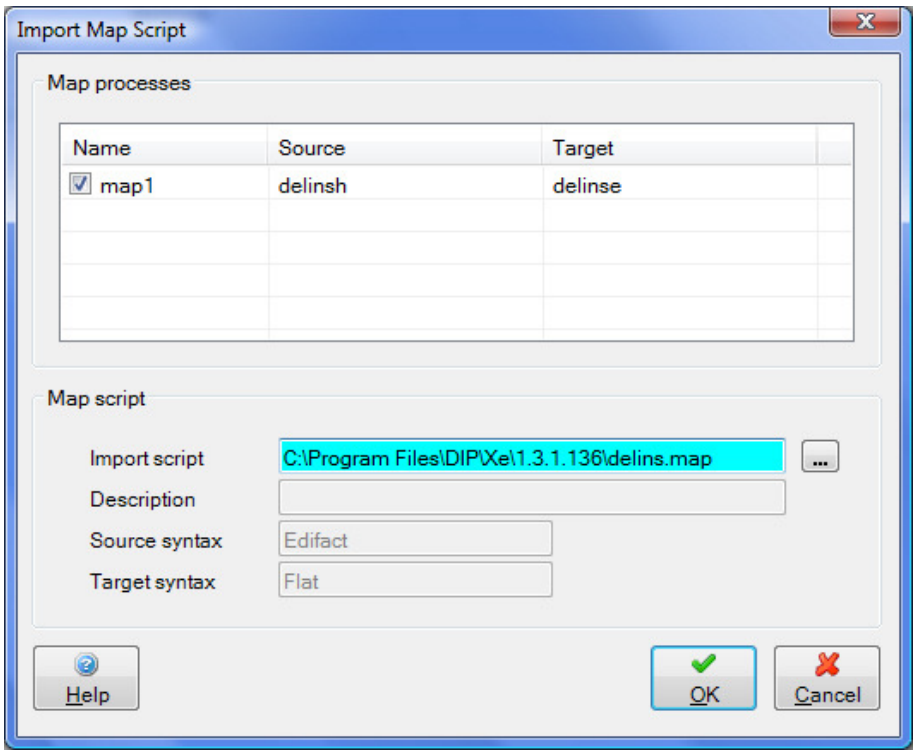


Enter the full path of an Xe map project in the **Project file** box, or click the browse button (**...**) to browse for a project file (projects have the extension .xmp and can be found in the MapProjects sub-directory of the installation directory of old versions of Xe). When a valid project file has been specified the source and target definition file and syntax text boxes will be populated in the dialog by reading them from the project.

When you are satisfied, click **OK** to import the project. Xe will create the necessary definitions and a new transformation with a single map process.

4.6.8 Import script

You can choose to import an existing map script into a map, to use with a selected map process, using the manual import mechanism. Use the context menu accessed by right-clicking on the required Map entity node in the Index Explorer tree view and choose **Import script**. The following dialog will be displayed:



The first section lists the **Map processes** already defined for the selected map (transformation). If there are multiple processes, choose **one** process against which the map script is to be imported using the check boxes in the list.

The second section allows you to choose the **Map script** to be imported. Enter the full path of an Xe map script in the **Import script** box, or click the browse button (**...**) to browse for a script file. When a valid script file has been specified the source and target syntax and description text boxes will be populated in the dialog by reading them from the script.

When you are satisfied, click **OK** to import the script. Xe will import he script and associate it with the selected map process. Note that the existing source and target document definitions associated with the process will be maintained and no new document definitions will be created.

5 Using the Mapper

5.1 Introduction

This chapter describes the elements of the Xe Mapper and explains in detail how to use the GUI to create a map.

During the development of a map, you will need to provide:

- A definition of the source file
- A definition of the target file
- Mapping links between the two files

These can then be used to generate a mapping script.

The combination of source file definition, target file definition and map file is created in the Index Explorer as described earlier. For the test environment you need to provide:

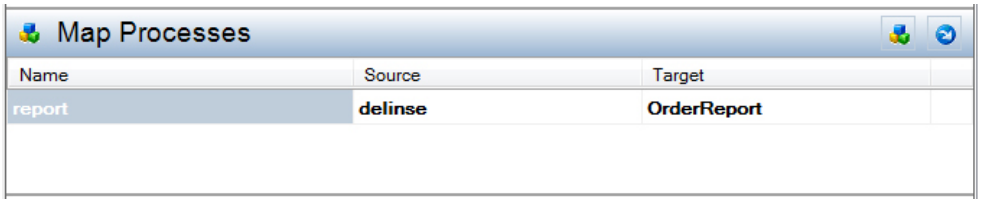
- the actual input data file
- a path and filename for the resulting output file
- a path and filename for the log file.

These can then be used to perform a transformation of the input data file to create an output file.

5.1.1 Loading a map

Formerly, Xe maps were defined by creating map projects to link together source and target definitions and the map script. A map was loaded by selecting a project from a list. In newer versions of Xe, maps are defined in the Index Explorer and the mapper is launched from the Explorer in one of these ways:

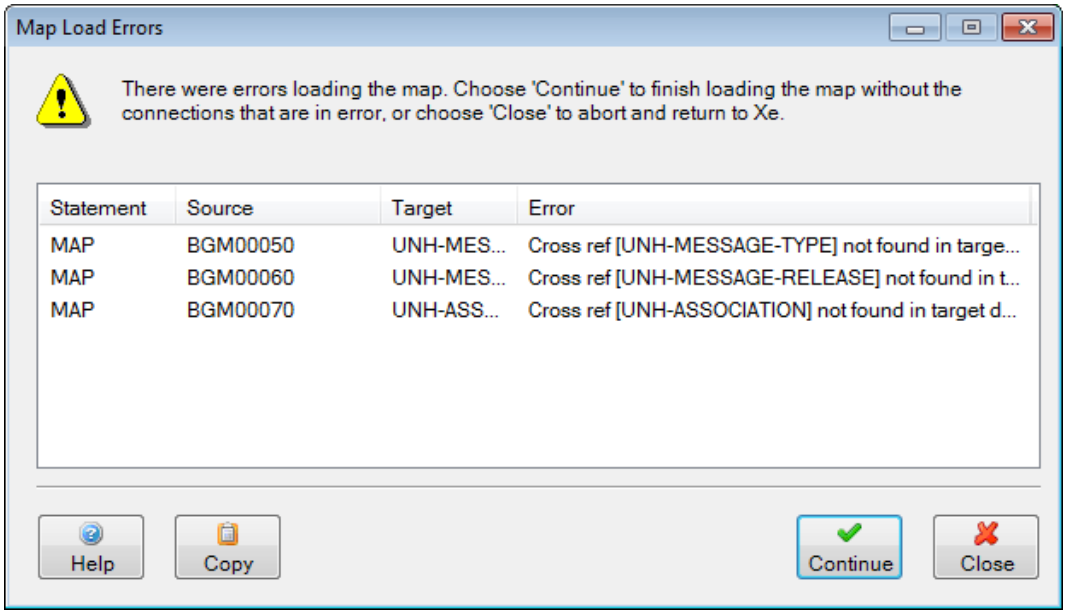
- Use the context menu accessed by right-clicking on the required Map entity node in the Index Explorer tree view and choose **Open mapper**.
- When a Map entity node is selected in the Index Explorer tree view, the Map Processes list view is displayed:



Click the **Open mapper** button ( ) to edit the currently selected map process in the mapper view.

5.1.2 Errors Loading a Map

When load a map and there are recoverable errors, Xe will show the following dialog.



The most common cause of recoverable errors is where a cross-reference in a definition file (EDF or HDF) has been changed. When Xe loads a map script file in which the old cross-reference appears, and consequently cannot find the entity in the definition file, the error will be added to the list. Not all errors are recoverable; in particular, problems encountered in the syntax of the map script file will usually be fatal.

If you click **Continue** then Xe will continue to load the map, but it will not load any map connections that are affected by the errors displayed. The next time that you generate the map, the script file will be saved without the listed connections (unless you have restored or replaced them using the mapper).

If you click **Cancel** then Xe will stop loading the map and close the mapper view. You can then fix any problems in the definition files before trying to load the project again.

The **Copy** button copies the list of errors to the clipboard. Use this button when you want to take a copy of the errors detected for future reference. The data copied can be pasted into any text editor.

5.1.3 Mapper menu

Click the Mapper menu item to see the menu options available for the Mapper. These options will be available only if the Mapper page tab is currently selected.

5.1.3.1 Connect cross-references

Select **Mapper >> Connect cross references** to create a simple map between cross-references. In Xe's predecessor, Xlate, a map was defined by matching cross-references in the EDF and HDF, and the EDFs and HDFs generated by the Data Dictionary will contain cross-references that match in this way. This option creates a map connection between each source and target node with a matching cross-reference. This option cannot be used where the source or target is XML (as there are no cross-references to match), but it can be used where the source and the target definitions are of the same type (EDF-to-EDF and HDF-to-HDF).

The cross-reference is the only basis upon which the connections are made, so if the data represented by the definitions are not related, but the definitions share cross references, the connections will be made. Existing map connections will be left untouched, but duplicate connections will not be created. This option

will often result in a large number of new connections being created and it should be used with caution.

5.1.3.2 Configuration

This option will only be available if the Mapper page tab is currently selected.

Select **Mapper >> Configuration** or click **F3** to bring up the Map Configuration dialog.

For more details, please refer to the section entitled “Configuring a map”.

5.1.3.3 Generate script

This option will only be available if the Mapper page tab is currently selected.

Select **Mapper >> Generate script** or click **F5** to generate the .map script file from the selected map.

The map script has two roles in Xe mapping. First, it is the file used to store map information created in the GUI. Second, the map script contains the instructions that are used to generate the mapping assembly. So generating the script using this option saves any changes to the map that have been made in the GUI, and means that the next time the map assembly is generated it will contain up-to-date map instructions.

5.1.3.4 Generate map assembly

This option will only be available if the Mapper page tab is currently selected.

Select **Mapper >> Generate map assembly** or click **F6** to generate the map assembly from the selected map.

The map assembly is a file generated from the instructions in the map script (it is a Windows Dynamic Link Library, sometimes called an Application Extension) including the MAP and LOOP statements and your code snippets. when Xe is used to process a file it loads the map assembly to carry out the mapping instructions.

Selecting this option has the result of both generating the mapping script, and then generating the map assembly from the instructions in the script. Before generating the assembly, you must first supply a filename in the Map Configuration dialog.

5.1.3.5 Test map

This option will only be available if the Mapper page tab is currently selected.

Select **Mapper >> Test map** or click **F7** to test the map assembly. This option should create the required output file from a given input file.

This option is used to test a single map from the Xe GUI. To use it a number of details must be entered in the Map Configuration dialog. As a minimum, they are:

- A source filename – the file to be used as input to the mapping process
- A destination filename – the output file to be produced by the mapper

More details are included in the section entitled Configuring a Map.

When this option is selected, Xe will generate both the map script and the map assembly, load and execute the map assembly, and optionally open the output

file. The map will not be executed if errors occur when writing the script or generating the assembly.

5.1.3.6 Run map process

This option will only be available if the Mapper page tab is currently selected.

Select **Mapper >> Run map process** or click **F8** to run the map assembly using the source and target files and index you have configured. This should create the required output file from a given input file.

The difference between using this option (Run map process) and the previous option (Test map) is that this option calls the external Xe command line application (XeCmd.exe) to do the mapping. It therefore allows you to simulate mapping in a production environment (under Xe or ODEX Enterprise).

5.1.3.7 Close mapper

This option will only be available if the Mapper page tab is currently selected.

Select this option if you want to close the Mapper page tab. If you have not yet generated a script from the map, you will be warned that closing the map page will lose any changes you have made and given the option to leave the page open.

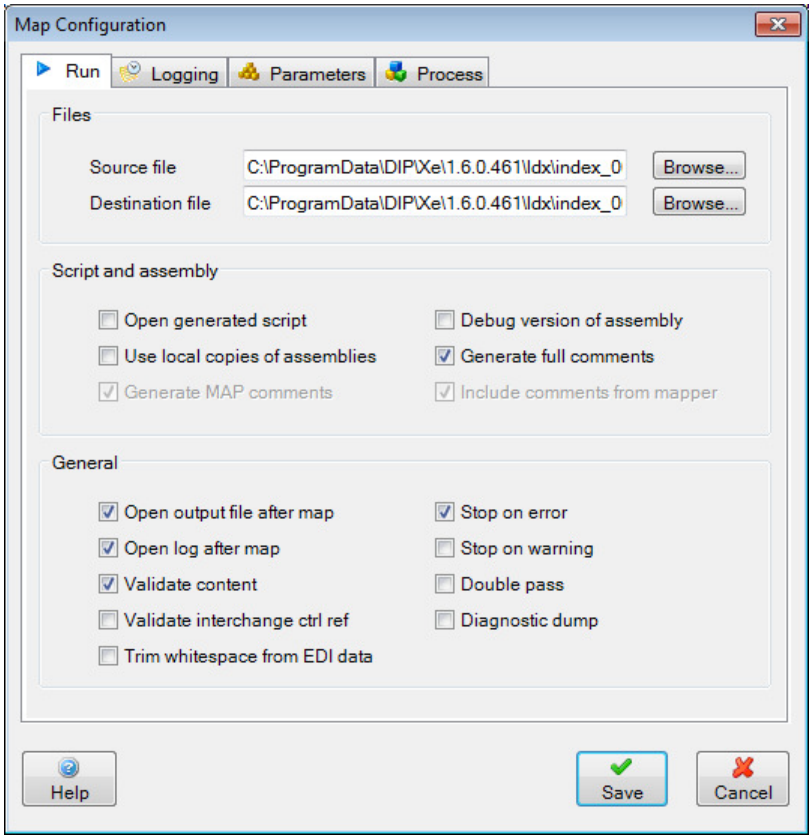
5.1.4 Configuring a map

During the development phase of a map, you will probably want to perform extensive tests to ensure that the map produces the expected output from any given input. To this end, Xe allows you to configure the map environment, thus avoiding the need to provide the required information every time you run the map assembly.

To configure a map, you can either select the menu item **Mapper >> Configuration** or click on **F3**. The map config dialog will be shown. This dialog has 4 page tabs – Run, Logging, Parameters and Process. These pages are described fully in the following sections.

5.1.4.1 Run

Use this page to provide details of the files that will be used when testing the map and to specify other options:



The first section of this page allows you to select files for testing:

- **Source file** – enter the full path of a source file that will be used when testing, or click the **Browse** button to browse for a file. You will be offered the option to import the file by copying it into the Xe data directory.
- **Destination file** – enter the full path of a target file that will be created when testing, or click the **Browse** button to browse for a file location.

The second section of this page allows you to select options relating to the map script and assembly:

- **Open generated script** – indicates whether the map script should be opened each time the script or map assembly is generated.
- **Use local copies of assemblies** – indicates whether Xe should copy map assemblies to a temporary location in its data directory before running the map. If you are working with a remote index that is generating map assemblies on a network drive, you may experience problems if your map requires certain security permissions (because you are using database functions or code snippets that try to write to disk files, for instance). These can be overcome by using local copies of the assemblies.
- **Debug version of assembly** – indicates that the map assembly should be generated in a debug version. The debug version includes extra logging indicating the map script statement being performed at each stage of the mapping process.
- **Generate full comments** – means a comment banner is written for each loop statement and map statement, including an automatically generated comment and any comment you enter against the map connection in the GUI.

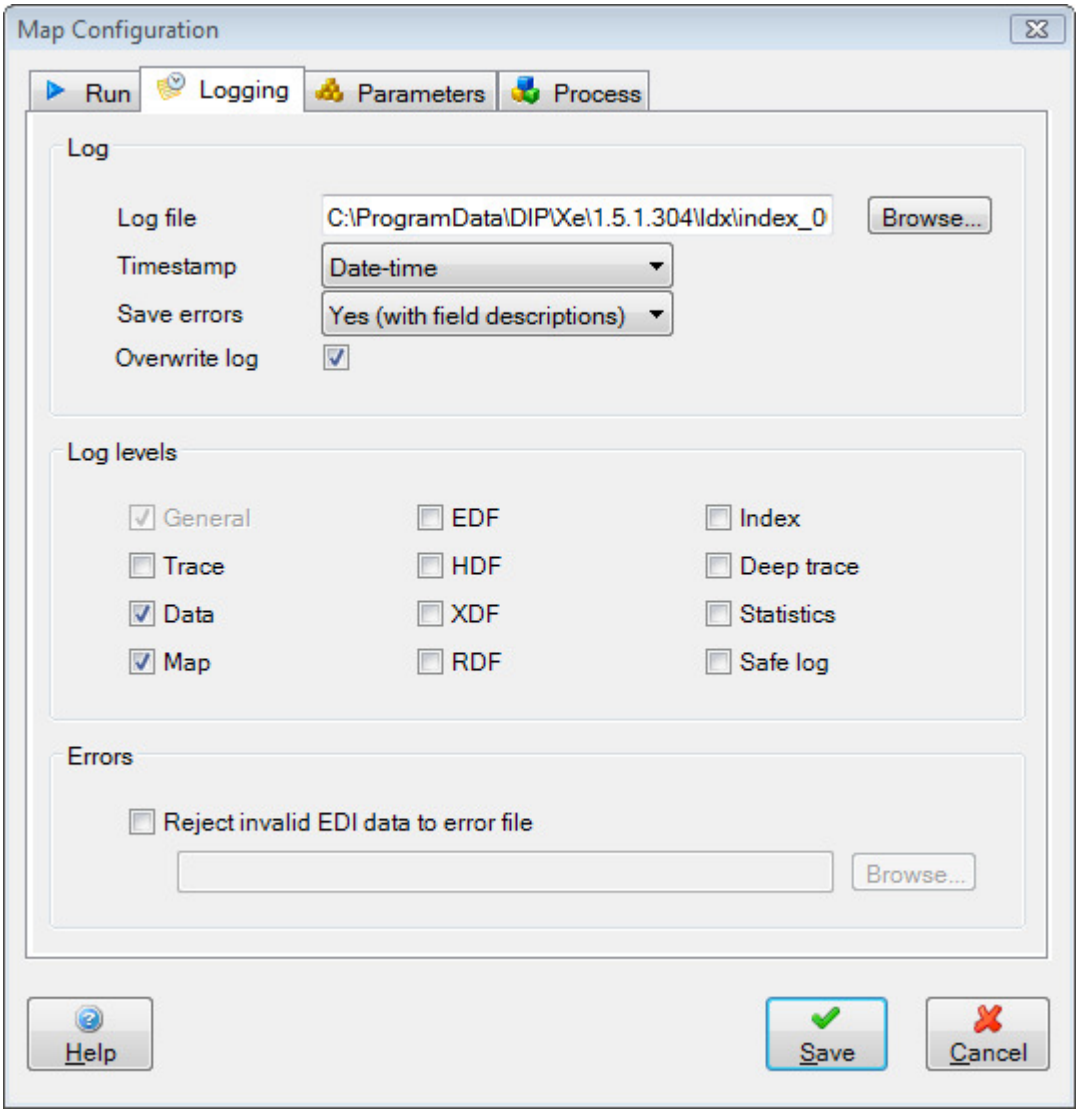
- **Generate MAP comments** – means a comment banner is written for each loop statement and map statement, including automatically generated comments.
- **Imclude comments from mapper** – means a comment banner is written in respect of comments you enter against map connections in the GUI.

The third section of this page allows you to select general options:

- **Open output file after map** – indicates whether the output file(s) should be opened each time the map is run.
- **Open log file after map** – indicates whether the log file should be opened each time the map is run.
- **Validate content** – indicates whether Xe should validate content and formatting against the definition file.
- **Validate interchange control ref** – indicates whether Xe should attempt to find a stream to use for validating the interchange control reference on an incoming EDI message. This option should be checked only if a suitable stream has been defined in the index.
- **Trim whitespace from EDI data** – indicates whether Xe should trim any whitespace padding from the end of EDI data elements.
- **Stop on error** – indicates whether Xe should stop map processing when a recoverable mapper error is issued.
- **Stop on warning** - indicates whether Xe should stop map processing when a mapper warning is issued.
- **Double pass** – indicates whether Xe should perform a double pass by processing the output from a map as the input of a second map. This option should be checked ony if the index is configured for both maps (that is, a transformation is defined that is triggered by the output of the first map).
- **Diagnostic dump** – indicates whether Xe should create files in the same directory as the output file with dumps of the contents of the source and target DOMs before each map is performed.

5.1.4.2 Logging

Use this page to specify logging options:



The Logging page allows you to specify whether to create a log and, if so, where to put it. You can then specify which types of logging you want to see.

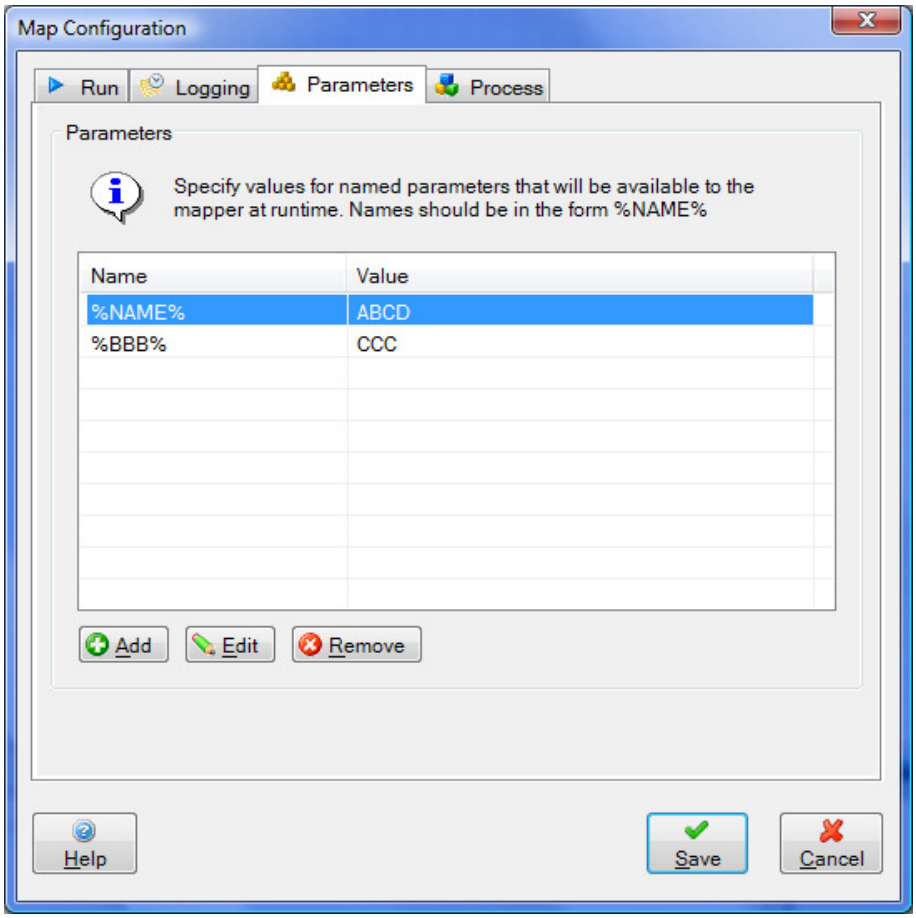
- **Log file** – specify the log file location to use when running a map from the GUI.
- **Timestamp** – specify the format of the timestamp that appears at the left hand side of each log line.
- **Save errors** – specify whether to extract any validation errors that are logged and save them for display after a map is run.
- **Overwrite log** – indicates whether the log file should be overwritten each time the map is run.
- **Log types** – the following log types can be configured:
    - **General** – the highest level of log messages; always written if logging is turned on.
    - **Trace** – log more detailed general messages.
    - **Data** – log each segment/record read from the source and each element/field loaded into/read from the DOM.
    - **Map** – log each data item that is mapped.
    - **EDF** – log detailed messages when loading an EDF.

- **HDF** – log detailed messages when loading an HDF.
- **XDF** – log detailed messages when loading an XDF.
- **RDF** – log detailed messages when loading an RDF.
- **Index** – log detailed messages when loading a runtime index.
- **Deep trace** – log very detailed diagnostic messages relating to the reading of data from the source DOM and writing of values to the target DOM during mapping (note that deep trace logging slows performance of the mapper dramatically).
- **Stats** – write performance statistics at the end of the log.
- **Safe log** – log file is opened and closed each time a message is written so messages are not lost if there is a fatal error (note that safe logging slows performance of the mapper dramatically).

- **Reject invalid EDI** – You can choose a file to which invalid EDI messages will be rejected.

5.1.4.3 Parameters

The Parameters page is used to display and manage external parameters that are provided to the mapper at runtime:



Parameters in the form %NAME% may be used in the map as placeholders for values to be provided when the map is executed. They allow you to provide per-execution data to the mapper that is outside of the source document. Parameters configured here are those that will be used when testing the map from the Xe mapper view. When invoking the map directly (using F7) the configured parameters are passed to the mapping engine. When invoking the

map to run in a separate process (using F8) they are written to the generated config file.

Parameters

The dialog includes a list of parameter names and values associated with the map project.

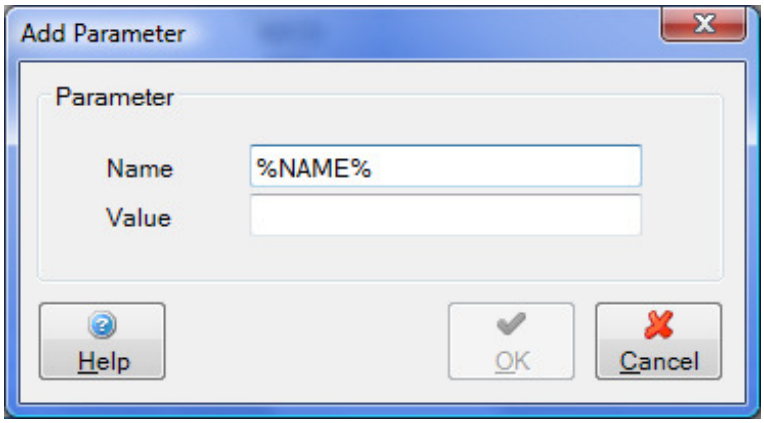
Click the **Add** button to add a new parameter.

Click the **Edit** button to change the value of an existing parameter selected in the list view.

Click the **Remove** button to delete an existing parameter selected in the list view.

Add

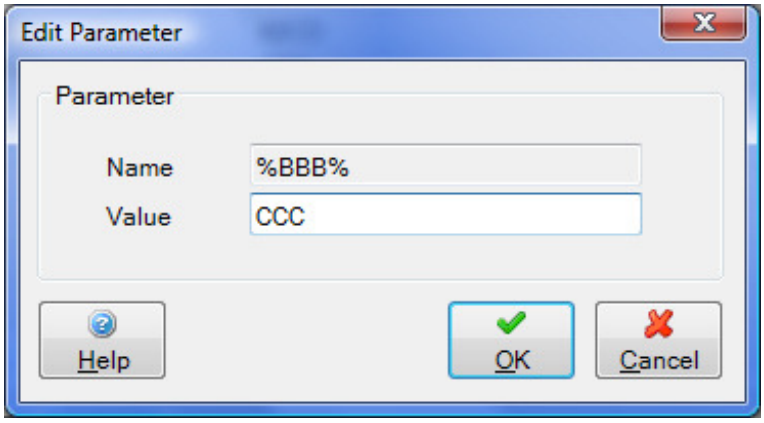
When the Add button is clicked the following dialog will be displayed



Enter a valid parameter name and value and click OK to save it.

Edit

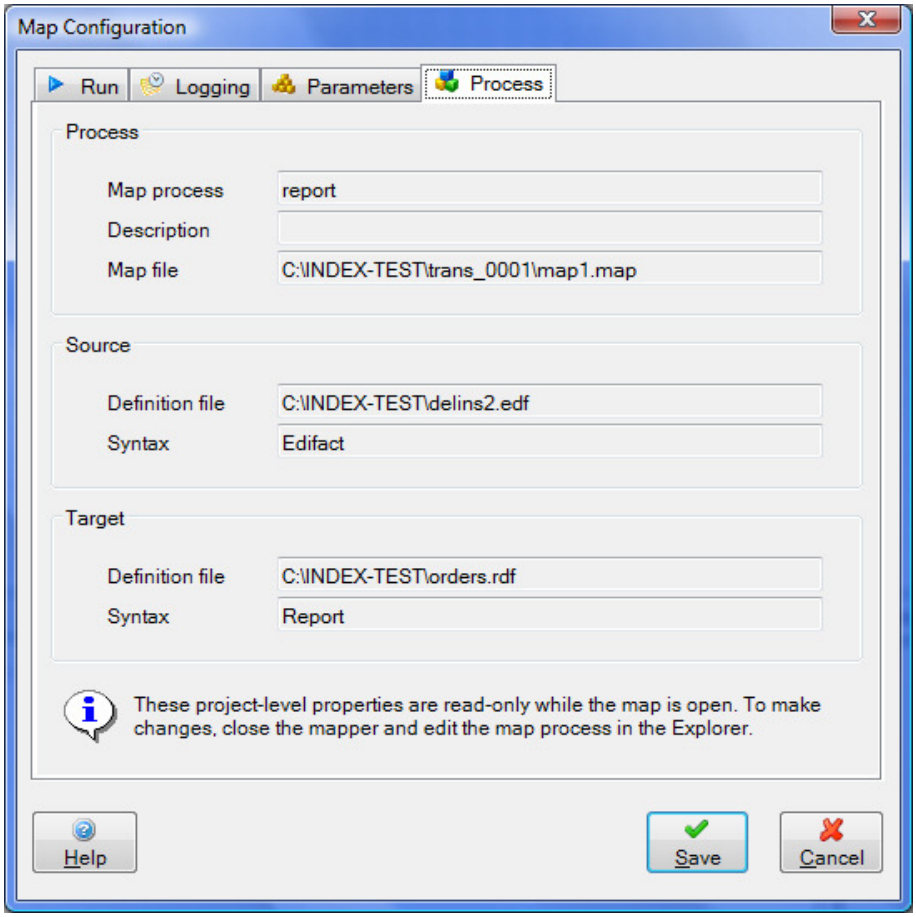
When the Edit button is clicked the following dialog will be displayed



You can edit the parameter value (but not its name) and click OK to save it.

5.1.4.4 Process

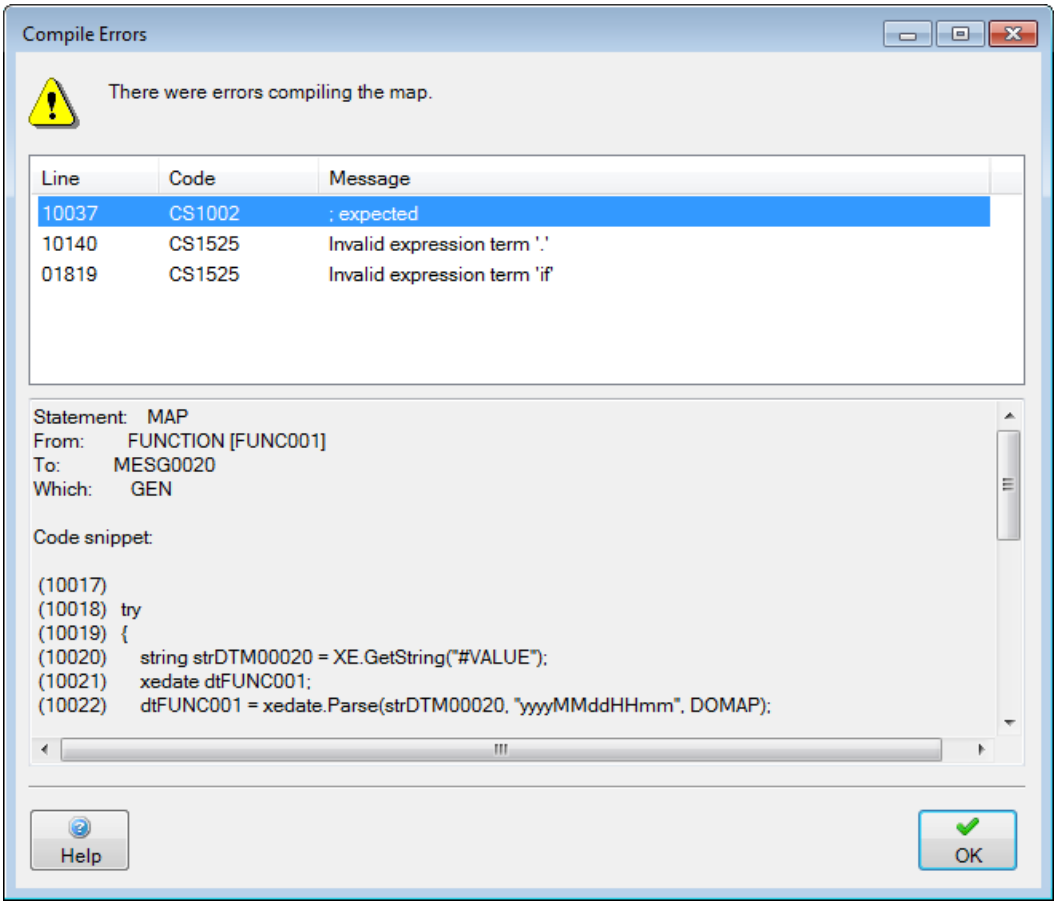
The Process page contains read-only information about the map process currently open in the mapper:



If you want to make any changes to these settings you must do so using the Index Explorer.

5.1.5 Map compilation errors

When you choose to compile the map DLL (or test the map from the GUI), Xe tries to create a valid .NET assembly that will perform the map. If it fails, the most likely reason is that a code snippet has been entered that has caused problems. In the event of compilation errors occurring, this dialog is displayed:



The list view in the top half of the dialog lists the errors returned by the compiler (line number in the generated code, compiler error number and compiler message).

When you select an item from the list, the text view in the bottom half of the dialog shows more information about the source of the problem:

- Statement – this is the map script statement type that was the source of the error
- From – this is the source of the connection that caused the error.
- To – this is the target of the connection that caused the error.
- Which – this identified which code snippet was to blame (`PRE`, `POST`, `GEN`, `INIT`, `INITMAP` or `ENDMAP`).
- Code snippet – this is the code snippet in which the error occurred (if it could be found); it includes the original line numbers in parentheses on the left.

5.2 The file definitions

Xe uses three types of definition file: the EDFs (EDI Definition Files) and HDFs (In-house Definition Files) that are also used by Xlate; and a new XDF (XML definition file) that Xe generates.

With a small number of exceptions, the syntax of the EDF and HDF definition files are identical to that described for Xlate, and so is not reproduced here. Please refer to the Xlate manual for full details.

Not all parts of the EDF and HDF, as currently used in Xlate, have any meaning when used with Xe. However, any existing EDF and HDF files you may have can be used in Xe, since Xe will ignore those parts that it does not need.

An EDF file can be generated quickly and easily from the Data Dictionary, as can an HDF file to match the EDF file. However, if you require a non-standard in-house file definition, you will either have to edit the one generated by the Data Dictionary or write your own.

The tables below summarise the relevance of each definition record type when used with Xe. For details of how to define the EDF and HDF, please refer to the Xlate manual.

5.2.1 EDF records

The EDF file (EDI definition file) is used to describe the contents and format of an EDI file.

Record	Purpose	Comments
CND	Conditions that determine validity of an EDI message	Still applicable, as it specifies checks to be performed on loading
DEL	Data element definition	Still applicable, but qal-ref field ignored
FMT	File format	Parsed but ignored
GRP	Explicit group	Still applicable
MSG	Message type	Parsed but ignored
PRM	Constant	Still applicable if used for validation, but cannot be used to specify values
PLS	Segment pool start	Parsed but ignored
PLE	Segment pool end	Parsed but ignored
QAL	Qualified element definition	Parsed but ignored
SEG	Segment definition	Still applicable
SEL	Sub-element definition	Still applicable, but qal-ref field ignored
X12	X12 message type	Parsed but ignored

In addition, all service segment definitions (UNB, UNH, ISA, STX etc) are ignored by Xe, which uses its own internal definitions when parsing EDI.

5.2.2 HDF records

The HDF file (in-house definition file) is used to describe the contents and format of the in-house file. In-house file formats supported by Xe include flat files, CSV files and SAP IDocs.

Record	Purpose	Comments
CHG	Change field definition	Treated as a FLD
DST, ORG	Specify FLDs where originator and destination details can be found	Still applicable
FLD	Field definition	Still applicable
FLS	Sub-field definition for fields in TRA record 'sections'	New in Xe
FMT	File format	Still applicable
GFL, GSR, GTG	Group field definitions	Parsed but ignored
MHD	Defines trigger records	Parsed but ignored
PRM	Constant	Still applicable if used for validation, but cannot be used to specify values

RDF	Redefine	Parsed but ignored
REC	Record definition	Still applicable
RPT	Repeat processing	Parsed but ignored
SEC	TRA record 'section'; corresponds to a composite in EDI	New in Xe
SFL	Group sub-field definition	Parsed but ignored
SRC	Source-only field definition	Treated as a FLD
TGT	Target-only field definition	Treated as a FLD

5.2.3 HDF changes

The extensions that that have been made to the HDF to support Xe are listed below:

5.2.3.1 FMT record

The FMT record supports processing of CSV and TRA files, and an extended range of flat file formats, including SAP IDocs.

It is now mandatory to provide an FMT record in the HDF, with at least the following minimum parameters, to tell Xe the length of the record type and its start position:

```
FMT,F/A/V/X/L/C/T1/T2,,typ-len,typ-start
```

The original definition of the FMT record was:

```
FMT,F/A/V/X,rec-size,typ-len,typ-start,typ-rep,seq,dec-sep,recn-base
```

This has been extended as follows:

```
FMT,F/A/V/X/L/C/T1/T2,rec-size,typ-len,typ-start,typ-rep,seq,
       dec-sep,recn-base,rec-delim,csv-sep,csv-qot,skip-blank,
       tra-rec-char,tra-rec-end
```

The meanings of the file format values are:

- F: Fixed length records (length given by rec-size).
- A (A1, A2, A3): Record delimited data (default delimiter is CR-LF).
- V (V1, V2, V3): MVS variable records (exclude empty fields).
- X: MVS variable records (include empty fields).
- L: Variable record length (length defined on RECs in HDF).
- C: CSV data.
- T1/T2: TRA files (T1 specifies minimal verification of structure on loading the file, T2 specifies full verification).

The meanings of the added parameters are:

- rec-delim: record delimiter where format is 'A' or 'C' (defaults to CR-LF).
- csv-sep: separator character for CSV data (defaults to a comma).
- csv-qot: quote character for CSV data (defaults to single quote).
- skip-blank: ignore wholly blank lines in the source where the format is delimited or CSV.
- tra-rec-char: specify a character to be used as the TRA record type field indicator (the default is '/' as in '/REC-TYPE').

- tra-rec-end: indicate whether the end-of-record field is used in the TRA file (that is, '/END').

In addition where the format is 'C' (for CSV), the type-start parameter is interpreted as the index of the type field in each record, and the type-len parameter is ignored

5.2.3.2 REC record

The original definition of the REC record was:

```
REC,rec-type,write-null,ref,dec-sep
```

This has been extended as follows:

```
REC,rec-type,write-null,ref,dec-sep,h-lev,typ-len,typ-start,
      rec-size,verb-cross-ref
```

The meanings of the added parameters are:

- typ-len: record type field length for this record only; overrides the value on the FMT record if present; ignored if the format is 'C'.
- typ-start: record type field start position for this record only; overrides the value on the FMT record if present; if the format is 'C' it is interpreted as the type field index for this record.
- rec-size: record length; for flat file types where the record length is variable and is given for each record in the definition model.
- h-lev: hierarchical level; allows an implicit structure to be specified for in-house data
- verb-cross-ref: cross reference of a field within the record that represents the record verb (e.g. the `UPDATE` in the record that starts `/GEOADDR, UPDATE`). Usually it will be the first field defined in the record:

```
REC,GEOADDR,,,,,,,,GEO00010
FLD,GEO00010,20,AN
```

5.2.3.3 FLD/FLS record

The original definition of the FLD record was:

```
FLD,ref,length,type,dec-pl,E/I,S/U/T,tbl,tbl-dir,bad
```

This has been extended as follows:

```
FLD,ref,length,type,dec-pl,E/I,S/U/T,tbl,tbl-dir,bad,field-name,
  field-inst,M/C/O,repeats
```

In addition, the FLS record has been introduced to represent a sub-field (within a section). Its parameters are the same as the those in the FLD record. The meanings of the additional FLD parameters are:

- field-name: the name of the field that will appear in the TRA file (in the form `field_name = data value`) – the field name is used to match fields in the instance document with fields in the definition.
- field-inst: the instance number of the sub-field (FLS-only) – see below for an explanation of instance numbers.
- M/C/O: the requirement for the field (mandatory, conditional, optional).
- repeats: the maximum repeat count.

In addition, the TYPE field reference has been extended to support these variations:

- TYPEF – the type field is replaced by filler (white space) in the output file.
- TYPEX – the type field is omitted altogether from the output file.
- TYPER – the type field is right-justified in the output file.

These variations have no effect on loading a flat file.

5.2.3.4 SEC record

The SEC record has been introduced to represent a section in a TRA file:

```
SEC,data-tag,M/C/O,repeats
```

The meanings of the SEC record parameters are:

- data-tag: the name of the section – as sections do not appear directly in the TRA file, the data-tag does not have to have any particular value; instead a name can be chosen that represents the content or purpose of the section.
- M/C/O: the requirement for the section (mandatory, conditional, optional).
- repeats: the maximum repeat count.

5.2.4 EDF and HDF changes for the Mapper GUI

The EDF and HDF have been extended to support the Xe GUI in displaying multiple instances of target entities. For instance, the user wants to map two instances of a segment. Both are populated solely from constants, values &c. It is necessary to show two instances on the GUI, even though they are the same segment and only one is defined in the EDF or HDF. To support this it is now possible to code the following EDF / HDF records with a subscript: `GRP`, `SEG`, `REC`, `DEL`, `FLD`. The only effect of the subscript is to indicate to the GUI that multiple instances of the same entity should be displayed. The structure of the definition model is not affected; only one instance of the entity is created in the model and a property (called `GuiInstances`) is used to indicate the subscripted value. For example:

`SEG[2],NAD,M,0,99,1`	show two instances of the NAD segment
`GRP[5],GROUP1,C,10`	show five instances of the group

Note that the subscript follows the record type (`SEG`, `GRP`), not the entity name (`NAD`, `GROUP1`). It is possible to code subscripts against more than one entity in a hierarchy branch, for example:

```
GRP[2],GROUP5,C,10
SEG,RFF,M,2,1,1
DEL,C506,M
   ,1153,M,RFF00900,03,V,AN
   ,1154,C,RFF00910,35,V,AN
   ,1156,C,RFF00920,06,V,AN
   ,4000,C,RFF00930,35,V,AN
SEG[2],DTM,C,3,1,1
DEL,C507,M
   ,2005,M,DTM00940,03,V,AN
   ,2380,C,DTM00950,35,V,AN
   ,2379,C,DTM00960,03,V,AN
```

This will result in two instances of `GROUP5` being shown, and each will have two instances of the `DTM` segment.

5.2.5 How Xe uses the definitions

In order to use the definitions encapsulated by EDFs and HDFs, Xe creates in-memory representations of them. This definition model is used in processing for a number of purposes:

- When loading EDI and HSE source data for mapping, the definition:
    - Provides the structural information that allows the loaders to populate the source data model (DOM) with the correct hierarchy of data.
    - Provides additional definition information that allows the format of data fields and the number of occurrences of source entities to be validated.
- When mapping to EDI and HSE formats, the definition provides structural information that aids the target DOM builder in constructing the output.
- When writing out EDI and HSE data from a target DOM, the definition:
    - Provides the structural information that allows the writer to validate the hierarchy of the target document.
    - Provides additional definition information that allows the output data fields to be formatted correctly.

The XML definition encapsulated by the XDF is used by Xe as follows:

- When loading XML source data for mapping, the definition provides information about the format of data fields that allows them to be processed and, optionally, validated.
- When mapping to XML formats, the definition provides structural information that aids the target DOM builder in constructing the output.
- When writing out XML data from a target DOM, the definition provides definition information that allows the output data fields to be formatted correctly.

5.3 The Xe DOM

The Xe data model (the DOM) is used to store in-memory representations of files and messages, both when loading source data in preparation for mapping, and when mapping to produce a target file. It supports XML documents, EDI messages and in-house data (including SAP IDocs). The Xe DOM is based on the W3C DOM specification for XML.

As you use the GUI to create a map, Xe develops a mapping script in the form of a text file. The script allows Xe to create a visual mapper in which properties of a map connection relate directly to the parameters of a statement in the script. The script is then used to generate a .NET DLL, which can then be applied to the source file to create the target file.

5.3.1 EDI data

In the case of EDI messages (including VDA messages), nodes are inserted into the DOM for all the different entities (segment, group, composite and element for true EDI, records and fields for VDA) in a way that reflects the structure of the document. The root node represents the message, its child nodes can be segments or groups. The child nodes of groups are segments or nested groups. The child nodes of segments are composites or elements. The

child nodes of composites are sub-elements. Only elements and sub-elements carry data values. An Xe map addresses elements and sub-elements of an EDI message using the Xlate cross-reference in the EDF.

5.3.2 In-house data

In-house data (flat, CSV, TRA) and SAP IDocs are represented in a similar way, but the structure is simpler. There is a single root document node. Its children are record nodes. The children of record nodes are field nodes. Only fields carry data values. An Xe map addresses flat-file fields using the Xlate cross-reference in the HDF.

5.3.3 XML data

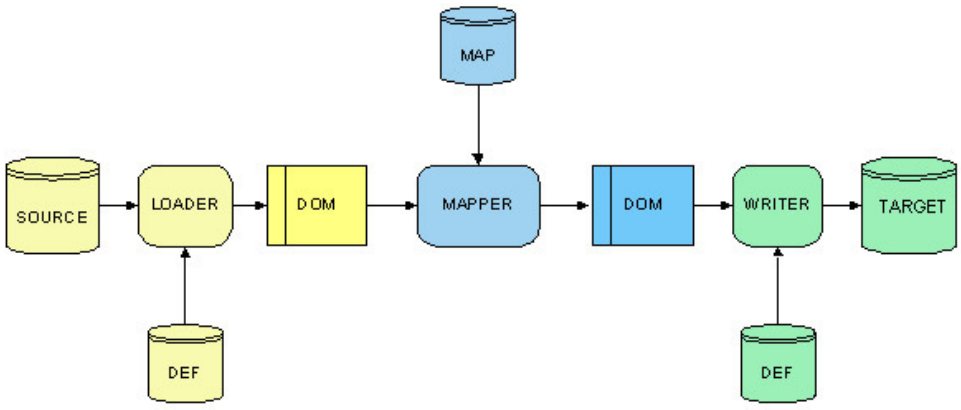
XML data is represented as elements and attributes that relate directly to the structure of the XML document. XML elements may have child elements, attributes or data values. Attributes may only have data values. There is a single root element in each XML document.

5.4 The mapping process

The key features of the mapping process are:

- Source data is loaded into an in-memory data model. Typically, a whole document or message is loaded before any mapping is carried out.
- The data model (the DOM) is a variant on the W3C DOM for XML. It contains some extensions that are Xe-specific and omits some. It is capable of storing XML, EDI, and in-house data.
- Data in the source DOM is mapped to a target data model, which is built up during the mapping process. Typically, a whole document or message is mapped before the data is written out.
- The mapping is carried out by a `.NET` DLL. Each DLL is generated from a mapping script, using the facilities in the `.NET System.CodeDom` namespace.
- Definition models must (unless we are dealing with an XML document) be used to define the structure of messages and documents i.e. when loading the source DOM, when mapping to the target DOM, and when writing out the contents of the target DOM. The definitions used are Xlate EDFs and HDFs (for EDI and in-house data respectively) and XSD schemata for XML.

The following diagram represents the mapping process at the highest level:



The application loads the source file into the source DOM using the source definition that has been specified in the Index.

The Mapper uses the map script to place the data in the target DOM, from where it is written in to the target file using the target definition that has been specified in the Index.

Full details of the Xe index can be found in the section entitled "**Error! Reference source not found.**".

5.5 For existing Xlate users

For existing Xlate users, it may be useful to have a little background information about the new concepts behind the Xe Mapper and how Xe differs from Xlate.

Xe offers all the existing capabilities of Xlate mapping and more besides, and makes their use much more intuitive.

To Xlate, the source and target file definitions (EDF and HDF) defined both the expected content of the respective files, and the map connections between them. To Xe, file definitions define only file content; the map has to be defined separately.

As previous users of Xlate, you will know that there is usually more to file mapping than a simple one-to-one mapping. Xe provides you with the means to achieve all the complex mapping that Xlate could do, plus a few more tricks besides. Not only does the Xe GUI make complex mapping easier to achieve, it also provides you with a visual representation of the mapping, making it easier to check.

As Xlate loads data, it maps it and then flushes it (with a few exceptions), which can cause problems with re-ordering of data, and particularly with changing data hierarchies.

By contrast, Xe loads the source file into a Document Object Model (DOM) in memory. As a consequence, source data may be addressed out of sequence, so the application controls what source data is to be read and in what order.

Although Xe uses some of the same conventions as Xlate in the definition of the source and target files, such as Level, Repeats and Requirement for an EDI segment, it does not use any of the manipulative techniques in the same way.

For example, where Xlate used the PLS and PLE records to process data in "segment pools", which allowed segments to be checked for specific qualifiers, this can be achieved in Xe by using the Conditions facility, whereby a source data element will only be processed if it, or a specified related source data element, meets the specified conditions. You test for the conditions in a simple line of program code.

The drag-and-drop technique also makes it simpler to map a single source field to several target fields.

In short, the GUI allows you to have more control of the data and to achieve more flexibility than with Xlate.

5.6 Source file definition in the GUI

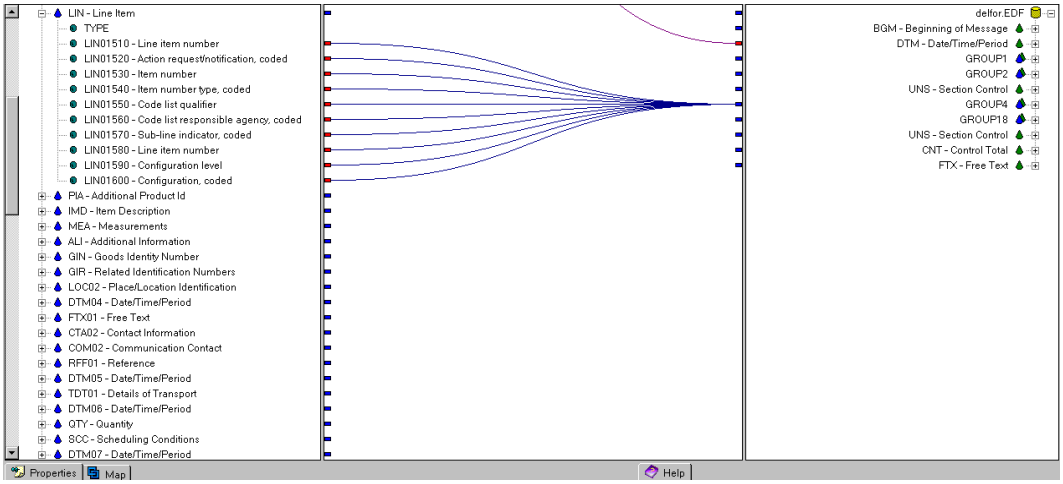
A visual representation of the source file definition is displayed in the top left panel of the window. If you have loaded a map file, the map file contains a reference to the definition of the source file, i.e. an EDF, HDF or XDF file, which is then displayed automatically.

Initially, this area simply shows the source file definition in tree view format, with none of its nodes expanded. Each node represents a single record or, in the case of an EDI file definition, a single segment or segment group.

Click on the plus-sign alongside any of the nodes to see the expanded node and its sub-nodes.

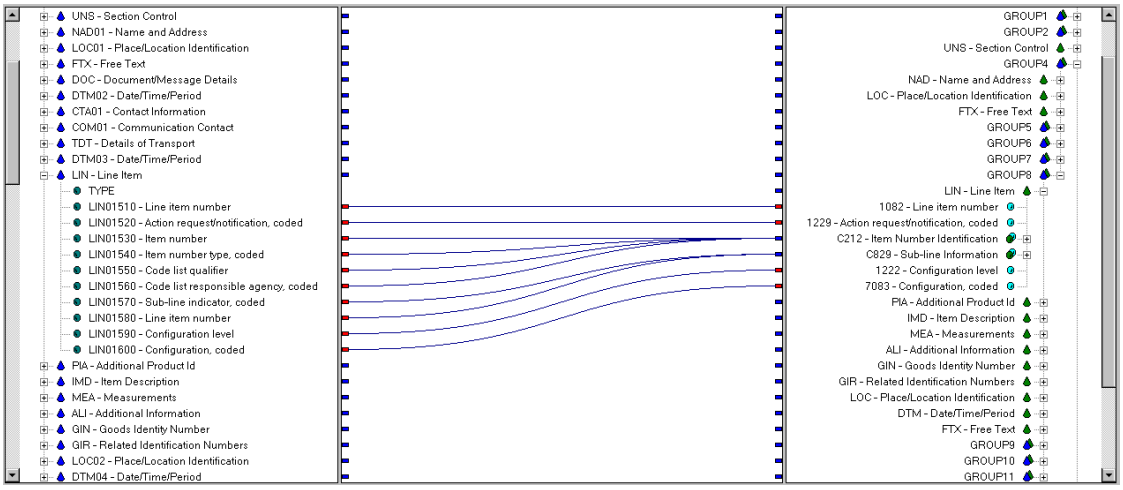
In the case of an HDF file, each sub-node represents a field within the record. In an EDF file, each sub-node may represent a segment, a composite element or a simple element, depending on what the expanded node represents.

An example, with mapping links, is shown below. The source is an HDF and the target is an EDF.



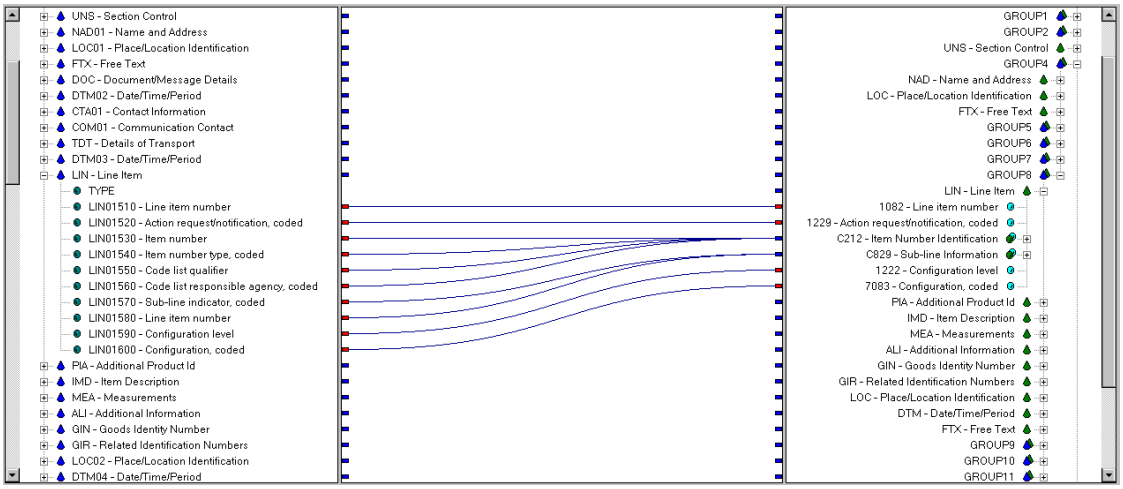
In this example, the LIN record of an HDF has been expanded to show all the fields in that record. The link from each field in the LIN record to its target in the target file is represented by a blue line, which terminates at the appropriate connector (i.e. the connector associated with the appropriate target node) on the right hand side. The example shows each blue line terminating at a single connector. This is because the target node has not been expanded to show the individual fields or data elements.

A further example, with the target node expanded, is shown below.



Here you can see more clearly how each source node maps to each target node.

You will see that there are still several source nodes which link to the same target node. This is because the target is an EDI file definition containing composite elements, and those source fields are mapped to the data elements within the composite. A further example will show the mapping links when the composite node in the target file is expanded.



Here you can clearly see that each source node maps to an individual target node.

5.7 Target file definition in the GUI

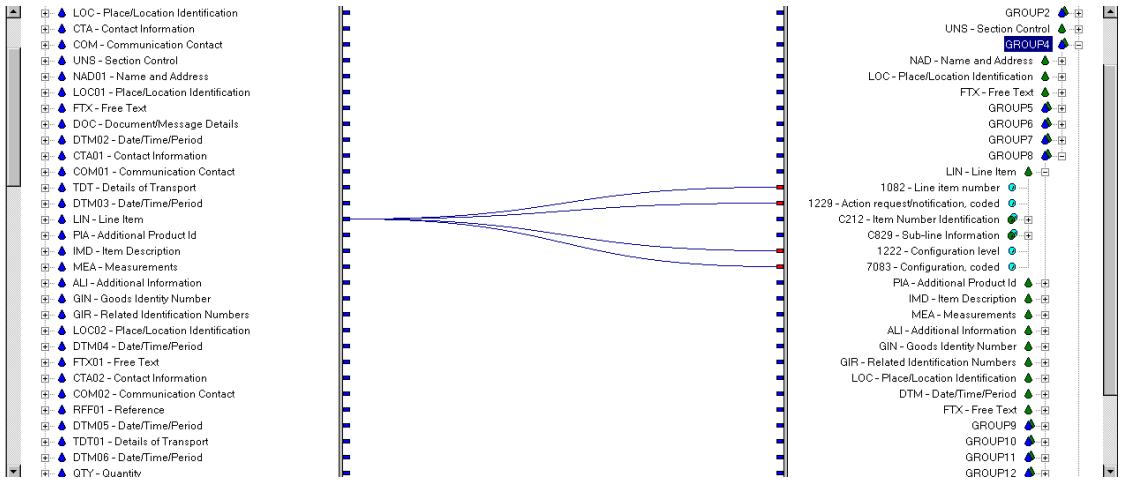
A visual representation of the target file definition is displayed in the top right panel of the window. If you have loaded a map file, the map file contains a reference to the definition of the target file, i.e. an EDF, HDF or XDF file, which is then displayed automatically.

Initially, this area simply shows the target file definition in tree view format, with none of its nodes expanded. Each node represents a single record or, in the case of an EDI file definition, a single segment or segment group.

Click on the plus-sign alongside any of the nodes to see the expanded node and its sub-nodes.

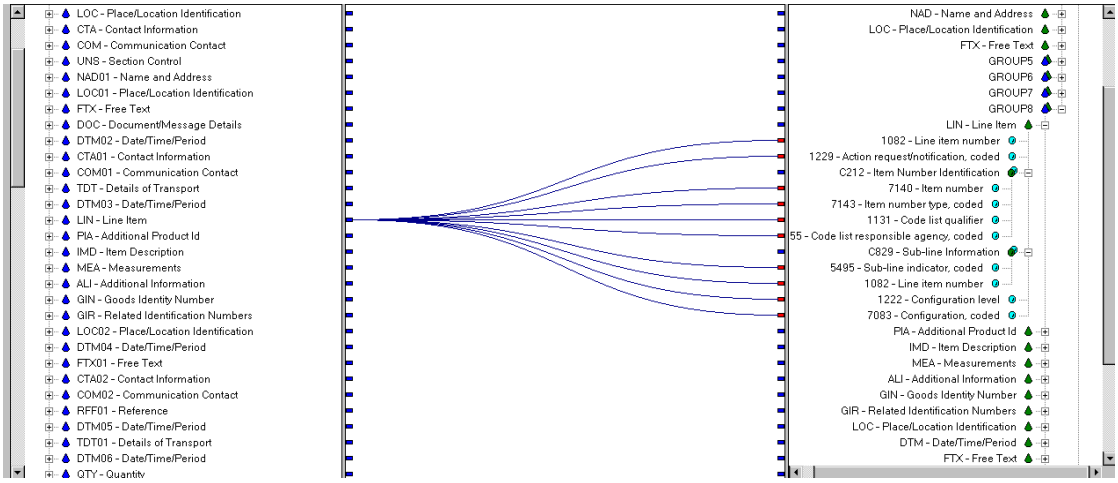
In the case of an HDF file, each sub-node represents a field within the record. In an EDF file, each sub-node may represent a segment, a composite element or a simple element, depending on what the expanded node represents.

An example, with mapping links, is shown below. The source is an HDF and the target is an EDF.



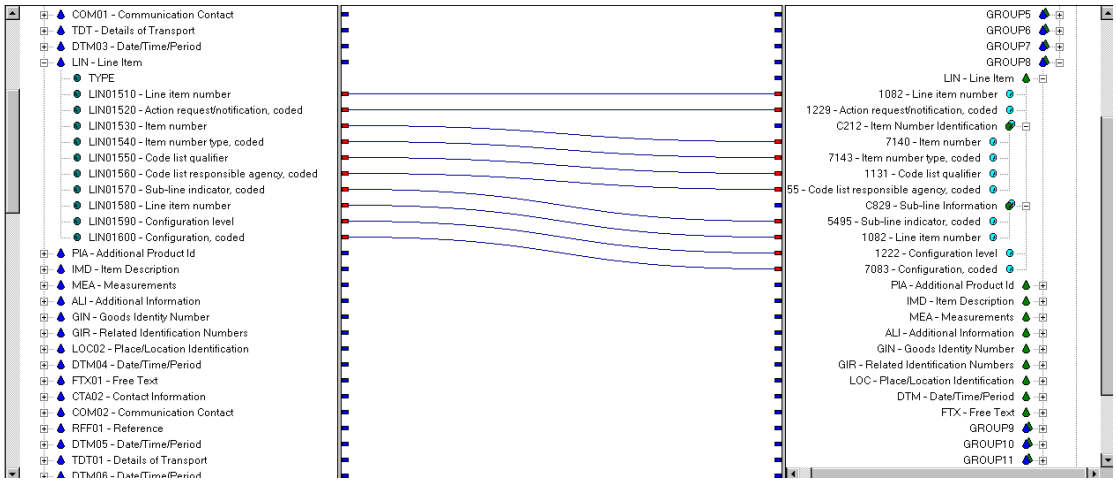
In this example, the LIN segment of an EDF has been expanded to show its simple and composite data elements. The simple data elements are shown with a link to their (unexpanded) source node in the source file. The composite elements themselves are not linked to the source file and so no link is displayed for them.

If we expand the composite elements of the LIN segment in the target file, the connections will expand as shown below.



In the example above, the LIN segment has been expanded to show all the data elements (simple and composite) in that segment. The link from each data element in the LIN segment to its source in the source file is represented by a blue line, which terminates at the appropriate connector (i.e. the connector associated with the appropriate source node) on the left hand side. The example shows each blue line terminating at a single connector. This is because the source node has not been expanded to show the individual fields or data elements.

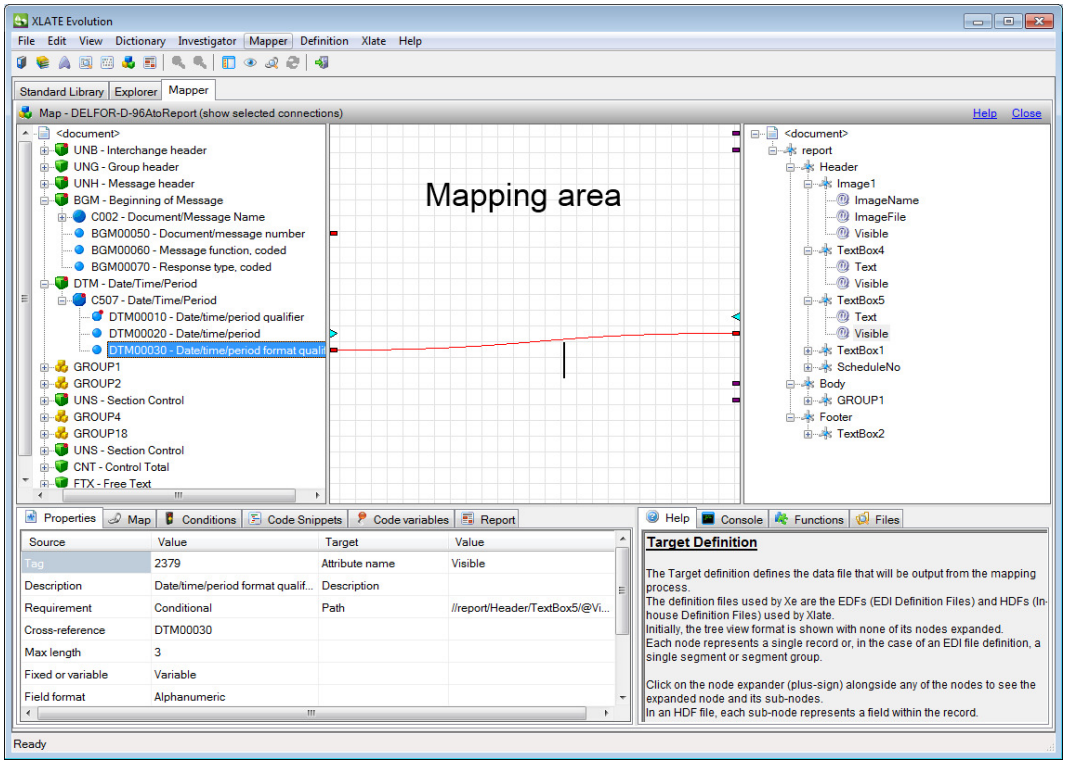
A further example, with the source node expanded, is shown below.



Here you can see more clearly how each source node maps to each target node.

5.8 Mapping area of the GUI

Shown below is the Xe GUI, showing the mapping area.



At its simplest, the mapping area is blank apart from the small blue connectors, which show where mapping links exist. Once you begin to expand the nodes in the source file or target file areas, the mapping area will display the existing links between the nodes on either side. These connectors will be displayed in red, with blue link lines. However, if you highlight one of these linked nodes, its link line will turn red.

Clicking on the mapping area will bring up a set of page tabs in the Properties and Mapping section (bottom left section of the window) which can be used in the processing or manipulation of the data prior to it being inserted in the target file.

5.8.1 Basic mapping

You should be able to perform most mapping via the simple drag-and-drop technique. This creates a one-to-one mapping from the source to the target.

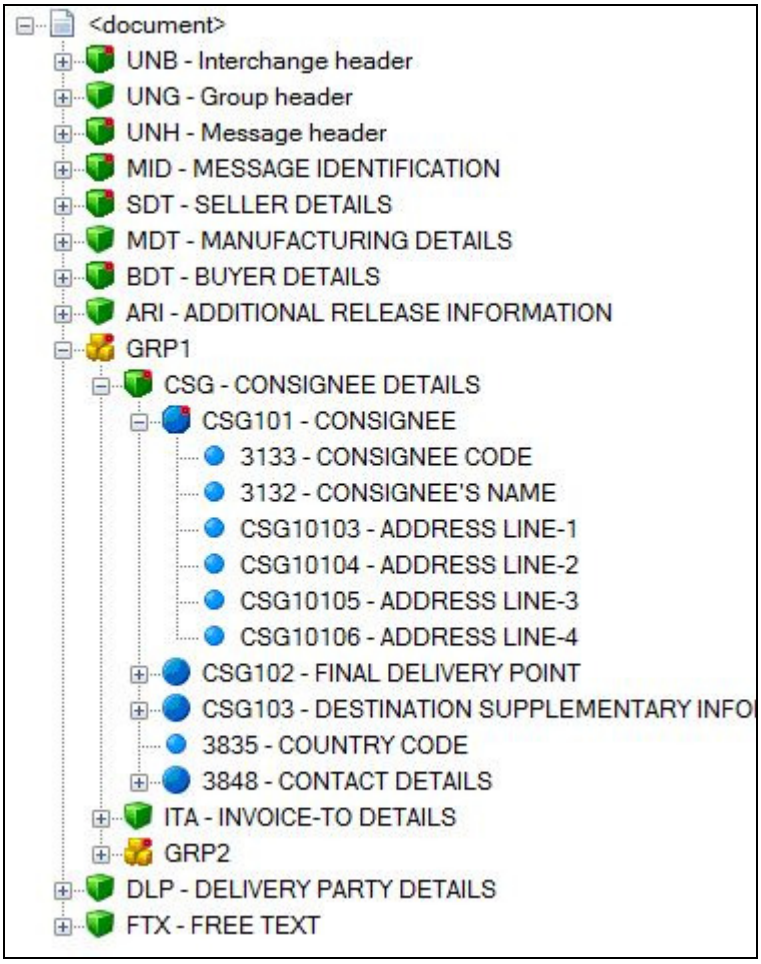
You can also create a mapping from a single source node to multiple target nodes, also via the simple drag-and-drop technique.

Drag-and-drop also allows you to map out of hierarchical sequence. All you need to do is drag the required source node to the required target node, and Xe will work out the hierarchical requirements for you.

If you want to perform any processing on the data before or after you map it, you will need to use the global and/or local page tabs provided in the Properties and Mapping area. The following sections explain how to use these page tabs and some common examples are provided.

5.8.2 Mandatory entities

Source and target definition files allow some entities (such as segments, elements and fields) to be marked as mandatory. Mandatory items are shown in the tree views using red targets at the top right or top left of the icons in the tree:



So the group named GRP1, the segment CSG and the composite element CSG101 are all mandatory, but the segment ARI and the element 3133 are not.

5.8.3 Working with multiple connections

Xe provides a mechanism for mapping more than one node in a single drag-and-drop actions. If you drag from a certain source nodes to certain target nodes while holding down the **Shift** key, more than one map will potentially be created. The table indicates what is mapped depending on the source/target node type:

Node type	Mapped nodes
Segment (EDF)	All mappable descendant nodes are mapped. That is, all simple elements and sub-elements of composite elements within the segment that are not program variables.
Record (HDF)	All mappable child nodes are mapped. That is, all fields within the record that are not type or filler fields.
XML element	All mappable attributes of the element are mapped. That is all attributes that are not namespace declarations (xmlns).

So, for instance, dragging a source segment with four elements to a target record with five fields results in the following connections being made.

```
SEG                    REC
 ELEMENT 1       -->    FIELD 1
 ELEMENT 2       -->    FIELD 2
```

```
COMPOSITE
  SUB-ELEMENT 3    -->    FIELD 3
  SUB-ELEMENT 4    -->    FIELD 4
                          FIELD 5
```

The last field is not mapped to because there is no source. This mechanism can only be used where the required maps are to mirror the source in the target precisely.

It is also possible to delete multiple map connections in one action. Right-clicking with the mouse on a tree node causes a context menu to appear. When right clicking on a segment record or XML element, one option will be to delete map connections. If this options is selected all connections to or from mapped descendant node will be deleted. For instance, in the example above, right-clicking on the REC node and selecting the delete option will result in the four connections shown being deleted.

5.8.4 Shortcuts

Xe provides two mapping shortcuts for you, for nodes you want to map from or to but without using a direct one-to-one mapping.

5.8.4.1 Source shortcut

This shortcut simplifies the process of mapping a source node to a local string variable so that you can perform some processing on the source data. This should be used when you are not mapping the source node directly to the target node.

Right-click on the source node you wish to map from, then click on the **New map connection** button which pops up. This will automatically create an entry on the Map page tab, containing the following:

Source type = Source node

Source value = the name of the source node you clicked on

Target type = Readvar

Target value = blank for you to provide the name of a new string variable which will be used in a POST code snippet

5.8.4.2 Target shortcut

This shortcut simplifies the process of mapping a literal value to a target node.

Right-click on the target node you wish to map to, then click on the **New map connection** button which pops up. This will automatically create an entry on the Map page tab, containing the following:

Source type = Value (i.e. a literal value)

Source value = blank for you to provide the appropriate value

Target type = Target node

Target value = the name of the target node you clicked on

5.9 Properties and Mapping

The Properties and Mapping section of the Mapper is found in the bottom left of the window. The contents of this section depends on whether you have clicked on the source or target or mapping area, and the type of node you have highlighted in the source or target area.

5.9.1 Properties

The Properties page will always be present, no matter where you have highlighted.

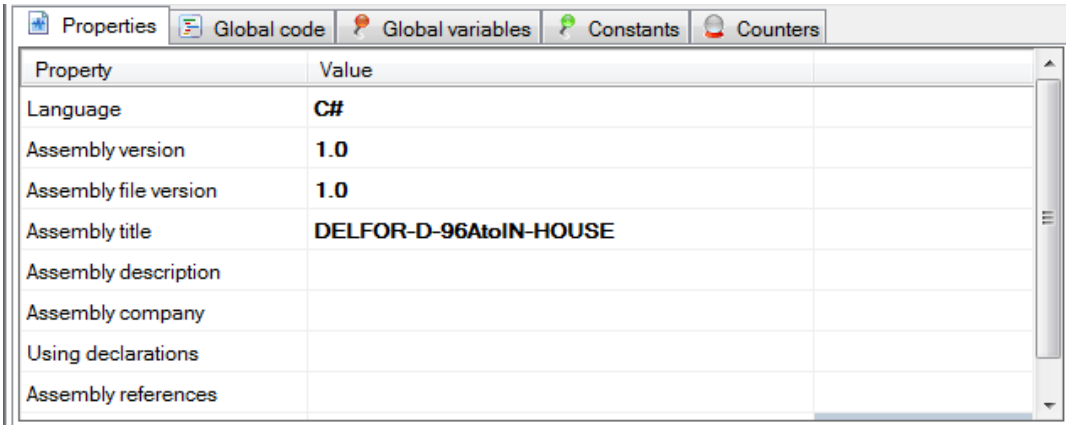
Click on the mapping area to see the page tabs related to global processing i.e. items on these page tabs affect or can be used by the whole map.

Click on any node in the source or target area to see the properties for the selected node.

If the node you select maps to a node in the other section (i.e. the link line is displayed), then properties for both source and target will be displayed. Otherwise only one set of properties will be displayed, relating to the node you have selected.

5.10 Global properties and mapping

Click your mouse on the upper central section of the Mapper (the mapping area) to view the global page tabs, as shown below.



There are 5 global page tabs: Properties, Global code, Global variables, Constants and Counters. Each is described in its own section below.

5.10.1 Properties

The Properties page tab displays a list of properties associated with this map, together with their values.

Property	Description
Language	The programming language in which code snippets for the mapping script are to be written. May be C# or VB.NET.
Assembly version	The version of the DLL created by this map, which you may edit. Although only 2 positions are shown by default, you may provide up to 4 positions e.g. 1.1.1.1 This parameter sets the assembly metadata 'Assembly version' property, which appears on the version tab of the file properties dialog in Windows.
Assembly file version	The file version of the DLL created by this map, which you may edit. Although only 2 positions are shown by default, you may provide up to 4 positions e.g. 1.1.1.1 This parameter sets the

	assembly metadata 'File version' property, which appears on the version tab of the file properties dialog in Windows.
Assembly title	The name you have given to this DLL, taken from the Project Name field of the Edit Map Project dialog. This parameter sets the assembly metadata 'Description' property, which appears on the version tab of the file properties dialog in Windows.
Assembly description	A description of this DLL, which you may edit. This parameter sets the assembly metadata 'Comments' property, which appears on the version tab of the file properties dialog in Windows.
Assembly company	The company associated with this DLL. This parameter sets the assembly metadata 'Company name' property, which appears on the version tab of the file properties dialog in Windows.

Using declarations	A list of namespaces that are to be referenced in code snippets.
	By default, only the `System` namespace is available to the writer of code snippets without using the fully qualified name. For instance, to use a stream writer in code snippet, you must write:
	`System.IO.StreamReader sr = new System.IO.StreamReader("file");`
	This can be simplified by declaring `USING="System.IO"` in the `GLOBAL` statement and coding:
	`StreamReader sr = new StreamReader("file");`
	The namespaces referenced in this way can be from any assembly that is available to Xe (by default, or by explicit reference – see Assembly references below) including `.NET` class library namespaces and the user's own.

Assembly references	A list of system assemblies that are referenced by the generated map DLL. The `.NET` assemblies referenced by default are: `System.dll`, `System.XML.dll` and `System.Data.dll`. To use code from any other assembly (`.NET`, third party or your own) in code snippets, the assembly must be referenced using this parameter.

5.10.2 Global code

There are 3 types of global code that can be used in the Mapper.

DEFN

These are definitions, performed once only, when the Mapper is run i.e. on Mapper construction.

DEFN allows you to declare global variables that can be used during the mapping process via references in code snippets.

You can also use the DEFN code to define and initialise a database connection.

The example below shows the declaration of 3 global variables, and the definition and initialisation of a database connection.

Example

```
int iTotalPart1;
int iTotalPart2;
int iTotalPart3;
OleDbConnection connection = null;
```

INITMAP

This is for initialisation, performed before map execution.

INITMAP allows you to initialise any global variables you have declared in the DEFN statement.

You would need to use INITMAP if you want to create a connection to and open a database.

You should ensure that error handling is also included.

The example below shows the initialisation of the 3 global variables that were declared in the DEFN section, the declaration and initialisation of a string for the DB path, code to create a connection to the DB and to open it, with error handling around the DB code.

Example

```
iTotalPart1 = 0;
iTotalPart2 = 0;
iTotalPart3 = 0;
string strConn = "Provider=Microsoft.Jet.OLEDB.4.0; Data
Source=\"lookup.mdb\"";
try
{
   connection = new OleDbConnection(strConn);
   connection.Open();
}
catch(Exception ex)
{
   connection = null;
   XE.Log("RECAD01E", "Failed to open database connection - " + ex.Message);
}
```

ENDMAP

This is for finalisation, performed after map execution.

This area can be used to close the database correctly and to perform final processing of the global variables. It provides the opportunity to clean up resources that were created by code snippets during mapping.

The example below shows the closing and disposing of the DB connection (if it is open) and logging to show the final values of the 3 global variables that were created in the DEFN section.

Example

```
if (connection != null)
 {
```

```
    connection.Close();
    connection.Dispose();
}

XE.Log("RECAD05I", "Total part #1 - " + iTotalPart1);
XE.Log("RECAD05I", "Total part #2 - " + iTotalPart2);
XE.Log("RECAD05I", "Total part #3 - " + iTotalPart3);
```

5.10.3 Global variables

The Global variables tab has a **New** and a **Delete** button.

The **New** button allows you to create new global variables which can be used to manipulate data before it is mapped to the target. They will be created with the name new_codevar1, new_codevar2 etc. You should change this to a more meaningful name and provide the data type for the variable e.g. string, int.

You can also give each variable an initial value, if required, using the Initial Value column. Otherwise it will default to an empty string.

The **Delete** button can be used to delete any existing variables defined in this area.

For a list of all the code variable types that can be used, please refer to the section entitled “Code variable types”.

5.10.4 Constants

This tab can be used to define any global constant values for the map, such as EDI codes, Trading Partner codes, message names etc.

Click the **New** button to create a new constant. Give it a meaningful name and provide its value in the Value column.

The **Delete** button can be used to delete any existing constants defined in this area.

5.10.5 Counters

This tab can be used to define global counters. Counters are used to map an incremented number to the target file, for example to EDI segments that contain a count of the number of their occurrence.

Please note that each counter is incremented each time it is mapped and is valid throughout one map only. If you want to reset a value to zero before the end of a map, or if you need to keep a running total or manipulate the value, you should not use a Counter for this. Instead use the Global Code page to define one or more integers for these purposes.

Similarly, if you require a counter that is valid across more than one map, you should use the Index Counters facility.

Click the **New** button to create a new counter.

Type in a name for the counter and provide its initial value if you do not want to use the default. The default initial value is 1.

The initial value is the first value that will be used i.e. it will be used and then incremented.

For example, a counter which has an initial value of 0 and an increment value of 10 would create counters as follows: 0, 10, 20, 30 ……

A counter which has an initial value of 1 and an increment value of 1 would create counters as follows: 1, 2, 3, 4 ……

Type in an increment value if you do not want to use the default. The default increment value is 1.

You only need to specify a mask if you want the counter to consist of more than just an incremented number, or if you want to specify how many digits the counter must have.

The mask should include a placeholder for the counter, with a reference to the number of digits it has. A counter with 5 digits should be specified as %CNT:5% (please note the colon character ":" after the CNT characters.

Before and after this placeholder you can specify other constants which make up the counter, such as a string of characters or digits.

A counter consisting of 5 digits prefixed by the letter A and followed by the letter Z, whose initial value is 100 (i.e. A00100Z) would require a mask of A%CNT:5%Z

Highlight an existing counter and click the **Delete** button to delete a counter.

5.11 Source and Target properties and mapping

Now let's have a look at the contents of the Properties and Mapping section, to find out how to understand a map.

The Properties page tab of the Properties and Mapping section will show different items of information depending on what type of item is highlighted in the upper section of the GUI and what type of mapping is being performed (e.g. EDI to CSV or SAP IDoc to EDI).

Click on any node in the source or target area to see the properties for the selected node.

If the node you select maps to a node in the other section (i.e. the link line is displayed), then properties for both source and target will be displayed. Otherwise only one set of properties will be displayed, relating to the node you have selected.

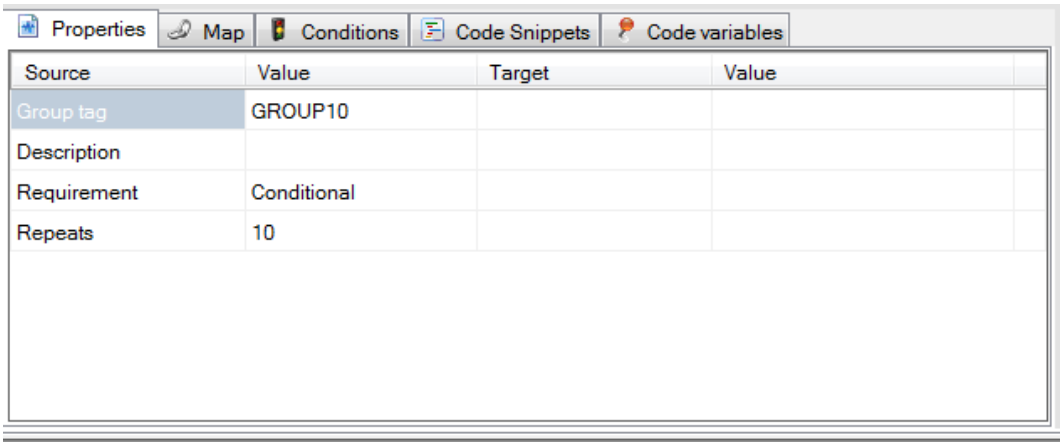
This section describes what the Properties page tab contains in each instance.

5.11.1 EDF Properties

If the source or target file is an EDF, the properties and values are described below.

5.11.1.1 Group Properties

If you have highlighted a source file group in the upper section of the GUI, the Properties page tab will look like the example below, where the Source and Value columns are populated. If you have highlighted a target file group, the Target and Value columns will be populated with these items instead.

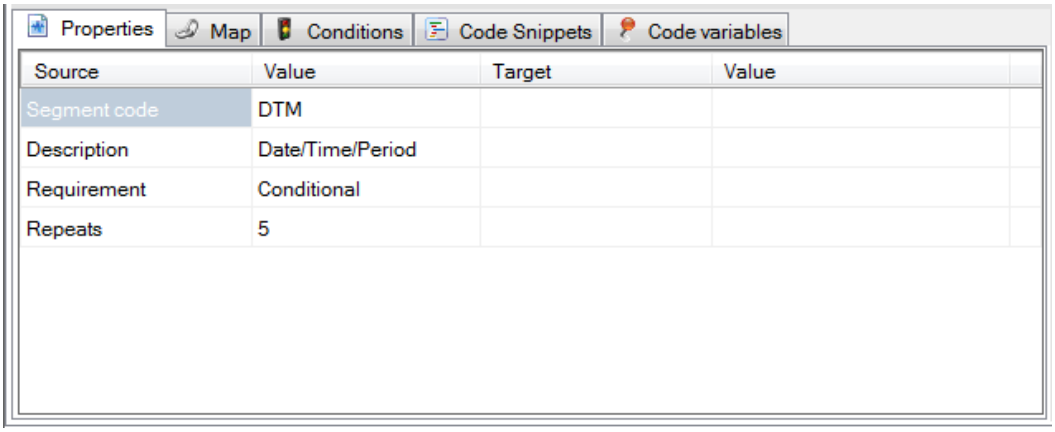


Below is a list of all the Source/Target properties for an EDF group. The Xlate equivalent is included for the benefit of those users familiar with Xlate.

Source/Target property	Description	Xlate equivalent
Group tag	The name of the Group	GRP (LOP), group-tag
Requirement	Indicates whether the Group is Mandatory or Conditional within the message.	GRP (LOP), M/C
Repeats	Indicates the maximum number of times this group of segments may occur within a message.	GRP (LOP), loops
Description	A brief description of the Group.	N/A

5.11.1.2 Segment Properties

If you have highlighted a source file segment in the upper section of the GUI, the Properties page tab will look like the example below, where the Source and Value columns are populated. If you have highlighted a target file segment, the Target and Value columns will be populated with these items instead.



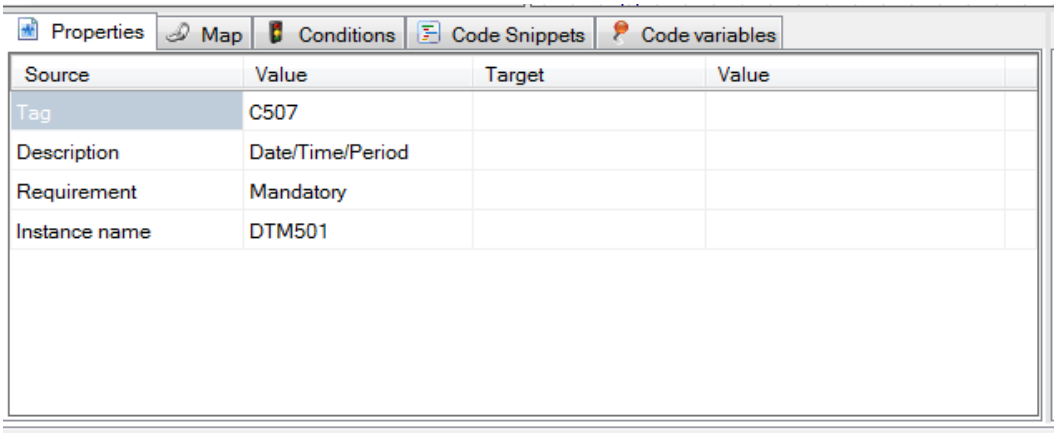
Below is a list of all the Source/Target properties for an EDF segment. The Xlate equivalent is included for the benefit of those users familiar with Xlate.

Source/Target property	Description	Xlate equivalent
Segment code	The name of the segment as in the message definition.	SEG record, segment-tag
Description	A brief description of the segment as in the message definition.	N/A
Requirement	Indicates whether the segment is Mandatory or Conditional within the message.	SEG record, M/C
Repeats	Indicates the maximum number of times this segment may occur	SEG record, repeats

	within a group or message.	

5.11.1.3 Composite Properties

If you have highlighted a source file composite element in the upper section of the GUI, the Properties page tab will look like the example below, where the Source and Value columns are populated. If you have highlighted a target file segment, the Target and Value columns will be populated with these items instead.



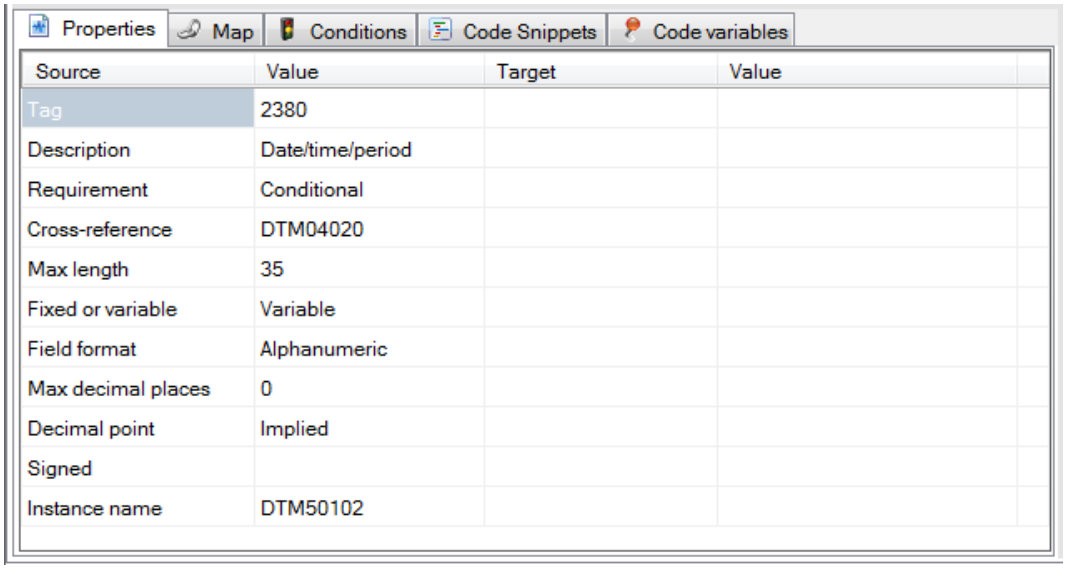
Below is a list of all the Source/Target properties for an EDF segment. The Xlate equivalent is included for the benefit of those users familiar with Xlate.

Source/Target property	Description	Xlate equivalent
Tag	The name of the composite element as defined by the standards organisation.	DEL record, data-tag
Description	A brief description of the composite element as in the message definition.	N/A
Requirement	Indicates whether the composite element is Mandatory or Conditional within the segment.	DEL record, M/C
Instance name	Unique identifier assigned to the source/target node. This property should not be referenced by the user, as it is liable to lead to unexpected results if future changes are made to the source definition.	N/A

5.11.1.4 Data element properties

If you have highlighted a data element of a source EDF file in the upper section of the GUI, the Properties page tab will look similar to the example below. This example shows details of a source data element that is not mapped to a target

file. Please note that the source properties would be target properties if the example showed a target data element.



The Xlate equivalent is included for the benefit of those users familiar with Xlate.

Source/Target properties of an EDF

Source/Target property	Description	Xlate equivalent
Tag	The name of the data element as defined by the standards organisation.	DEL record, data-tag
Description	A brief description of the data element as in the message definition.	N/A
Requirement	Indicates whether the data element is Mandatory or Conditional within the segment.	DEL record, M/C
Cross-reference	The name of the field in the target file to which this element should be mapped.	DEL record, ref
Max length	The maximum length of the data element.	DEL record, length (min, **max**)
Fixed or variable	Indicates whether the data element is fixed length or variable length.	DEL record, F/V
Field format	Defines the data element's contents.	DEL record, type
Max decimal places	For numeric data elements, this is the maximum number of decimal places (i.e. the number of numeric digits	DEL record, dec-places (min, **max**)

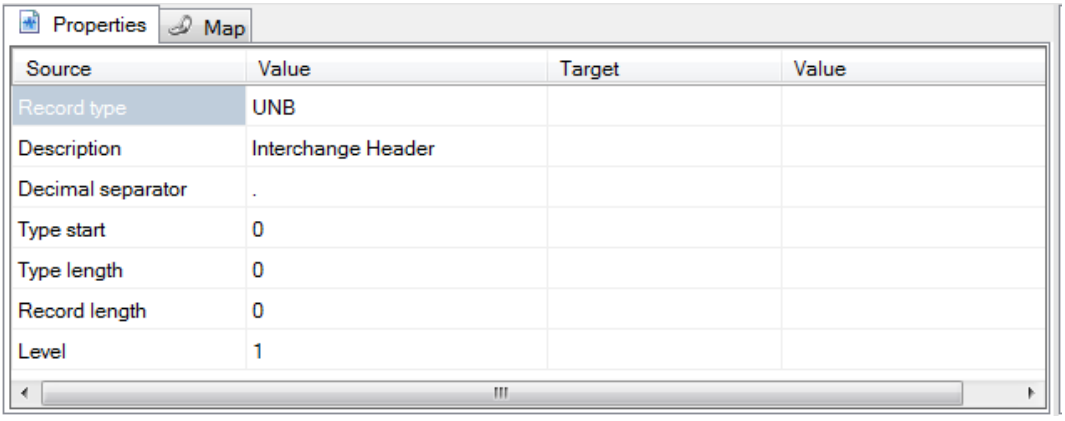
	to the right of the decimal position).	
Decimal point	For numeric elements, this indicates whether the decimal point is Implicit or Explicit.	DEL record, E/I
Signed	For numeric elements, this indicates whether the data is Signed or Unsigned.	DEL record, S/U
Instance name	Unique identifier assigned to the source node. This property should not be referenced by the user, as it is liable to lead to unexpected results if future changes are made to the source definition.	N/A

5.11.2 HDF Properties

If the source or target file is an HDF, the properties and values are described below.

5.11.2.1 Record Properties

If you have highlighted a source file record in the upper section of the GUI, the Properties page tab will look like the example below, where the Source and Value columns are populated. If you have highlighted a target file record, the Target and Value columns will be populated with these items instead.



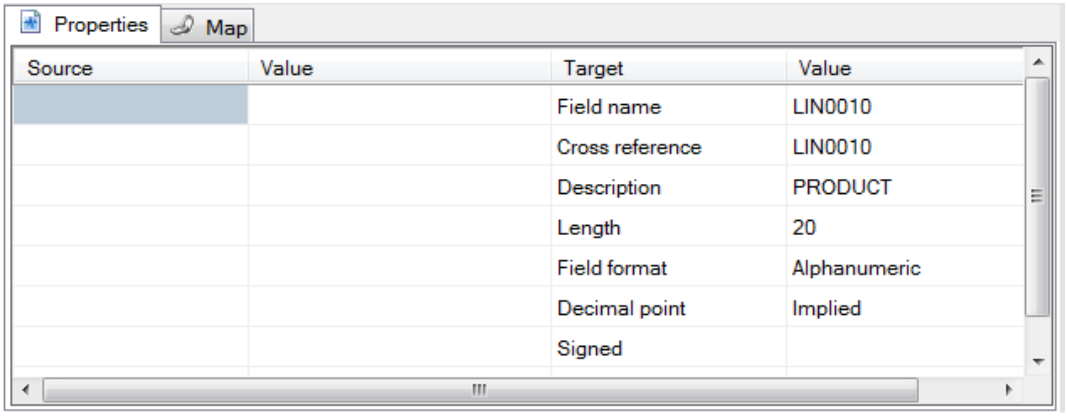
Below is a list of all the Source/Target properties for an HDF record. The Xlate equivalent is included for the benefit of those users familiar with Xlate.

Source/Target property	Description	Xlate equivalent
Record type	The record type of the source/target record.	REC record, rec-type
Description	A brief description of the record.	N/A
Decimal separator	The value of the decimal	REC record, dec-

	separator character for numeric fields in this record.	separator
Type start	The position of the record type in the record. Default is 0 i.e. the first character of the record.	FMT record, type-start
Type length	The number of characters in the record type.	FMT record, type-length
Record length	The record length of every record. Only applicable to fixed length records.	FMT record, rec-size

5.11.2.2 Field properties

If you have highlighted a field of a target HDF file in the upper section of the GUI, the Properties page tab will look similar to the example below. This example shows details of a target field that is not mapped from a source file. Please note that the target properties would be source properties if the example showed a source field.



Source/Target properties of an HDF

Below is a list of all the Source/Target properties for an HDF field. The Xlate equivalent is included for the benefit of those users familiar with Xlate.

Source/Target property	Description	Xlate equivalent
Field reference	The name you have assigned to the field for mapping purposes. See notes below	FLD record, ref
Description	A brief description of the field.	N/A
Length	The length of the field.	FLD record, length
Field format	Defines the field's contents	FLD record, type
Decimal point	For numeric fields, this indicates whether the	FLD record, E/I

	decimal point is Implicit or Explicit.	
Signed	For numeric fields, this indicates whether the data is Signed or Unsigned.	FLD record, S/U
Instance name	Unique identifier assigned to the target node. This property should not be referenced by the user, as it is liable to lead to unexpected results if future changes are made to the target definition.	N/A

Field reference

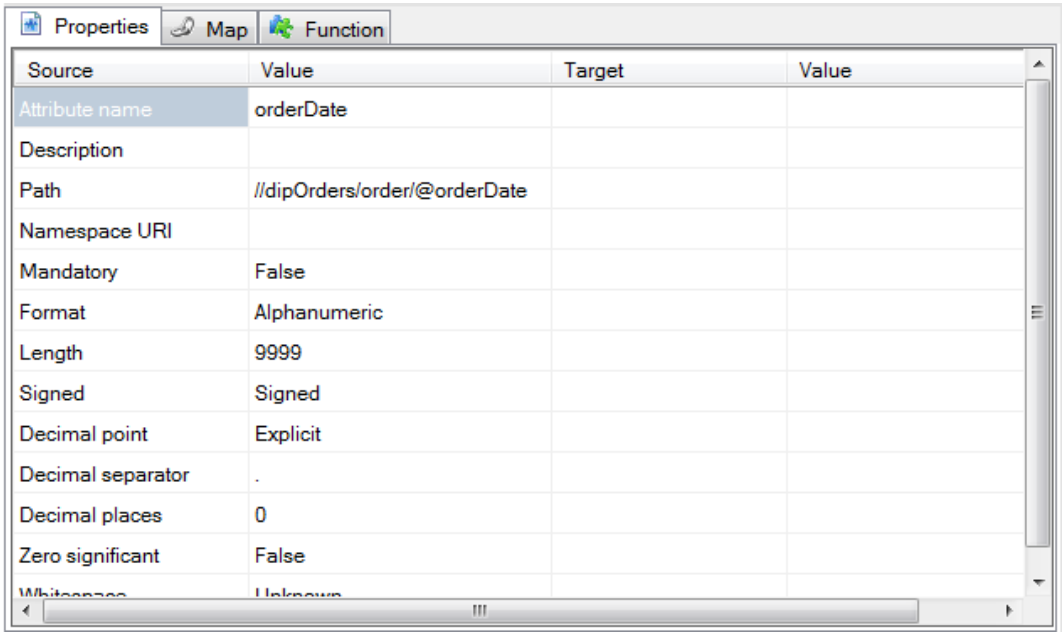
As in Xlate, the Field Reference parameter may be given the special value of TYPE. This defines the field as the record type and means that it will not be linked to the source file.

5.11.3 XDF Properties

If the source or target file is an XDF, the properties and values shown are described below.

5.11.3.1 Element Properties

If you have highlighted a source file element in the upper section of the GUI, the Properties page tab will look like the example below, where the Source and Value columns are populated. If you have highlighted a target file element, the Target and Value columns will be populated with these items instead.



Below is a list of all the Source/Target properties for an XDF record.

Source/Target property	Description

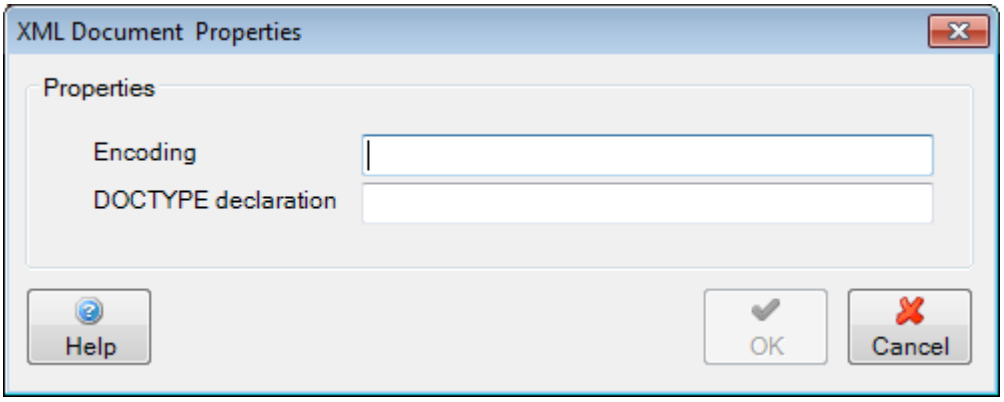
Element name	Full name of the element (including namespace prefix if there is one)
Description	A user-defined text description
Path	An expression representing the absolute path of the element in the XML instance document
Namespace URI	The URI of the target namespace to which the element belongs
MinOccurs	The minimum number of repeats of the element that are required.
MaxOccurs	The maximum permitted number of repeats of the element
Format	The data-type expected (numeric, (alphanumeric, &c)
Length	The maximum length of the data value
Signed	Whether the value is signed
Decimal point	Whether the decimal point is implied or explicit
Decimal separator	The separator used to separate integer and decimal parts of a number
Decimal places	The number of decimal places (number of implied decimal places if the decimal point is implied, otherwise the maximum permitted number of decimal places)
Zero significant	Whether zero values are significant (if true then a zero in the element means that the element value is present but has a value of zero; if false then an element with a zero in it is treated as an element with no value)

5.11.3.2 Attribute properties

Properties of XML attributes are the same as those for elements, with the exception that the MinOccurs and MaxOccurs properties are not present. Instead there is a single true-or-false Mandatory property used to indicate whether the attribute must appear.

5.11.4 XML Document Properties

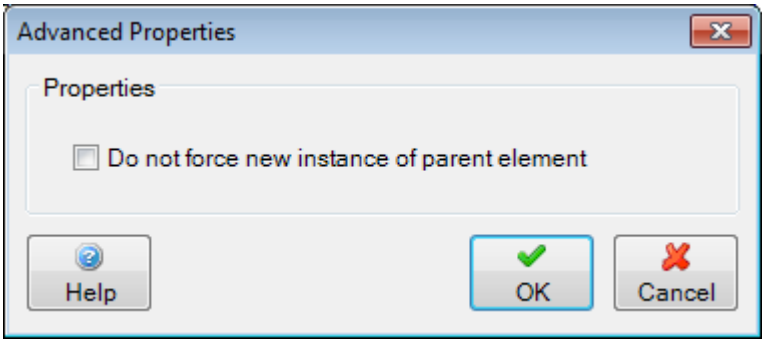
When mapping to XML it is possible to set properties that apply to the document as a whole.



To display the dialog, right click on the root (`<document>`) XML element in the right-hand (target) tree view, and choose the Document Properties option. The **Encoding** text box can be used to enter an encoding name to appear in the target document prolog. The **DOCTYPE Declaration** text box can be used to enter a literal `!DOCTYPE` node that will be written to the target document.

5.11.5 XML Advanced Properties

In addition to the properties displayed in the property pages, an advanced properties dialog is available when mapping to XML.

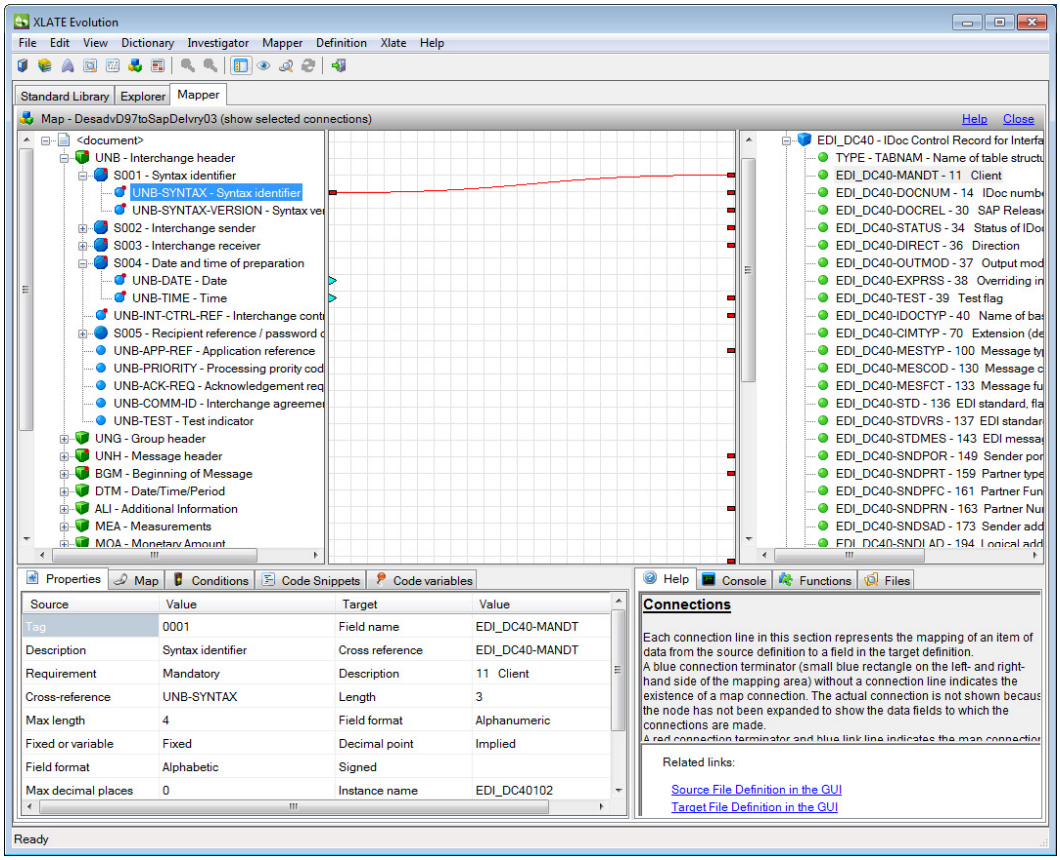


To display the dialog, right click on any XML element in the right-hand (target) tree view, and choose the Advanced Properties option. The check box can be used to indicate that a new instance of an element (should or) should not cause an instance of its parent automatically to be created.

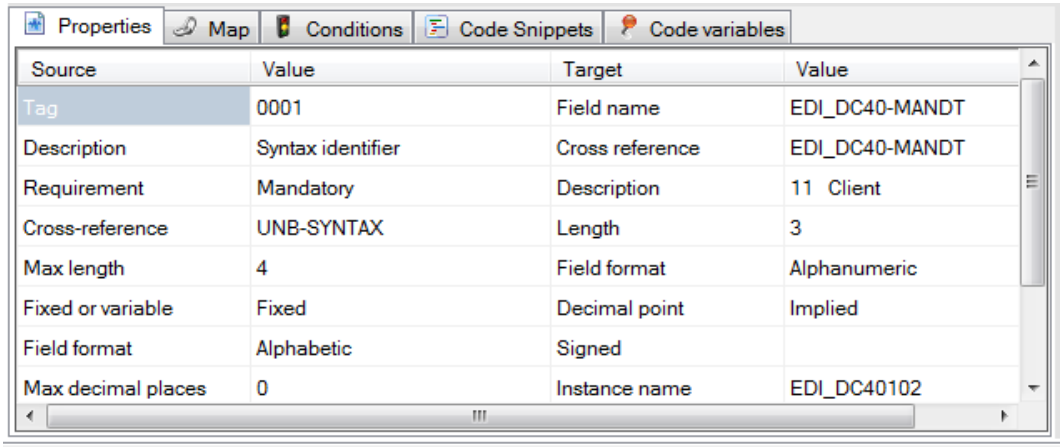
5.11.6 Map

The Map page tab displays information about the mapping link, if one exists.

To see existing link information, select a source or target node at element or field level, or a connection line. The results will be similar to the example below.



The Map page tab itself for this mapping link will look like the example below.

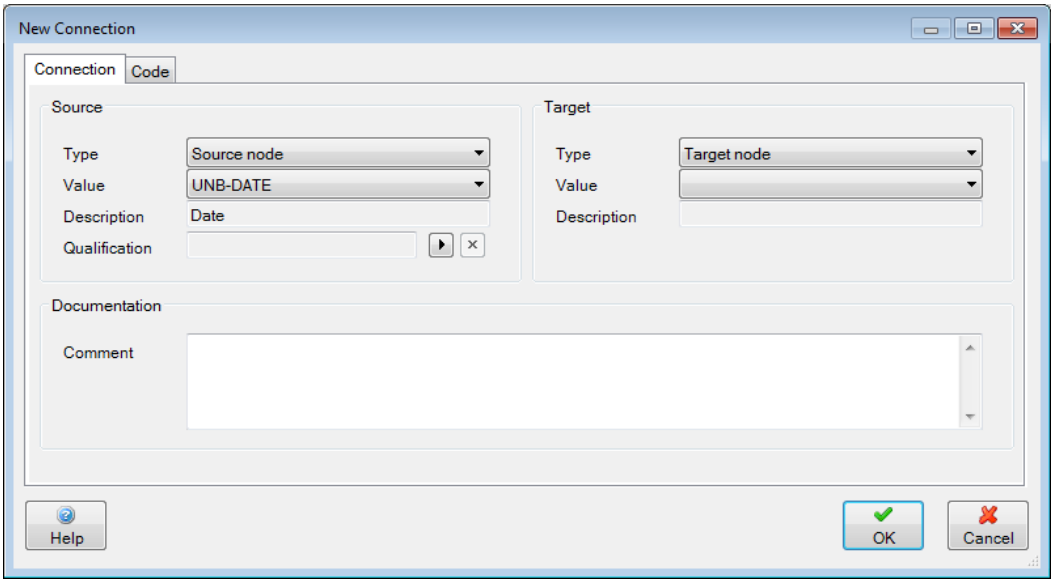


It indicates the source type and source value, and target type and target value.

To add a new connection, select a source or target node at element or field level (i.e. one that cannot be expanded any further) and click the **Add** button.

To edit an existing connection click the **Edit** button. If the **Edit** button is not enabled, then this node is currently not mapped.

When adding or editing a connection the map connection properties dialog will be shown:



The meanings of the source and target types and values are set out in the following sections. The documentation section allows you to enter a comment relating to the map. If you select the appropriate options in the map configuration dialog, this comment will appear in the map script and/or get logged during map execution (when you create a debug version of the map assembly). The “qualification” field against the source allows you to specify that the map represented by the connection will not be performed unless a qualifier has a particular value. Click on the button with an arrow to display a dialog for editing the qualifier properties, or on the button with a cross to remove the qualifier properties. For more information see the section “Qualifier properties”.

5.11.6.1 Source type

The value of Source type can be any of the following:

- Source node
- Trigger node
- Constants (Const)
- Counter
- Fromvar
- Source
- System variables (Sys)
- Literal Values (Value)
- Built-in functions (Function)
- Code variables (Setval)
- Null value
- Parameter

Each of these is described in a section of its own.

5.11.6.2 Source value

Depending on the Source type you select, the Source value field will offer different options.

If you have selected “Source node” as the Source type in the field above, then you must select the source value from the dropdown list in this field, which will contain all the data elements from the source file. The source value will normally be the same as the Cross-reference name from the Properties page tab, if you are performing a simple one-to-one direct mapping, or the path of the source node if the source syntax is XML.

N.B. When using the dropdown list, take care that you select the appropriate instance of the source value. Most EDIFACT EDI messages have several instances of certain common segments.

If you have selected “Trigger node” as the Source type in the field above, then you must select which source node is to be the trigger node, using the dropdown list in this field. The options offered will be EDI segments, HSE records or XML elements from the source file. Please see the section entitled “Trigger node” for further information on triggers.

If you have selected “Const” as the Source type in the field above, you will be offered a list of all the global Constants you have defined.

If you have selected “Counter” as the Source type in the field above, you will be offered a list of all the global Counters you have defined.

If you have selected “Fromvar” as the Source type in the field above, you will be offered a list of all the global Variables you have defined.

If you have selected “Source” as the Source type in the field above, you must type in a path expression for the required source node or simply #VALUE to obtain the value of the current node.

If you have selected “Sys” as the Source type in the field above, you must type in a system variable. For a list of examples, please refer to the section entitled “System variables (Sys)”.

If you have selected “Value” as the Source type in the field above, you must type in a literal value.

If you have selected “Setval” as the Source type in the field above, you must type in a code snippet variable.

If you have selected “Parameter” as the Source type in the field above, you must type in a valid external parameter name.

5.11.6.3 Target type

The value of Target type can be any of the following:

Target node

Readvar

Triggered node (only if Trigger node is selected in Source value)

N.B. If you selected Trigger node as the source value, the target type will automatically be set to Triggered node and cannot be changed unless you choose a different type of source value.

5.11.6.4 Target value

Depending on the Target type you select, the Target value field will offer different options.

If you selected Target node as the target type, you must select the appropriate target node from the dropdown list in the Target value field. The target value you

require will normally be the same as the Field reference value from the Properties page tab for your target, or the node path if the target syntax is XML.

If you select a Target type of Readvar, you must specify the name of a new variable string in the Target value field, so that it can be addressed by code snippets. Readvar is used where the source node you have selected is not mapped directly to the target.

If you selected Trigger node as the target type, then you must select which target node is to be the triggered node, using the dropdown list in this field. The options offered will be EDI segments, HSE records or XML elements from the target file.

5.11.6.5 Default

This property will not be made available until you have saved the source and target property details.

The Default parameter is used to specify a default value that will be mapped if the source node value is empty or null. Simply type in the default value to be mapped.

Please note that, if the segment or record containing the source value is missing from the source file, the default value will not be mapped.

5.11.6.6 Lookup table and direction

These properties will not be made available until you have saved the source and target property details.

You may provide the name of a Lookup table, which can be used to translate code values from one value in the source file to a different value in the target file.

If a Lookup table is specified, you must also select the appropriate direction for the lookup (Left-to-right or Right-to-left).

5.11.6.7 Map from local var

This property will not be made available until you have saved the source and target property details.

Use the “Map from local var” parameter to specify that the target value is to be mapped from a local variable, rather than from a source element.

Its value should be a meaningful code variable name that you have already defined on the Code variables page tab. If the variable is not a string, then the value mapped is the result of calling its `ToString()` method.

5.11.6.8 Re-use service segment data

In addition, if the selected node is part of an EDI service segment, the Re-use service segment data property (not shown in the example) allows you to specify whether the node should be reusable (Yes) or should only be used once (No). However, this property will not appear if you are mapping a fixed value.

5.11.6.9 Map page tab buttons

The following buttons are included on the Map page tab:

Previous

Click the **Previous** button to go to the previous mapped node higher up the hierarchy.

Next

Click the **Next** button to go to the next mapped node lower down the hierarchy.

New

Click the **New** button to show the map connection properties dialog for creating a new connection.

For a straightforward one-to-one source-to-target link you can create the link using the drag-and-drop facility without using the **New** button. Otherwise, simply configure the Map parameters as described above.

Delete

Highlight a mapped node and click the **Delete** button to delete an existing link.

Edit

Highlight a mapped node or connection and click the **Edit** button to show the map connection properties dialog for editing the connection.

Save

Click the **Save** button to save any changes you have made.

Cancel

Click the **Cancel** button to cancel any changes you have made.

5.11.7 Conditions

This page tab is applicable to any mapped node.

This feature allows you to:

- specify a condition to be tested, whose result determines whether mapping will occur
- specify a field or variable that is to trigger mapping only when its value changes

Please note that the condition may be tested on an element or field other than the actual element or field to be mapped. For example, you could test the segment qualifier held in one element to determine whether to map the associated segment data. The data on which the condition is tested can be anywhere in the file, provided you can specify the correct path expression for it.

The format of the IF condition must consist of:

- a relative or absolute path expression on the left-hand side (e.g. LVAL=”#VALUE”)
- a condition to be tested for (e.g. CND=”EQ”)
- a literal value to be test for the right-hand side of the expression (e.g. RVAL=”AAK”)

Several conditions can be tested together by means of the AND and OR operators.

5.11.7.1 Map Condition Examples

```
(LVAL="#VALUE",CND="EQ",RVAL="AAK")
```

Means "if the value [of the context node] is equal to AAK"

```
(LVAL="#VALUE",CND="EQ",RVAL="BY, MI, ST")
```

Means "if the value [of the context node] is equal to BY or is equal to MI or is equal to ST"

```
(LVAL="#VALUE",CND="NE",RVAL="2")
```

Means "if the value [of the context node] is not equal to 2"

```
(LVAL="#VALUE",CND="NE",RVAL="2,3,4")
```

Means "if the value [of the context node] is not equal to 2 and is not equal to 3 and is not equal to 4"

```
((LVAL="#NAME",CND="EQ",RVAL="ABC") AND
((LVAL="CHILD[#REF=ABC00010]/#VALUE",CND="EQ",RVAL="10") OR
(LVAL="CHILD[#REF=ABC00020]/#VALUE",CND="EQ",RVAL="20")))
```

Means "if the name [of the context node] is ABC and its child with reference ABC00010 has the value 10 or its child with reference ABC00020 has the value 20".

The full list of conditions that can be tested for is as follows:

- EQ – Equals
- NE – Is not equal to
- GT – Is greater than
- GE – Is greater than or equal to
- LT – Is less than
- LE – Is less than or equal to

5.11.7.2 Change field

Click the **New change field** button to specify a field that you want to trigger mapping only when its value changes in the source file. Use the value field to specify the name of the field or to provide its XML path expression. The default value is #VALUE i.e. the value of the context node.

You may set up any number of change fields for a node. Mapping will be triggered if the value of any one of them changes.

The effect of using this parameter is roughly equivalent to using the CHG record in Xlate.

5.11.7.3 Change variable

Click the **New change variable** button to specify the name of a global variable that is to trigger mapping when its value changes. Use the value field to specify the name of the global variable. There is no default value.

You may set up any number of change variables for a node. Mapping will be triggered if the value of any one of them changes.

In order to use this option, you need to set up a global variable and, using a code snippet on a source node that is processed before this one, store a value from that source node in the global variable each time that source node is processed.

5.11.7.4 Delete button

Highlight an existing change field or change variable and click the **Delete** button to delete a change field or change variable.

5.11.8 Code snippets

This page tab is applicable to any mapped node.

This feature allows you to add snippets of code which will be processed either immediately before or immediately after mapping the value with which they are associated. It also allows you to view code that has been generated by Xe to carry out the actions of built-in functions.

The available options are:

PRE (executed before mapping a value)

POST (executed after mapping a value)

GEN (generated code – read only)

You could use the PRE option, for example, to reformat a date from the source file before mapping it to the target.

POST could be used to store a source value in a global variable after the variable has been used for mapping but before it is used the next time.

Simply type the code snippet into the space provided.

GEN code is recreated by Xe each time a built-in function is edited, and each time a connection to or from a function is changed, so it cannot be edited directly. You can, however, view the code that Xe generates and possibly use it as the basis for your own code snippets.

For a list of all the code variable types that can be used in code snippets, please refer to the section entitled “Code variable types”.

5.11.9 Code variables

This page tab is applicable to any mapped node.

This feature allows you to add code variables to the map file. These variables can then be used during mapping to manipulate the data values.

Click the **New** button to add a new code variable to be associated with the selected node. It will be created as new_setvar1, new_setvar2 etc.

Give the variable a meaningful name (optional) and edit its type e.g. string, int

In the Path field you must provide the Path value. The default value is #VALUE, which equates to the value of the current node. For further details of how to construct a path value, please refer to the section entitled “Path values and expressions”.

Highlight an existing variable and click the **Delete** button to delete a variable.

For a list of all the code variable types that can be used, please refer to the section entitled “Code variable types”.

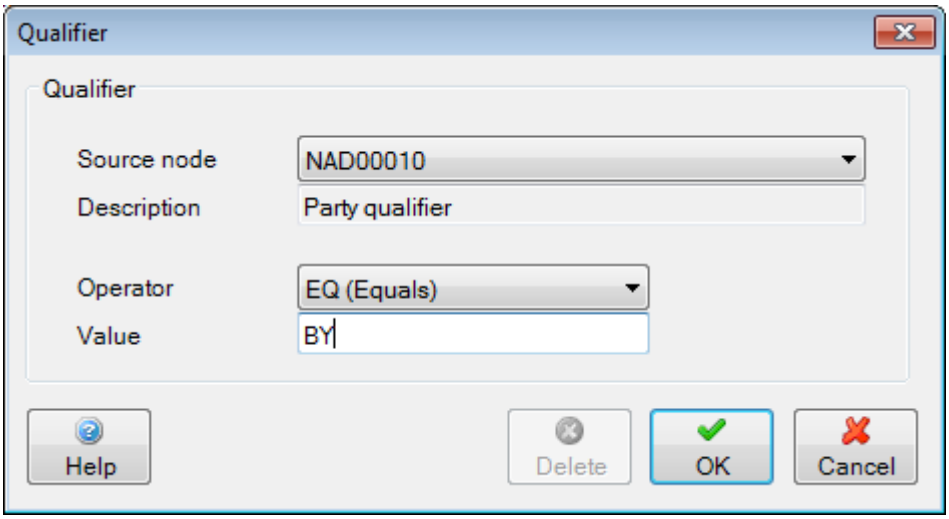
5.11.10 Qualifier properties

In EDI messages it is common for a segment to be used for different (but related) purposes indicated by qualifier values that appear in the segment. For instance, the common EDIFACT NAD (Name and Address) segment uses a qualifier in the first element to indicate what the name and address represents.

So NAD+BY… represents the name and address of a ‘buyer’, and NAD+SE… represents the name and address of a ‘seller’.

Xe allow you to specify that a map should be performed only if a field contains (or does not contain) a particular qualifier value. This mechanism is primarily designed to support mapping qualified segments, but it can equally be applied to other situations in which the map to be performed depends on the value of a data item.

A qualifier condition can be added to any map connection whose source is a tree node. You can add qualifier properties when adding or editing the connection, or by right clicking on the connection and choosing the “Qualifier properties” menu item. The following dialog will be displayed.



The fields shown have the following meanings:

- **Source node** – shows the cross-reference of the source node that contains the qualifier value (or the path if the source is an XML document). The drop-down arrow can be used to select a different qualifier field. The **Description** field displays a description of the qualifier field selected if there is one in the document definition.
- **Operator** – determine whether the ‘equal’ (EQ) or ‘not-equal’ (NE) operator is used when testing the qualifier value.
- **Value** – specified the value to check for in the given source node. The map represented by the connection is not performed unless the value of the source node matches the value specified here (if the operator is EQ – or does not match it if the operator is NE).

The **Delete** button is enabled when editing existing qualifier details and is used to remove those details from the connection.

5.12 Path values and expressions

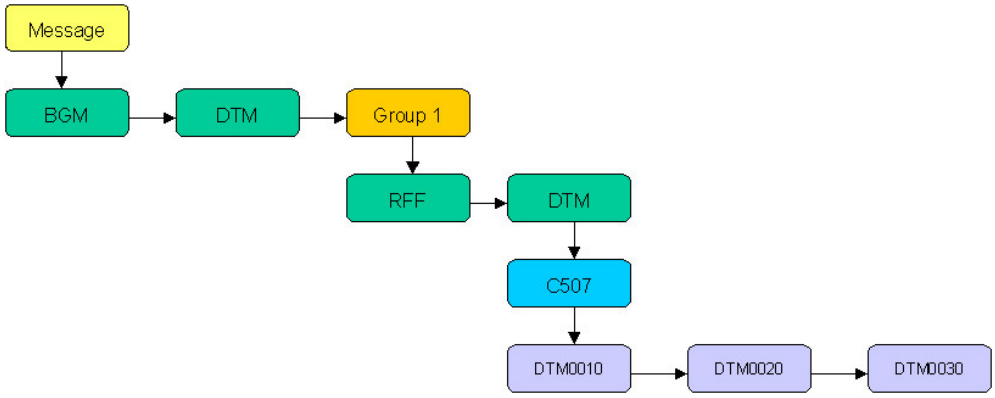
It is important to understand the way in which data is held in the Xe DOM, so that you can take advantage of the facility that allows you to code path values and expressions in order to access specific data in the DOM using code snippets.

The data model (DOM) is an XML structure and the objects and data in it must be addressed using standard XML addressing techniques.

Data is held in a specific format, depending on whether it is EDI data, in-house data or XML data. A schematic representation of each type can be seen in the following sections.

5.12.1 EDI Data

The hierarchical representation of EDI data in the DOM is illustrated in the following diagram, where each entity is a data node:



The key features are as follows:

- Messages are always children of message groups or interchanges.
- Entities at the same level (e.g. segments at the same group level, or elements within a segment) appear as data node siblings.
- Non-hierarchical repeats of entities appear as node siblings.
- Entities at lower levels appear as node children of the parent entity.
- The actual element data (e.g. DTM0010, DTM0020 and DTM0030 in the example) appears as a value in the data node. Each data element or sub-element is represented by a value in a node.

Note that in a traditional EDI hierarchical diagram, the DTM segment would be shown as the “child” of the RFF segment, whereas in the DOM hierarchy these segments are both siblings at the same hierarchical level.

Similarly, the three sub-elements of the composite element are siblings as well as children of the composite.

5.12.1.1 Examples

To access the value of the very first instance of element DTM0030 in our example, you could use the following absolute address:

```
//CHILD[#NAME=GROUP1](1)/CHILD[#NAME=DTM](1)/CHILD[#NAME=C507](1)/CHILD
[#REF=DTM0030]/#VALUE
```

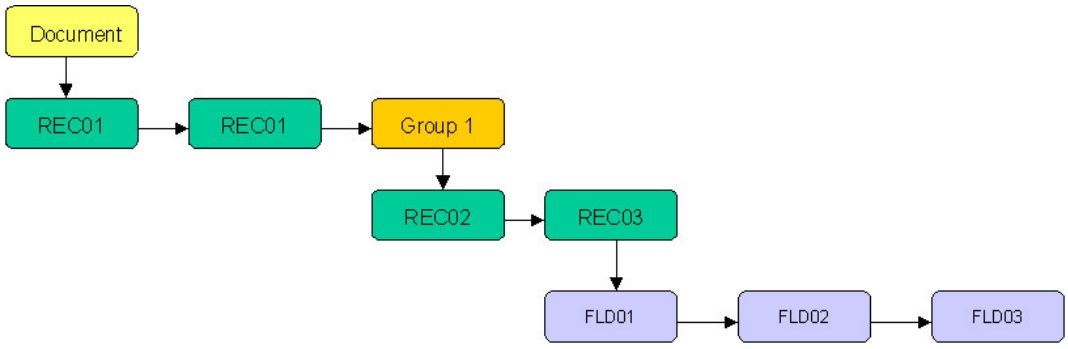
Or equally:

```
//CHILD[#NAME=GROUP1](1)/CHILD[#NAME=DTM](1)/DESCENDANT[#REF=DTM0030]/#
VALUE
```

As you can see in the second address example, you can select a specific descendant instead of navigating via the composite element. However, this is only recommended as a way of addressing a data node once you have navigated as far as the required segment, since it is otherwise not an efficient method.

5.12.2 In-house data

The representation of in-house data in the DOM is illustrated in the following diagram:



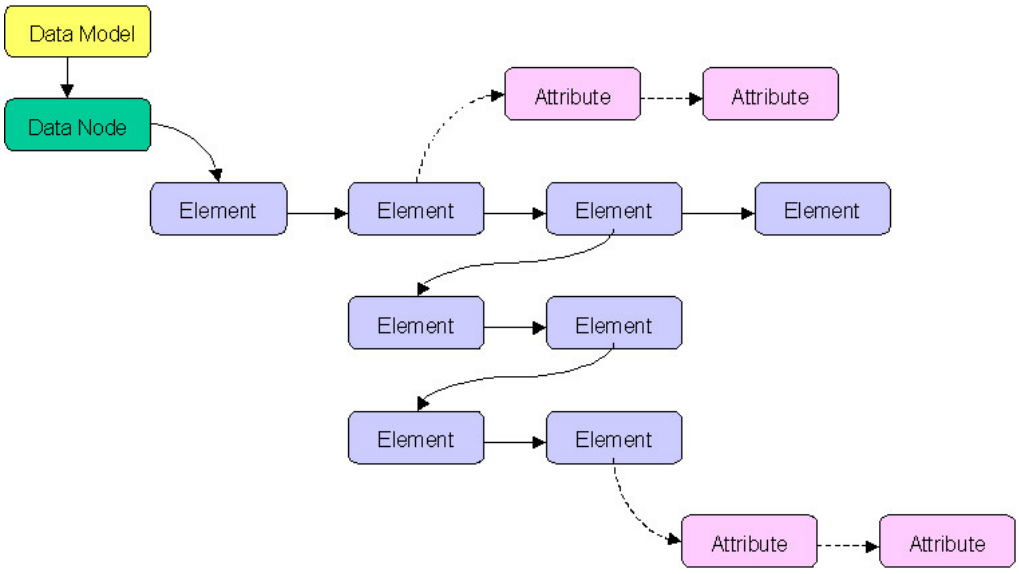
The key features are as follows:

- Each in-house document is represented by a document node. An in-house file either contains a single document, or `MHD` or `FID` processing in the index is used to define records that trigger new documents.
- All HSE entities within a message are XML node objects.
- Entities at the same level appear as data node siblings.
- Non-hierarchical repeats of entities appear as node siblings.
- Entities at lower levels appear as node children of the parent entity.

The actual field data appears as a value in the data node. Each field is represented by a value in a node.

5.12.3 XML data

The representation of XML data in the DOM is illustrated in the following diagram:



The key features are as follows:

- There is a single root data node as a child of the data model object; an XML document cannot have more than one root node.
- XML elements are node objects.
- XML attributes are attribute objects.

5.12.4 Using path expressions

Path expressions are made up from these elements:

- Context
- Path
- Identifier (optional)

If an identifier (such as `#VALUE`) is specified, the path expression evaluates to a single value.

If there is no identifier specified, the expression evaluates to a sequence of nodes (which may be empty, or contain only one node).

Context

The opening characters of the expression will define the context:

- To start with the current context node, for relative addressing, use `'./'`.
- To start with the root node, for absolute addressing, use `'//'`.

The default is relative addressing, as a result of which the opening may be dropped and the path expression started with a valid path.

Path

The path consists of zero or more path elements separated by the `'/'` character. Each path element is made up of:

- A node 'axis' (which is one of a subset of axes taken from XPath).
- Zero or more node tests.
- An instance number.

The axes that may be used are:

- `CHILD` - immediate children of the context node.
- `PARENT` - immediate parent of the context node.
- `DESCENDANT` - all nodes below the context node on the same branch.
- `ANCESTOR` - all nodes up to and including the root node that is a direct or indirect parent of the context node.
- `SEG` – the owning segment of an element or composite where the document is EDI.
- `REC` – the parent record of a field where the document is HSE.
- `ATTRIBUTE` – all attribute nodes of an element where the document is XML.
- `ATTRIBUTE-OR-SELF` – all attribute nodes and the text value of an element where the document is XML.
- `FOLLOWING-SIBLING` - all nodes to the right of the context node at the same level.
- `PRECEDING-SIBLING` - all nodes to the left of the context node at the same level.
- `SIBLING` - all nodes to the left or right of the context node at the same level.
- `SELF` – the context node.

A node test consists of a single comparison operation on one of a number of pre-defined node characteristics. The pre-defined identifiers are:

- `#NAME` - the node name, such as `'DTM'`
- `#REF` - the node reference, such as `'DTM00010'`
- `#VALUE` - the node value

A node test is expressed in the form `[#VALUE='123']`. An instance number can be specified in the form `'(x)'` where `'x'` is any integer, and may follow the node axis, or any one or more of the node tests. An instance test of `'(~)'` indicates that the last instance of the specified node is required.

Identifier

If the path expression concludes with one of the identifiers listed above, then the expression resolves to the value of that identifier on the node indicated by the rest of the path expression (or the context node if the path is empty).

Examples

There follows a number of examples of how to use path expressions

```
./#VALUE or #VALUE
```

The value of the context node.

```
./FOLLOWING-SIBLING
```

All following siblings of the context node.

```
//DESCENDANT[#NAME=CSG]
```

All descendant nodes of the root node that are named CSG.

```
./CHILD(3)/#VALUE
```

The value of the third child of the context node

```
./PARENT/FOLLOWING-SIBLING(2)/CHILD[#REF=ABC00010]
```

The node with the unique reference 'ABC00010' that is the child of the second following sibling of the parent of the context node.

```
//CHILD[#NAME=GROUP1](1)/CHILD[#NAME=ABC](1)/CHILD[#REF=ABC00010]/#VALUE
```

The node with the unique reference 'ABC00010' that is the child of the first occurrence of the child named ‘ABC’ that is the child of the first occurrence of the child of the root node that is named ‘GROUP1’.

Short syntax

A path expression short syntax is available with terse equivalents for the axes and properties described above. The shortened axis names are:

Axis	Short syntax
CHILD	**$C**
PARENT	**$P**
DESCENDANT	**$D**
ANCESTOR	**$A**
SEG	**$^**

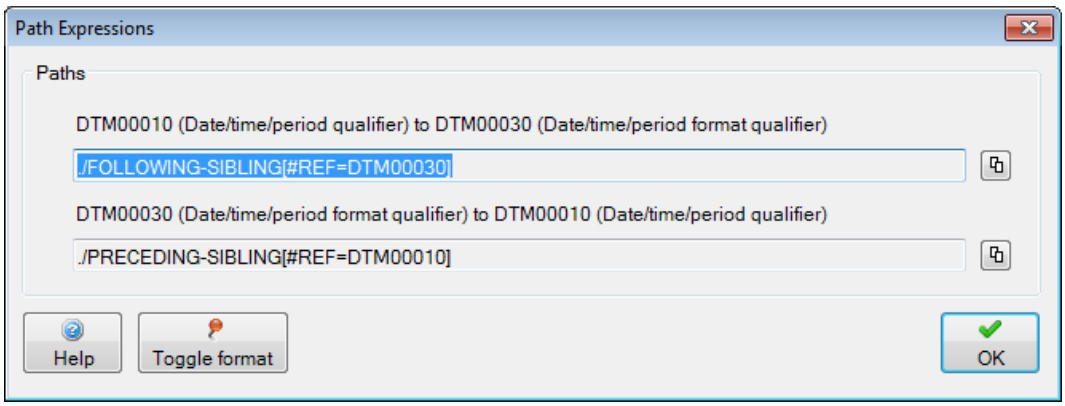
REC	**$^**
ATTRIBUTE	**$@**
ATTRIBUTE-OR-SELF	**$***
FOLLOWING-SIBLING	**$>**
PRECEDING-SIBLING	**$<**
SIBLING	**$S**
SELF	**$!**

The shortened property names are:

Axis	Short syntax
#NAME	**#N**
#REF	**#R**
#VALUE	**#V**

5.12.5 Creating path expressions in the mapper

The mapper has a facility for helping you to generate path expressions. On the left hand tree view, highlight any node. Then hold down the **Ctrl** key and drag with the mouse until the cursor is over any other tree node. When you release the mouse, this dialog is shown:



It shows two paths - the path from the first node selected to the node over which the mouse was released, and the path for the reverse navigation. Use the buttons following the paths to copy the text to the clipboard. Use the **Toggle format** button to switch between the standard (verbose) path expression format and the abbreviated syntax.

Note that the expressions created do not contain instance tests. If you want to test for particular instances of certain nodes then you must insert the node tests manually when using the expressions.

5.13 Mapping from different sources

Most of the time you will probably be using straightforward mapping from a source node to a target node. However, in some cases you may want to map to the target node in a different way.

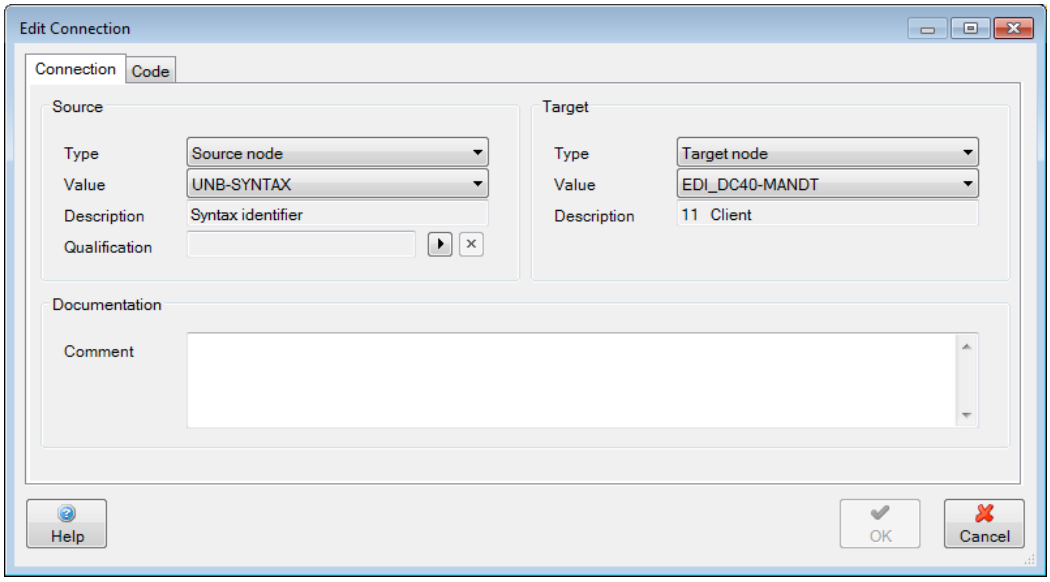
The following source type values can be used for this:

- Const – specify a named constant.
- Counter – specify a named counter.
- Fromvar – specify a local or global variable.
- Setval – specify a code snippet variable.
- Source – specify a source node to map from, using a path expression.
- Sys – specify a system variable.
- Trigger node – specify a source node that will trigger the creation of a target node
- Value – specify a literal value.
- Function – specify a built-in function.
- Null value – no parameter required.
- Parameter – specify an external parameter name.

In addition, the value mapped can be manipulated using:

- Lookup table and Lookup direction – specify a lookup table and direction to convert the value.
- Readvar – specify a node in the source file definition, whose value is to be read into a variable, so that it can be addressed by code snippets. It is used where the source node in question is not mapped directly.
- Static variables – specify a static variable that can be used across more than one map.

Existing map connections can be edited, or new ones created, using the connection properties dialog:



To edit an existing connection:

- Click on the connection with the right mouse button and select Edit Connection from the menu.
- Double-clicking the connection.
- Clicking the Edit button on the Map Properties tab.

To add a new connection:

- Click on a source or target tree node with the right mouse button and select Add Connection from the menu.
- Clicking the New button on the Map Properties tab.

The “qualification” field against the source allows you to specify that the map represented by the connection will not be performed unless a qualifier has a particular value. Click on the button with an arrow to display a dialog for editing the qualifier properties, or on the button with a cross to remove the qualifier properties. For more information see the section “Qualifier properties”.

5.13.1 Source node

This mapping source is the most commonly used and has been included in this section for completeness.

Simply select the required Source Value from the dropdown list. The list contains all the data fields/elements/attributes that are defined in the source file definition.

5.13.2 Trigger node

When defining or using a map in which data from a low level in the source file is used at a higher level in the target file, you may define a trigger record that will in effect manage the hierarchical structure of the target file.

To define a trigger, select a segment node or record node (not element or field) from the source file using the right mouse button and drag it across to the target node. This will create a purple dashed link line from the source segment or record to the target record or segment.

You can also achieve the same thing from the Map page tab, by setting the source type of any node (at segment or element level for an EDI file, or at record or field level for a non-EDI file) to be the Trigger node and the target type to be the Triggered node. However, the trigger will always be set at the segment or record level.

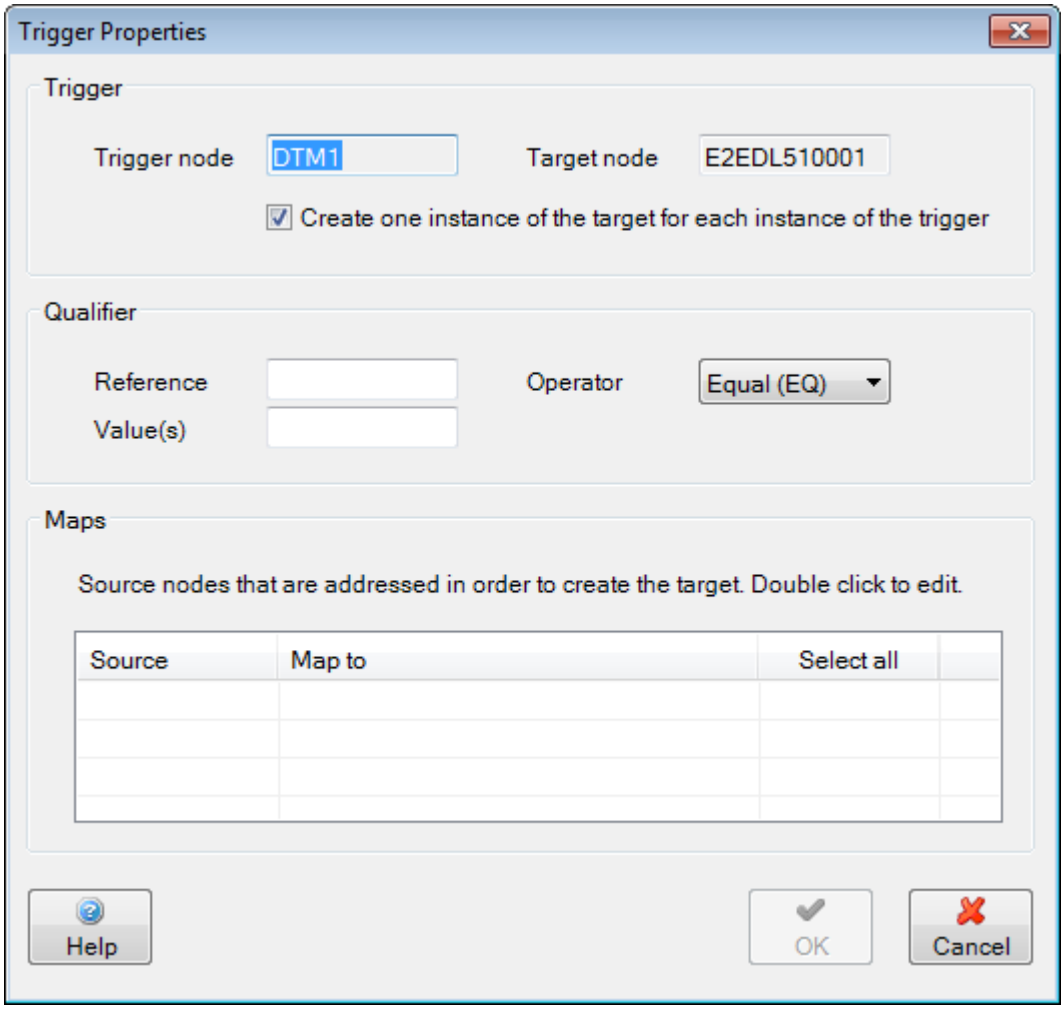
The Map page now shows the source value and target value of the trigger. The source type will be trigger node, while the target type will be triggered node.

Each time a node with the given source value is read from the source file, this will trigger the creation of a node with the given target value in the target file.

Remember – triggers act at EDI segment (or HSE record or XML element) level, not at EDI element (or HSE field or XML attribute) level.

Once a trigger connection has been created, it is possible to set advanced properties against it. These provide more fine control over the map generation process for cases where Xe's default mapping rules do not produce the results that you need. If advanced properties are set against a trigger connection then the link appears as a dashed orange line.

To display the advanced properties dialog, click the right mouse button on the dashed trigger connection and choose Advanced Properties on the pop-up menu:



There are three sections to the dialog, described below:

Trigger

This section is used to indicate the trigger node and triggered (target) node. The display names used are the unique names used by Xe to reference segments or

records. So, for instance, a segment named LIN may appear more than once in an EDI definition, in which case Xe will assign unique names to the instances - LIN1, LIN2 etc.

In addition a ‘repeats’ checkbox is displayed. By default, one instance of the target node is created for each instance of the trigger node. The check box can be unchecked to indicate that the target node is be created once only, for any number of instances of the trigger node.

Qualifier

This section can be used to define a qualifier test. In the Reference field enter the cross-reference of a source node (element or field) that contains a qualifier value. In the Value field enter a value to be tested (or multiple values separated by commas). Use the operator combo box to choose the operator for comparison (equal or not equal).

When the map is run, Xe fetches the value in the element or field given by the specified Reference and compares it with the Value field. It the result of the comparison is false then target node is not mapped.

Maps

When Xe generates a map that requires data to be taken from more than one source segment or record to create a single target segment or record, treats the trigger node as the primary source of data and creates loops that address the other sources of data. By default, only one instance of those other sources of data will be addressed. There are circumstances in which you may want to address them all; for instance, where the source of data is a qualified segment and you are performing a condition test based on the qualifier value. This section shows each source node (record or segment) from which data is mapped to the target – excluding the trigger node – the target elements or fields that each supplies, and a flag indicating whether Xe should select all or just one of the instances of the source node. Double click on an item in the list to toggle between ‘Y’ and ‘N’.

5.13.3 Constants (Const)

There may be some instances where a specific value is required in the target node but it is not present in the source file e.g. your own or your trading partner’s EDI code. In this case, you could define the EDI code as a global constant and use it in the mapping.

Click on the central mapping area of the GUI and select the Constants page tab.

Click the **New** button to create a new constant. Give it a meaningful name and provide its value in the Value column.

5.13.4 Counter

A counter must be set up as a global counter or an index counter, so that it is available throughout the mapping process. It can then typically be used as a segment counter. You will need to set up a separate counter for each different usage.

Click on the central mapping area of the GUI and select the Counters page tab.

Click the **New** button. The name “new_counter1 will appear in the Name field. You may replace this with your own name for the counter.

The initial value will be set to 0 and the increment value will be set to 1. You should change these if necessary.

If you do not provide a value in the mask field, the counter will be a basic counter without formatting or fixed length.

For example, a counter which has an initial value of 0 and an increment value of 10 would create counters as follows: 0, 10, 20, 30 ……

A counter which has an initial value of 1 and an increment value of 1 would create counters as follows: 1, 2, 3, 4 ……

If you require formatting or a fixed length, type the required placeholder details in the Mask field.

See also “**Error! Reference source not found.**”.

5.13.4.1 Mask field

The mask should include a placeholder for the counter, with a reference to the number of digits it has. For example, a counter with 5 digits should be specified as %CNT:5% (please note the colon character “:” after the CNT characters.

Before and after this placeholder you can specify other constants which make up the counter, such as a string of characters or digits.

A counter consisting of 5 digits prefixed by the letter A and followed by the letter Z, (e.g. A00001Z) would require a mask of A%CNT:5%Z

5.13.5 Fromvar

The `FROMVAR` parameter is used to specify a variable name that is the source of the data value to be mapped. The referenced variable will be one that has been declared in the `GLOBAL` statement's `DEFN` parameter. If the variable is not a string then the value mapped is the result of calling its `ToString()` method.

5.13.6 Source

This parameter allows you to specify a source node using a path expression. Normally you would achieve this type of mapping using drag-and-drop or by using the Source Node parameter. However, those two mapping methods are only useful if the mapping is to be performed every time the source node is encountered.

Sometimes you might need to map a very specific item of data which only occurs once in the source file. In this case you would need to map to a target node from a specific address. The path expression you provide may be a relative path or an absolute path. For more information about path expressions, please refer to the sections entitled “Path values and expressions” and “Using path expressions”.

The example below shows an absolute path expression that could be used to obtain a specific item of information.

```
//CHILD[#NAME=GROUP1](1)/CHILD[#NAME=ABC](1)/CHILD[#REF=ABC00010]/#VALUE
```

The node with the unique reference 'ABC00010' that is the child of the first occurrence of the child named ‘ABC’ that is the child of the first occurrence of the child of the root node that is named ‘GROUP1’.

Please note that you should only use the Source parameter in this way if you are absolutely certain of the address of the data item you want.

5.13.7 System variables (Sys)

You may use system variables to map to a target.

The following system variables are supported:

- *DATE: A date in the form YYMMDD
- *DATE001: A date in the form CCYYMMDD
- *TIME: A time in the form HHMM
- *DT(fmt) where 'fmt' can have one of these values:
    - HHMM
    - HHMMSS
    - YYMMDD
    - CCYYMMDD
    - YYMMDDHHMM
    - YYMMDDHHMMSS
    - CCYYMMDDHHMM
    - CCYYMMDDHHMMSS
- *CDT(fmt) where 'fmt' can be any standard or bespoke format string usable with the .NET `System.DateTime.ToString()` function. For example, *CDT("F") produces output in the form `"Thursday, August 17, 2000 16:32:32"`.
- *OFN: the original (source) file name, without path or extension
- *OFN_W_E: the original (source) file name, without path but with extension
- *OFN_E: the original (source) file extension
- *OFP: the original (source) file path (directory)
- *OFP_F: the full original (source) file path (directory and file name with extension)

5.13.8 Literal Values (Value)

There may be some instances where a specific fixed value is required in the target node but it is not present in the source file e.g. a segment qualifier in an EDIFACT message. In this case, you would want to map a literal value.

To map a literal value, highlight the target node, select the map page tab and click the **New** button.

The Source type Value field and Target type Value field will show (null).

Click on the Source type Value field to see the dropdown list of available values, and select “Value”.

Click on the Source value Value field and type in the literal value to be used for this mapping.

Click on the Target type Value field to see the dropdown list of available values, and select “Target node”.

Click on the Target value Value field and select the appropriate entry from the dropdown list i.e. the cross-reference of this field as specified in the target file definition.

5.13.9 Built-in functions (Function)

To map a built-in function you first need to create an instance of one using the Functions tab page. The use of built-in functions is described fully in the section “Built-in functions”.

A built-in function can be the source or target of a map connection. If the selection in the Source Type field is “Function” then the Source Value field will show a list of functions that have already been created and can be mapped to. If the selection in the Target Type field is “Function” then the Target Value field will show a list of functions that can be mapped from.

5.13.10 Null value

If you choose to map a literal value, then you must specify at least one character to map. However, if you do want to map an empty value, then the map source type of Null Value should be chosen. This will be most useful when creating XML, as is can be used to force out an empty instance of an element.

5.13.11 Parameters

Parameters provide a way for you to include external data in a map, separate from the source document. This data may change on a per-execution basis. Parameter names are specified in the form %NAME%. To map a parameter value, select the source type of Parameter from the Source Type drop-down list and then type the required parameter name in the Source Value field. When Xe is invoked to perform a map, values must be provided for the parameters that are used. When running from Xe mapper GUI, parameter values may be specified in the configuration dialog. When running the command-line executable XeCmd, the parameter values are included in the config file used to invoke it. When running as a job under ODEX Enterprise, parameter values may be configured as job parameters in the Administrator.

5.13.12 Code variables (Setval)

You may sometimes want to manipulate data before inserting it in the target file. This is where code variables (the Setval source type) play their part.

Let’s take the simple example of a source file which contains a date in the format CCYYMMDD which you want to place in the target file in the format YYMMDD.

There are 3 parts involved in achieving the required outcome:

- specify the code variable in which to place the data from the source node prior to manipulation
- specify the local variable in which to place the data after manipulation (this is the value that will be mapped). Alternatively, you could use a global variable instead.
- provide the code snippet that will perform the actual manipulation.

Select the source node from which the data is to be taken and click on the Code variables page tab.

Click the **New** button and provide the name of the variable (e.g. strDate), its type (e.g. string) and the path. In this particular example, we are using the actual source data, so the path will be ./#VALUE to indicate the value of the current node.

Select the Map page tab. In the “Map from local var” field, type in the name of a local variable (e.g. strShortDate). This is the value that will be mapped to the target node after manipulation.

Select the Code Snippets page tab and use the dropdown arrow to select PRE (since in this example we are manipulating the data before it is mapped).

The code snippet for our example would be

```
strShortDate = strDate.Substring (2, 6);
```

The effect of this is to take 6 digits from the source date (starting at the 3rd digit – the Substring works with a zero base) and put them into strShortDate, which is then mapped to the target.

If the variable to be mapped is not a string then the value mapped is the result of calling its `ToString()` method.

For a full list of code variable types that may be used in Xe, please refer to the section entitled “Code variable types” in Appendix B.

5.13.13 Lookup table and direction

Sometimes you may have the situation where a code value from the source file needs to be translated into a different but equivalent code value for insertion in the target file. In such cases you should use the lookup table property.

You need to provide the name of a Lookup table, which can be used to translate code values from one value in the source file to a different value in the target file.

You also need to specify the lookup direction.

For details of how to create a lookup table, please refer to the Xlate on-line help.

5.13.14 Readvar

The `READVAR` statement is used when you want to designate a node in the source file definition (a source field or element), whose value is to be read into a variable, so that it can be addressed by code snippets. It is used where the source node in question is not mapped directly.

When using Readvar, you must provide the name of a new string variable in the Target Value field. The value of the designated source node will be assigned to this variable. The variable can then be manipulated using a POST code snippet.

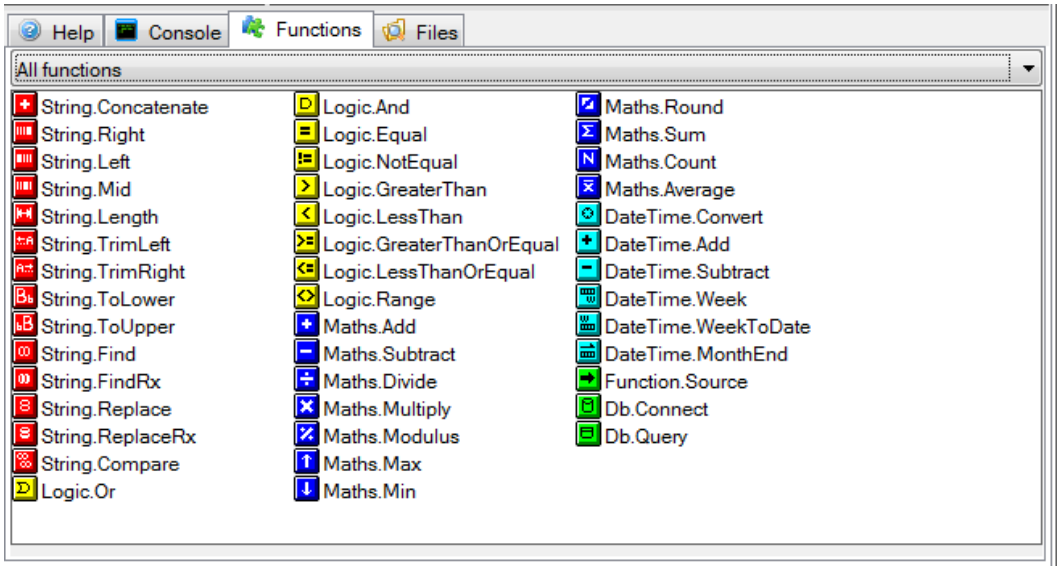
There is also a shortcut for this process. Please refer to the section entitled “Source shortcut”.

5.14 Built-in functions

Xe has a number of built-in functions that are used in the mapper to simplify the performance of commonly used actions, such as string manipulation, condition testing and lookups.

When functions are used in the mapper, they cause Xe to generate code snippets that are inserted into the mapping code. Functions can be used in place of your own code snippets to customise maps.

The functions available for use are shown in the functions tab in the bottom right section of the mapper screen.

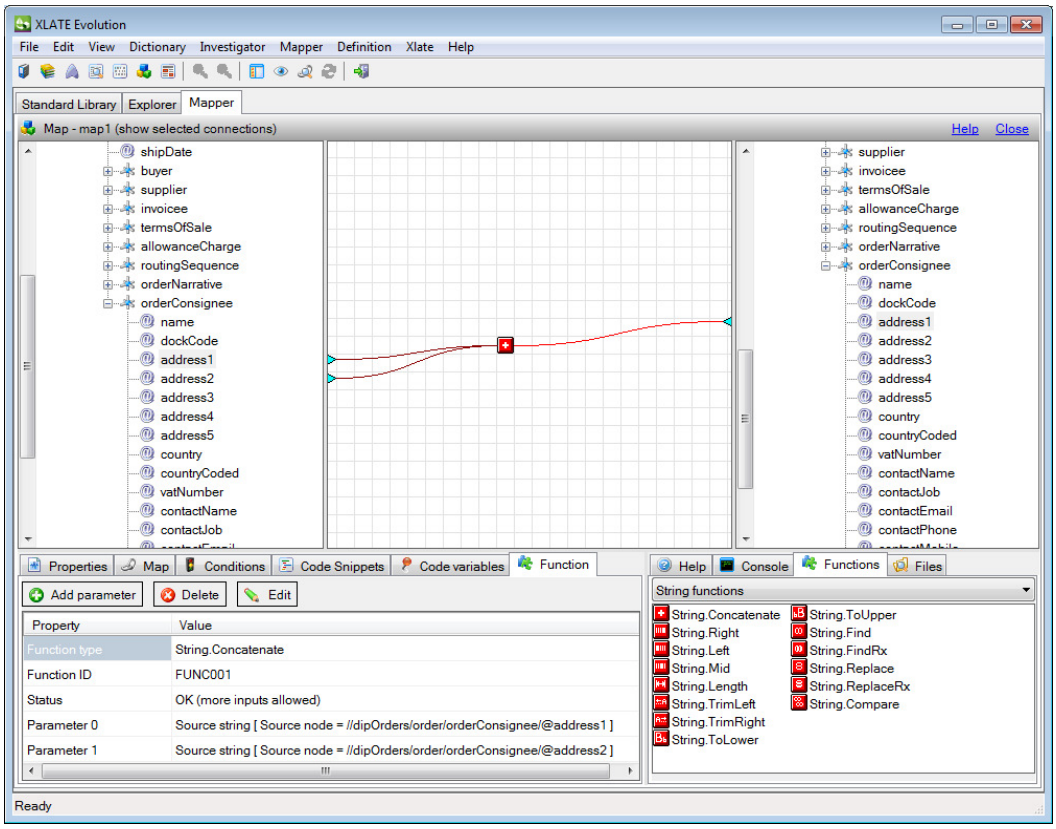


The built-in functions each belong to one of a number of categories. The category being displayed appears in the drop-down list at the top of the functions tab. Use the drop-down to select the category of function to display, or select “All” to show all of the available functions.

The following sections explain how to make use of built-in functions:

5.14.1 Using built-in functions

The simplest way to use a function is to drag its icon from the function tab to the mapping grid in the centre of the screen. Map connections can then be dragged to and from the function to represent the input parameters and the function output. The example below shows a **String.Concatenate** function being used to join two text strings together.



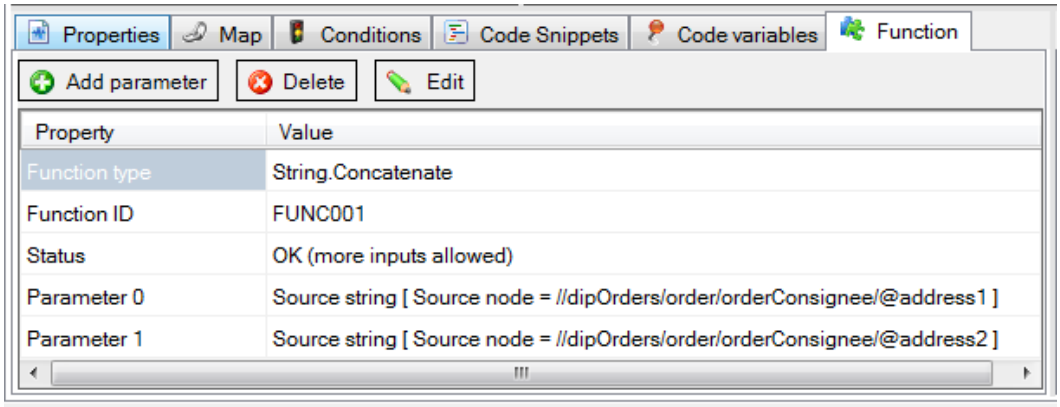
The mapping grid represents a virtual surface that contains a number of cells, each of which may contain a function. The grid extends beyond the visible boundaries of the area between the left and right hand tree views. To move the grid you can click on an empty area of it and move the mouse while holding the left mouse button. The cursor will turn to a hand and the visible part of the grid will shift. When you come to any edge of the grid, the cells will appear shaded and the grid will scroll no further in that direction.

You can place a function anywhere on the grid except on or adjacent to another function and in the shaded edges. A function can be moved around the grid by first highlighting it with a single click of the left mouse button (which will cause it to be displayed with a shaded border) and then dragging it to a new position. If you do not highlight the function first, trying to drag it will cause a new map connection to be dragged from the function instead. If you drag a function outside the visible area of the grid then the grid will be scrolled in the direction you drag.

As well as dragging functions from the list in the bottom right of the mapper, you can also right click on any function there for a menu containing options to create a new instance. You can choose to create a new instance on its own, or to create an instance connected to a highlighted source or target tree node.

5.14.2 Function properties tab

When a function has been added to the mapping grid and selected with the mouse, its properties will be displayed in the function properties tab in the bottom left section of the screen.



The properties displayed here are:

- Function type - a short name indicating the function category and purpose.
- Function ID - a unique named used by Xe to identify functions; these are assigned automatically when functions are created, but can be edited in the function properties dialog if required.
- Status - indicates the current function status based on the required number of input parameters; if the required minimum number of parameters have been defined, the value will either be “OK” or “OK (more inputs allowed)” otherwise it will be “Too few inputs”.
- Parameters - each of the input parameters defined so far will be displayed in sequence; the parameter description is shown first, followed in brackets by the source type and value.

The following buttons are included on the function properties tab:

Add Parameter

This button will be enabled if more input parameters can be defined for the selected function. Click **Add Parameter** to show the function parameter dialog, which is used to define a new parameter. See “Function parameter properties”.

Delete

Click the **Delete** button to delete the selected function and all of the map connections to and from it.

Edit

Click the **Edit** button to view and edit the properties of the selected function in a dialog. See “Function properties”.

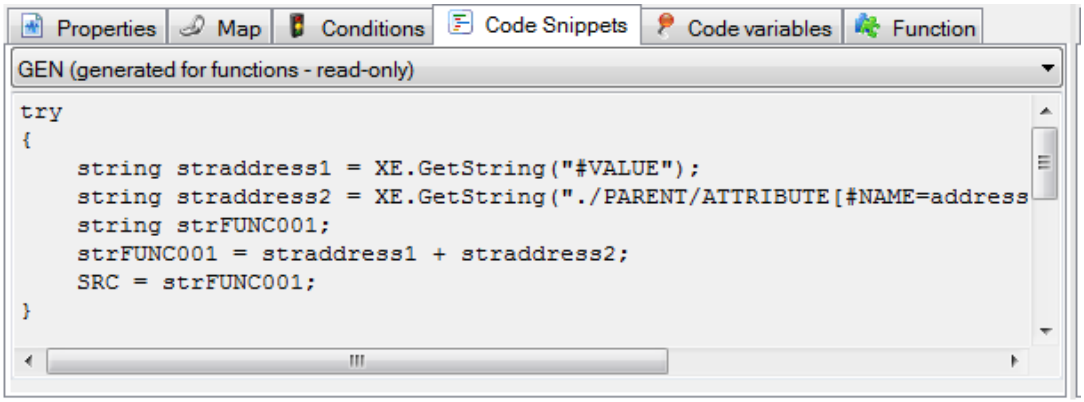
These options are also available from the function context menu. To display the menu, right click on any function.

5.14.3 Generated code snippet

The actions of built-in functions are implemented in Xe by generating code snippets that are executed when mapping to a target node. Code snippets are generated on-the-fly as the map is edited. When the map script is created, the generated code snippets are added to the script. When the mapping assembly is created, the code snippets are compiled into the mapping code. When the map is executed the code snippets are executed directly before the map instruction is carried out.

A map involving one or more built-in functions is equivalent to a map from a source (node, constant, value and so forth) to a target node that has a code snippet attached. In the example above the values of two source nodes are concatenated to produce a single value and then mapped to the target. Xe treats this as a map connection from the first source node to the target with an added code snippet to fetch the value of the second source node and join it to that of the first.

The generated code snippet is therefore attached to the connection from the function to the target node. Indeed, connections from source nodes to functions and those between functions can never have code snippets attached. The generated code snippet can be viewed by selecting the connection to the target node and then the code snippets tab in the bottom left section of the screen.

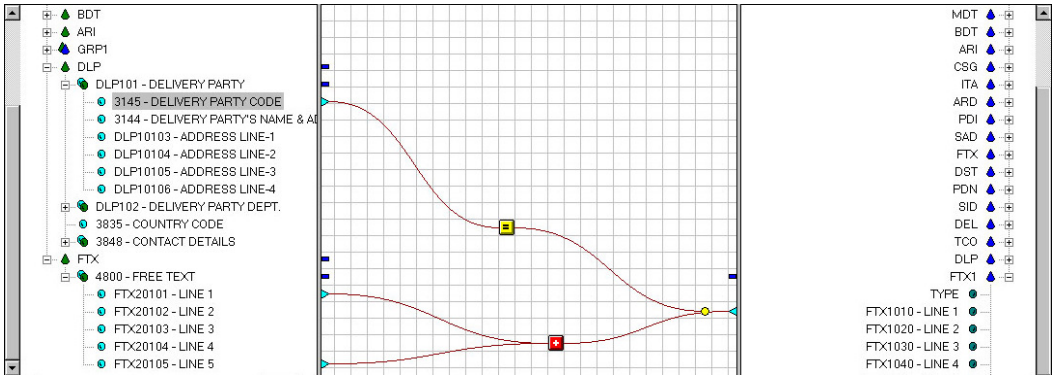


The code is generated in the language selected in the Global Properties tab (C# or VB.NET). The code can be viewed but not edited. If you required customised functionality you will need to write your own code snippet instead of using built-

in functions. You could use the code generated by the built-in functions as a template for your own code snippets.

5.14.4 Using logic functions

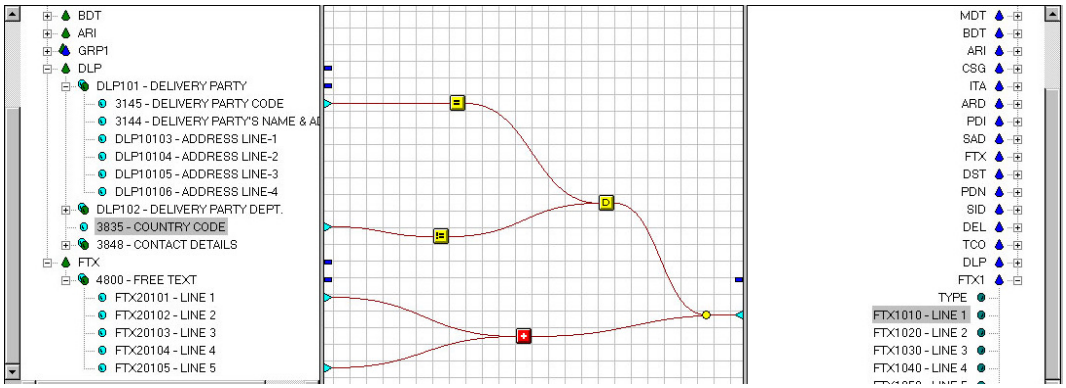
Logic functions are provided which can be used to test conditions. There are a number of comparison functions (equal, greater-than and so forth) as well as the Boolean operators AND and OR, which can be used to combine the outputs of other instances of logic functions. Logic functions can be dragged to the mapping grid in the same way as other functions, but their outputs are not mapped to target nodes, but to other map connections.



The result of this arrangement of functions is that the output of the **String.Concatenate** function is performed only if the output of the **Logic.Equal** function is TRUE. In this case, the logic function is comparing the value of a source node (“delivery party code”) with a literal value (“QWE345”). The code snippet generated for the logic function is part of the snippet for the map connection to the target node.

Multiple tests for equality and inequality can be carried out by providing multiple values for comparison (connections from source nodes, values and so forth) to a single function. The **Logic.Equal** function returns true if the first parameter is equal to any of the others. The **Logic.NotEqual** function returns true if the first parameter is not equal to each of the others.

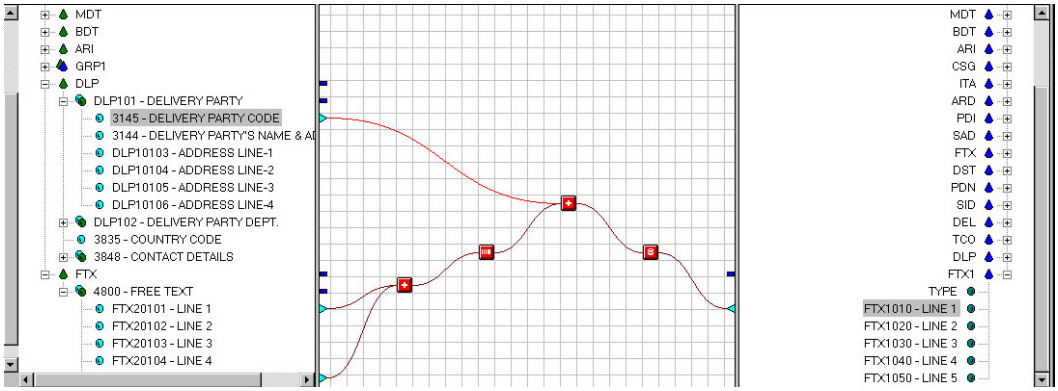
Multiple logic functions can be combined to create complex logic tests, using the Boolean AND and OR operators.



In this example the **Logic.And** function is used to combine the outputs of two other logic tests. Only if both of these tests are true will the map be performed.

5.14.5 Advanced function use

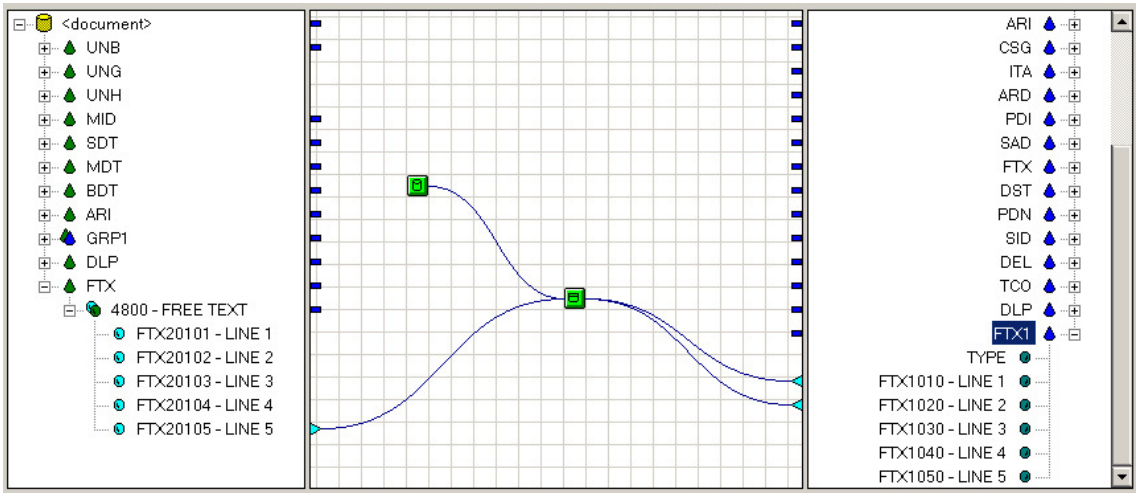
Functions can be combined with one another for complex processing of data.



The output from one function is used as the input to another. In this case, we are carrying out a **String.Concatenate**, followed by **String.Left**, followed by **StringConcatenate**, followed by **String.ReplaceRx**. You may combine functions to an arbitrary level of complexity, though if you are frequently using many levels of built-in functions, you might want to consider writing your own code snippets to do the processing instead.

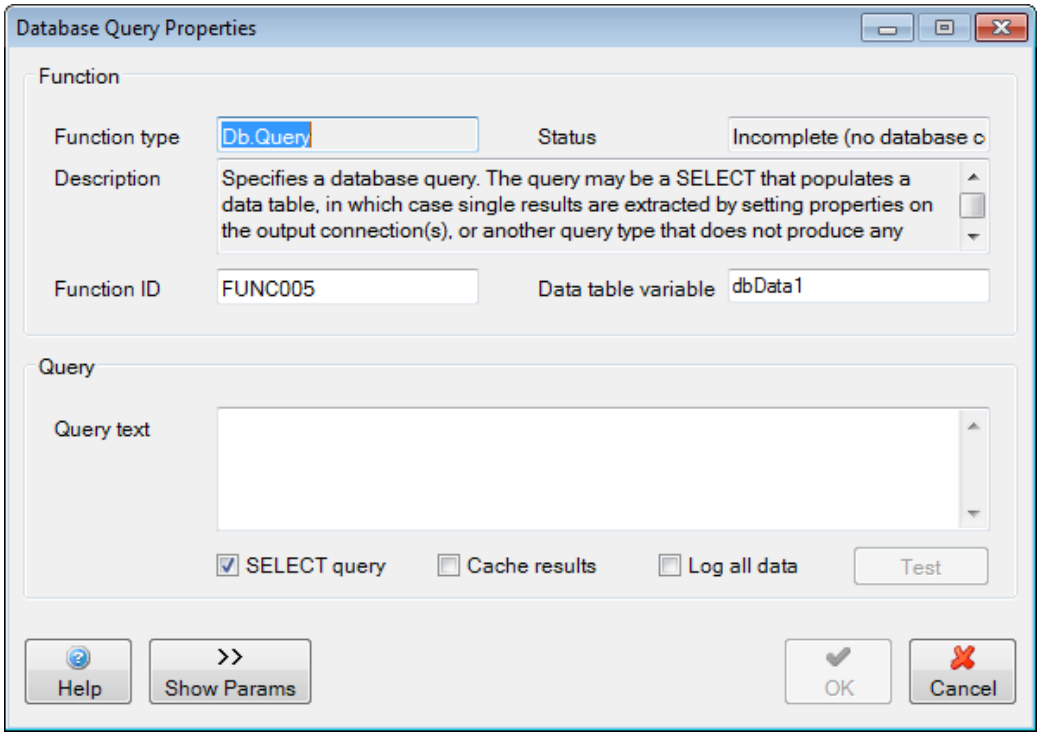
5.14.6 Database function use

Two database functions are available. They are always used together. The function **Db.Connect** defines a connection to a database (SQL Server, Microsoft Access or ODBC). **Db.Query** defines a query to be executed, which may be a SELECT query that returns results to be mapped:



The image shows a database connection function as the input to a database query function. The query function is using a value from the source as a parameter in the query and is mapping query results to locations in the target.

As a query may return multiple results (many rows and/or many columns of data) it is possible to select the value to be mapped as a property of the connection to the target. To do this, right-click on the connection and choose **Select Result** from the menu that appears. The following dialog will be displayed:

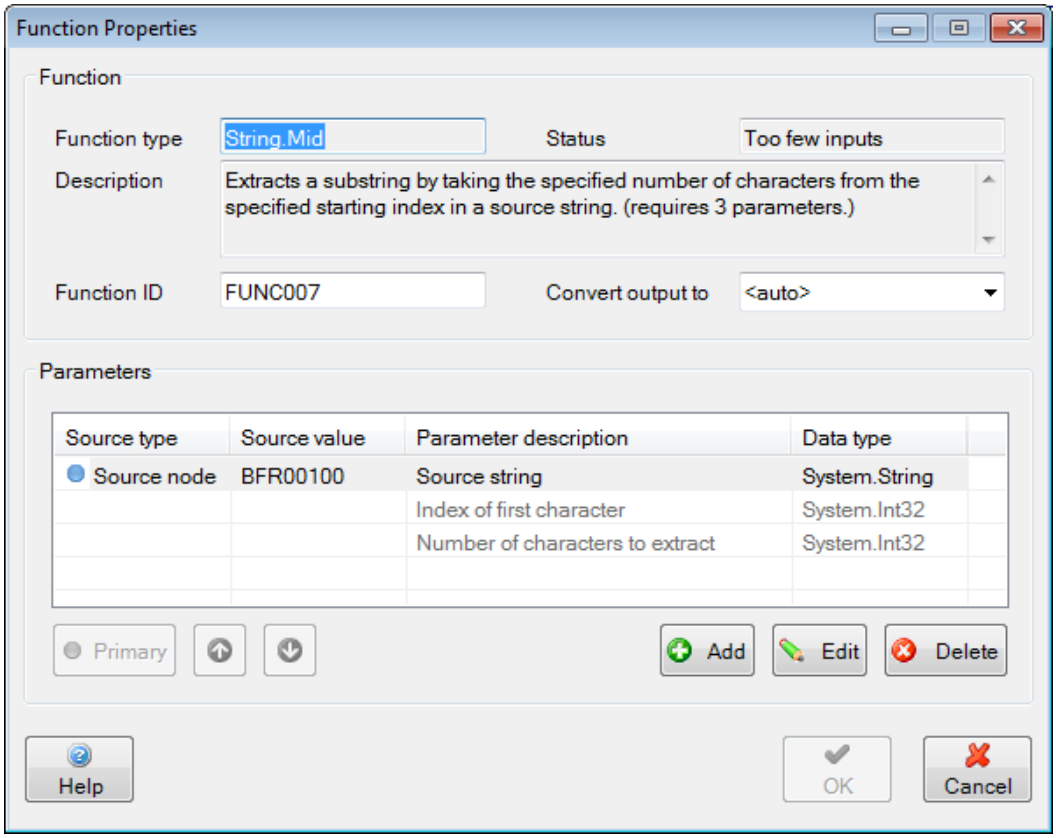


To set one or more of the properties for selecting a result, check the appropriate box and enter the required value. In the example above we are selecting the value from the column named 'Code' in row three of the returned data. If values are not entered, Xe defaults to using row index and column index zero (the indices are zero-based). If a column index and a column name are specified the name will be used in preference to the index.

For more information about using the database functions see Database connection function and Database query function.

5.14.7 Function properties

When you choose to edit a function, the function properties dialog will be displayed:



The following information is shown on this dialog:

Function type – The descriptive name of the function (read-only).

Status – The status of the function; whether it has enough input parameters defined and whether it can accept more (read-only).

Description – A description of the action performed by the function (read-only).

Function ID – The unique ID by which Xe identifies the function. This value is assigned automatically when the function is created, but it can be changed here if you wish. For example, you may want to provide an identifier that describes the purpose of the function in your map.

Output data type – Each function has an implicit output data type, for instance, the **String.Mid** function shown always produces a string. The output data type field of the dialog can be used to specify an explicit conversion to another data type. For example, if you known that the string produced by the function will be numeric, you can select a numeric type in the drop-down list (System.Int32, System.Single etc.) and the code snippet generated by Xe for the function will include the conversion. For mathematical functions, the operation of this list is slightly different, in that the type chosen tells Xe to treat the source and target values as a particular type. In most cases, however, you will leave the selection as **<auto>** and allow Xe to handle any data type conversions that may be required.

Parameters –The input parameters already defined are displayed in a list. The list shows:

- Source type – the type of location from which a parameter value is mapped (for example, a source node, a literal value, etc.).
- Source value – the value that identifies the source (for example, the source node cross-reference, the literal value, etc.).

- Parameter description – describes the purpose of the parameter (i.e. what part it plays in the action performed by the function.)
- Parameter type – the data type expected for this parameter. If Xe determines that the data type of what is being mapped to the parameter does not match that expected, it will include an expression in the generated code snippet to convert to the required type.

In addition to the parameters that have been defined, the list also shows greyed-out descriptions and data types for any mandatory parameters that are missing (have not yet been defined).

Each defined parameter represents a map connection. This may be a connection that is visible on the GUI (from a source node or another function) or one that is hidden (from a literal value, constant, etc.) When parameters are added, edited or removed, the result is to add, edit or remove the associated map connections. The parameters can be manipulated using the buttons below the list view:

Primary

Click the **primary** button to cycle which parameter is to be regarded as the "primary" when the code snippet is generated. The primary parameter is the one that is treated as triggering the function to be performed. By default, the first parameter is shown as the primary. In most cases, you will specify a parameter whose type is "source node" to be the primary parameter. In more complex cases you may specify a parameter whose type is another function. In all cases, Xe will try to trace back through the primary parameters to find a source node that triggers the function. If a parameter is chosen to be the primary that cannot be traced back to a source node, that selection will be ignored when the code snippet is generated.

Up (arrow)

Click the **Up arrow** button to move a selected parameter up in the list.

Down (arrow)

Click the **Down arrow** button to move a selected parameter down in the list.

Add

The **Add** button will be enabled only if the function will allow more parameters to be specified. Click the button to add a parameter. Xe will display the function parameter properties dialog with default values. See "Function parameter properties".

Edit

The **Edit** button will be enabled only if an existing parameter in the list is selected. Click the button to edit a parameter. Xe will display the function parameter properties dialog with the values to be edited. See "Function parameter properties". A parameter can also be edited by double-clicking on the item in the list.

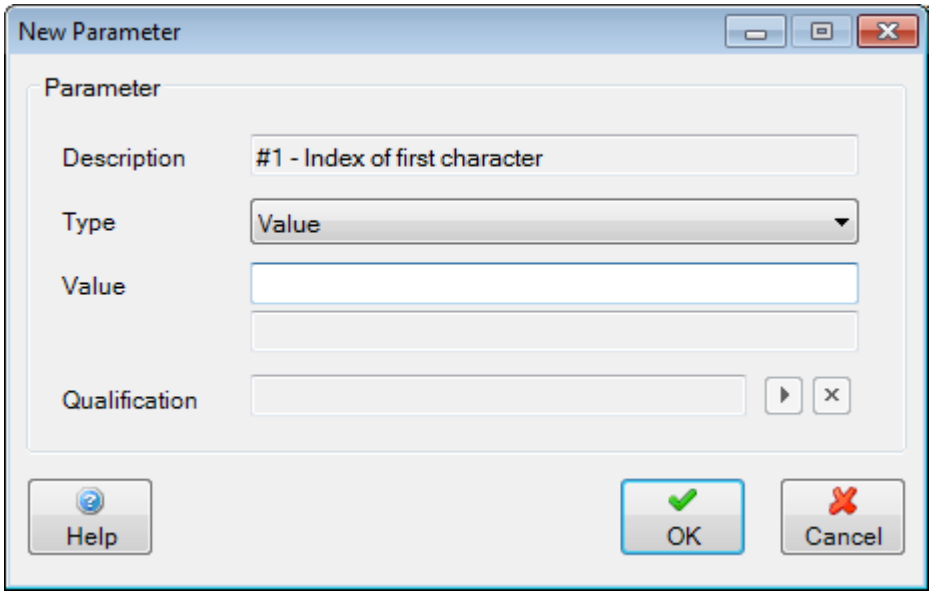
Delete

The **Delete** button will be enabled only if an existing parameter in the list is selected. Click the button to delete a parameter.

When you have finished entering data, click **OK** to accept the changes or **Cancel** to exit and lose the changes.

5.14.8 Function parameter properties

When you choose to edit a function, or to create a new one, the function parameter properties dialog will be shown:



The following information is shown on this dialog:

Description – Describes the purpose of the parameter (i.e. what part it plays in the action performed by the function.) prefaced by the parameter number, which is a zero-based index.

Type – The source entity type from which the parameter is mapped (for instance, source node, literal value etc.)

Value – The value identifying the source entity from which the parameter is mapped (for instance, the source node cross-reference, the literal value, etc.) If a source node is chosen as the type, and a cross-reference is selected as the value, the description of the node (if any) will be shown in the read-only text box.

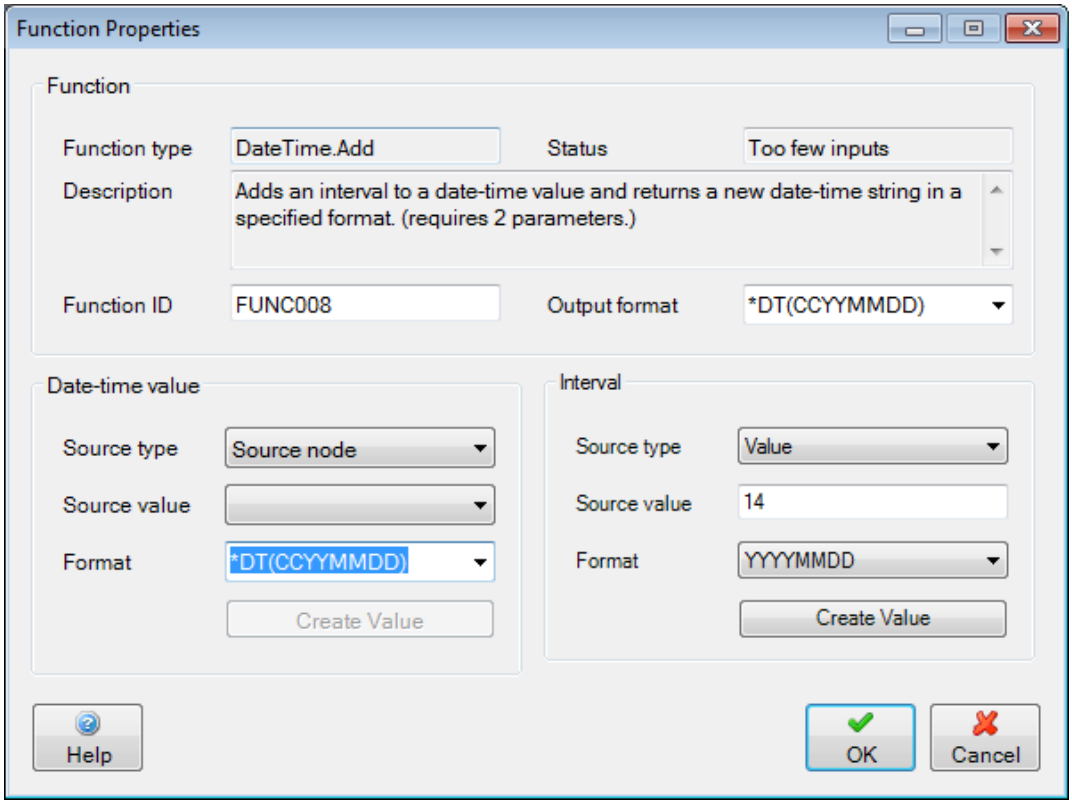
This dialog will be shown in two circumstances:

- Adding a new parameter. By default the dialog will be set up with a type of **Value**, meaning a literal value. Accept this type and provide a literal value, or change the type selected.
- Editing a parameter. The dialog will be populated with the type and value of the parameter to be edited. You can edit the type and/or value.

When you have finished entering data, click **OK** to accept the changes or **Cancel** to exit and lose the changes.

5.14.9 Date-time function properties

Date-time functions are not handled in the same way as functions of other types. This is because it is always necessary to specify the string format of a date-time value both on input and output. Therefore a separate dialog is used when editing the properties of these functions:



The following information is shown on this dialog:

Function type – The descriptive name of the function (read-only).

Status – The status of the function; whether it has enough input parameters defined and whether it can accept more (read-only).

Description – A description of the action performed by the function (read-only).

Function ID – The unique ID by which Xe identifies the function. This value is assigned automatically when the function is created, but it can be changed here if you wish. For example, you may want to provide an identifier that describes the purpose of the function in your map.

Output data type – The format in which the output date-time will be written, for example, *DT(CCYYMMDD) is equivalent to the standard EDI date format of **CCYYMMDD** or the .NET date-time format string of **yyyyMMdd**. The drop-down list contains a number of fixed formats. You can also choose a custom format in the style *CDT(fmt) where fmt is replaced by a .NET date-time format. For instance *CDT(F) produces output in this format “15 December 2005 10:15:25”. When the default combo box entry “*CDT(format_string)” is selected a dialog will be shown to help you pick a value (see Custom date-time formats). Alternatively, if you are familiar with .NET date-time formats, you can type one in directly.

Date-time source type – the source type of the input date-time value (source node, literal value, constant and so forth).

Date-time source type – the source value of the input date-time (source node cross-reference, constant name, literal value and so forth).

Date-time format – the format of the date-time source value. Permitted formats are the same as for output data type.

Interval source type – the source type of the input interval value to be added on to the date-time value (source node, literal value, constant and so forth).

Interval source type – the source value of the input interval value (source node cross-reference, constant name, literal value and so forth).

Interval format – the format of the interval value to be added on to the date-time source value. Permitted formats are YEARS, MONTHS, DAYS, HOURS, MINUTES, SECONDS, HHMMSS, YYYYMMDD and YYYYMMDDHHMMSS.

When specifying a date-time or an interval whose source type is "value", use the **CreateValue** button to display a dialog that will assist in creating the required value and format. See "Date-time and interval values".

The sample dialog shown above is for the **DateTime.Add** function and the dialog shown for the **DateTime.Subtract** function is identical, but the dialogs shown for the other date-time functions are different:

The **DateTime.Convert** and **DateTime.MonthEnd** functions require only one parameter so the "interval" section is not shown.

The **DateTime.Week** function outputs an integer so the output date-time format cannot be edited. Also, a "week" section is shown in place of the "interval" section. The "week" section allows you to set the rules for how weeks of the year are calculated:

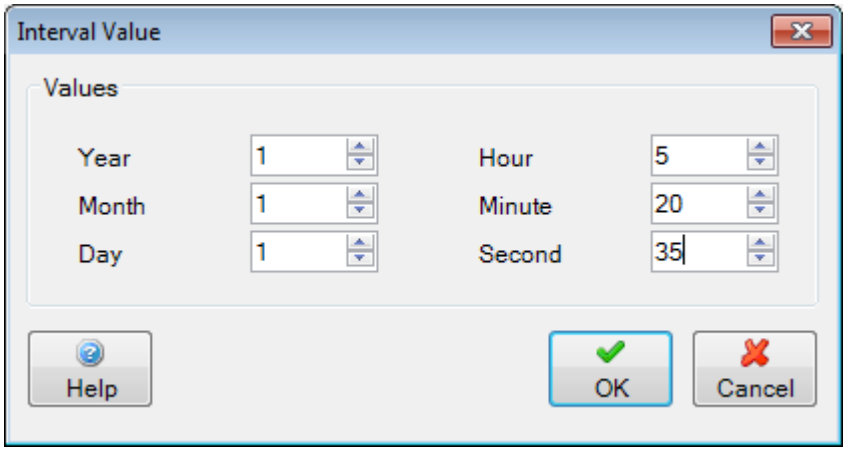
- Week start rule – the first week of the year is either the week of January 1st, or the first full week of the year.
- Week start day – the day on which the week starts (usually Sunday or Monday, but any day can be specified).

The **DateTime.WeekToDate** function requires input in the form YYWW or YYYYWW, so those are the formats shown in the drop-down list. This function also requires a "week" section in place of the "interval" section. The "week" section specifies the rules for how the required date and weeks of the year are calculated:

- Week start rule – the first week of the year is either the week of January 1st, or the first full week of the year.
- Week start day – the day on which the week starts (usually Sunday or Monday, but any day can be specified).
- Required day – the function returns a date given a week number; the day of the week whose date is required must be specified.

5.14.10 Date-time and interval values

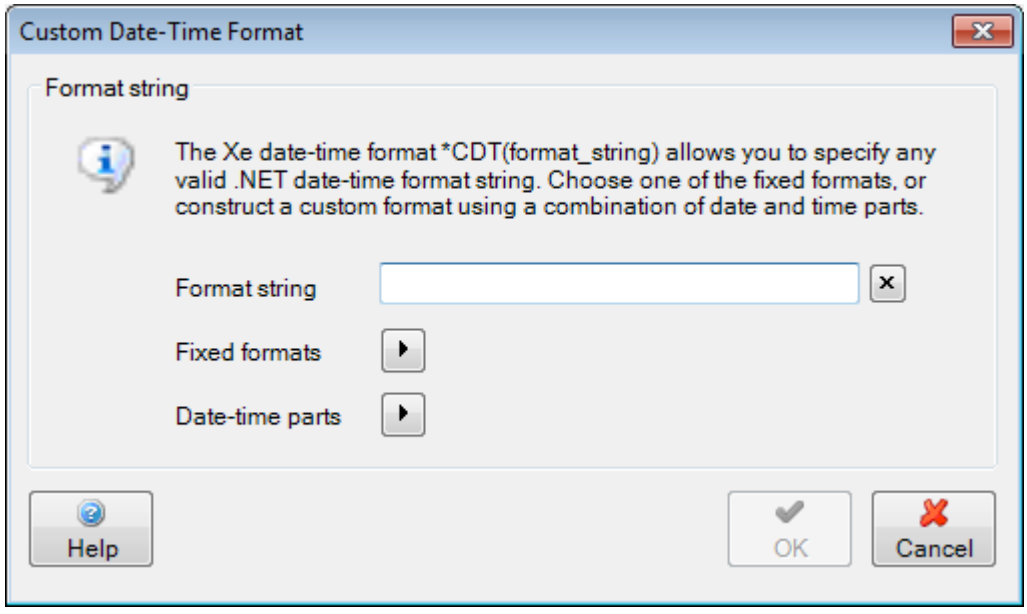
When specifying a date-time or an interval in the date-time function properties dialog (see "Date-time function properties") and the source type is "value", the **CreateValue** button will be enabled. When clicked this will show a dialog that will assist in creating the required value and format:



Any valid date-time or interval value may be specified either by using the up and down buttons against the date and time components, or by entering values directly with the keyboard. When the dialog is dismissed with the OK button, the function properties dialog will be updated with an appropriate source value and format.

5.14.11 Custom date-time formats

When you choose a date-time format of *CDT(format_string), the “format_string” part must be a valid .NET date-time format string. For instance, the .NET format string “dd-MM-yyyy” returns a date in the form “05-08-2007”. The dialog below is displayed to help you construct a valid format string.



There are two buttons that are used to select or build the format string:

- The **Fixed Formats** button offers a selection of standard (single character) format strings. Select one by clicking the button and then clicking on an item on the menu that appears. The format string that relates to the selected item will appear in the text box. You can only select one fixed format string at a time; selecting one will cause it to replace any existing format string in the text box.
- The **Date-time parts** button offers a selection of format substrings that can be built up to create a custom format string. Select any number of these in sequence by using the button and the menu that appears, adding separator and other characters as required directly to the text box.

Use the delete button (marked with a **X**) to clear the contents of the text box.

5.14.11.1 Date-time fixed formats

The table below shows the fixed date-time formats supported in .NET:

Format string	Output	Sample
d	Short date	7/2/2006
D	Long date	February 07 2006
f	Date-time	February 07 2006 22:20
F	Full date-time	February 07 2006 22:20:30
g	Short date, short time	7/2/2006 22:20
G	Short date, long time	7/2/2006 22:20:30
m	Day-month	07 February
r	RFC 1132	Tue, 07 Feb 2006 22:20:30 GMT
s	Sortable date-time	2006-02-07T22:20:30
t	Short time	22:20
T	Long time	22:20:30
u	Universal date-time	2006-02-07 22:20:30Z
y	Month-year	February 2006

5.14.11.2 Date-time parts

The table below shows the custom date-time format parts supported in .NET.

Note: When using a single character date-time part on its own, it must be preceded by the % sign so it is not confused with any of the fixed date-time formats (e.g. to display the current month number with the year use “M yyyy”, but to display the month number only use “%M”.

Format	Output
d	The day of the month (single digit values do not have leading zeros)
dd	The day of the month (single digit values do have leading zeros)
ddd	The short name of the day of the week (e.g. Tue)
dddd	The full name of the day of the week (e.g. Tuesday)
M	The month of the year (single digit values do not have leading zeros)
MM	The month of the year (single digit values do have leading zeros)
MMM	The short month name (e.g. Feb)

MMMM	The full month name (e.g February)
y	The year without the century (single digit values do not have leading zeros)
yy	The year without the century (single digit values do have leading zeros)
yyyy	The year with the century
gg	The period or era
h	The hour - 12 hour clock (single digit values do not have leading zeros)
hh	The hour - 12 hour clock (single digit values do have leading zeros)
H	The hour - 24 hour clock (single digit values do not have leading zeros)
HH	The hour - 24 hour clock (single digit values do have leading zeros)
m	Minutes (single digit values do not have leading zeros)
mm	Minutes (single digit values do have leading zeros)
s	Seconds (single digit values do not have leading zeros)
ss	Seconds (single digit values do have leading zeros)
f	Fraction of a second – single digit precision (other digits are truncated)
ff	Fraction of a second – two digit precision (other digits are truncated)
fff	Fraction of a second – three digit precision (other digits are truncated)
ffff	Fraction of a second – four digit precision (other digits are truncated)
fffff	Fraction of a second – five digit precision (other digits are truncated)
ffffff	Fraction of a second – six digit precision (other digits are truncated)
fffffff	Fraction of a second – seven digit precision (other digits are truncated)
F	Fraction of a second – most significant digit (trailing zeros not shown)
FF	Fraction of a second – two most significant digits (trailing zeros not shown)
FFF	Fraction of a second – three most significant digits (trailing zeros not shown)
FFFF	Fraction of a second – four most significant digits (trailing zeros not shown)

FFFFF	Fraction of a second – five most significant digits (trailing zeros not shown)
FFFFFF	Fraction of a second – six most significant digits (trailing zeros not shown)
FFFFFFF	Fraction of a second – seven most significant digits (trailing zeros not shown)
t	First character of the AM/PM designator
tt	The AM/PM designator
z	The time-zone hour offset – '+' or '–' followed by the hour (single digit values do not have leading zeros)
zz	The time-zone hour offset – '+' or '–' followed by the hour (single digit values do have leading zeros)
zzz	The full time-zone offset – '+' or '–' followed by the hour and minutes (single digit values do have leading zeros)
:	The default time separator
/	The default date separator
\c	Used to escape a character; for instance, to display the time in the format "11H-05m" use the format string "HH\H-mm\m"

5.14.12 Data type conversions and data types

Built-in functions are defined with an implicit output data type (for instance **String.Left** will output a string) and required input types for each parameter (**String.Left** expects one parameter that is a string and another that is an integer). Data loaded from the source document is always stored in Xe in strings, so it is sometimes necessary for the generated code snippets to perform conversions on data before and after the substantive operation indicated by the function.

You can force a conversion to a different data type than the one implicit in the function definition, using the output data type list in the function parameters dialog (see "Function properties"). However, this will rarely be necessary, and it is recommended that you allow Xe to handle conversions. The exception is when using date and time functions, which require the formats of date-time and interval strings to be specified.

For some function types, Xe handles the problem of data type conversion by using the **var** or VARIANT data type. In particular, the logic functions and mathematical functions use the var type extensively.

Logic functions that take **var** types as inputs can, for instance, compare numeric values where one is in string format (because it comes from a source node) and the other is in a numeric variable (because it has been output by a mathematical function).

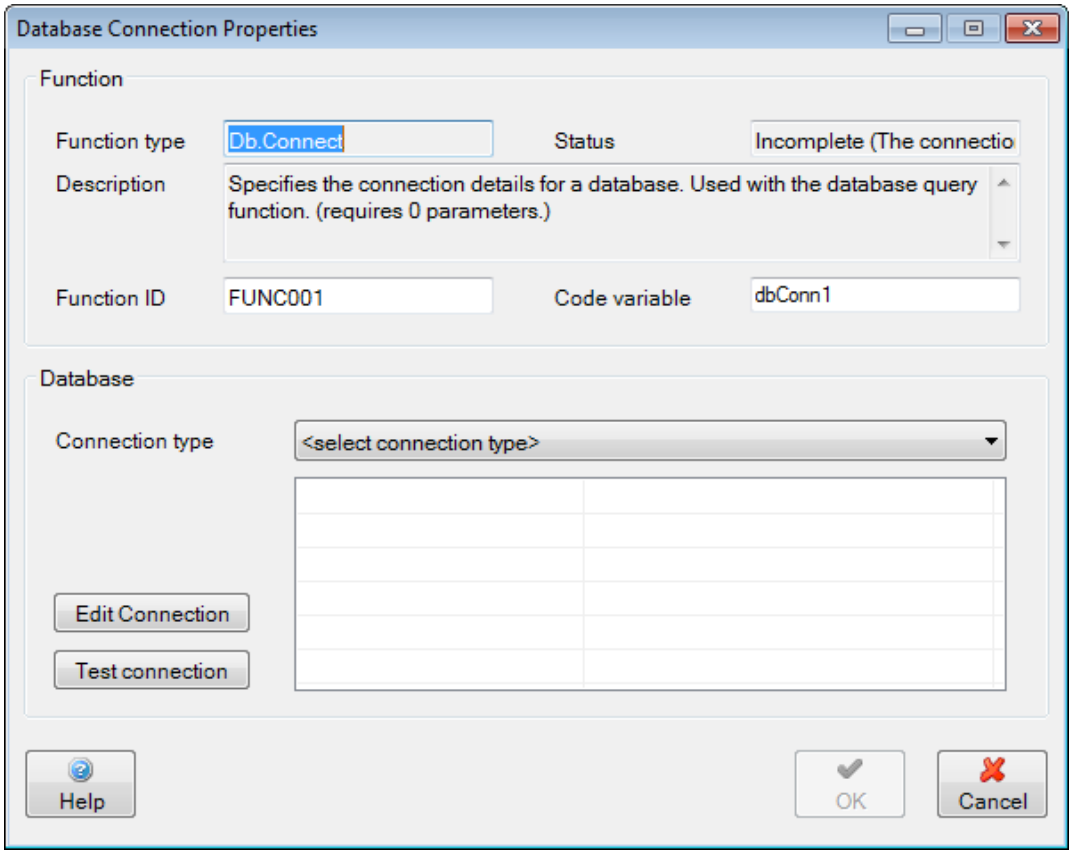
Similarly, mathematical functions can accept and output any integer or floating point type and any numerical string by using the **var** type to hide the precise system type used. If you specify a value in the data type list on the function properties dialog, then that is the type that will be used when the function is

executed. Note, however, that a runtime error will occur if the value supplied is not convertible to the type specified.

The **var** type is internal to Xe, is not related to variant types used in COM, and is only there to support Xe's built-in functions. Other types that you may see in the generated code snippets are **xedate**, **interval**, **substring** and **regex**. Like **var** these are used solely to implement functions within Xe. You may copy and change generated code snippets that make use of these types, but no additional documentation is provided about their interfaces, and their behaviour is not guaranteed to remain the same between major releases of Xe.

5.14.13 Database connection function

The function **Db.Connect** defines a connection to a database that can be used in conjunction with one or more database query functions. For details about using the **Db.Query** function see Database query function. Database functions are not handled in the same way as functions of other types so a separate dialog is used when editing the properties of these functions:



The following information is shown on this dialog:

Function type – The descriptive name of the function (read-only).

Status – The status of the function (read-only). Indicates whether sufficient information has been defined for it to create a database connection. Note that a status of OK does not indicate that a connection to the database will be successful, only that an attempt is possible.

Description – A description of the action performed by the function (read-only).

Function ID – The unique ID by which Xe identifies the function. This value is assigned automatically when the function is created, but it can be changed here if you wish. For example, you may want to provide an identifier that describes the purpose of the function in your map.

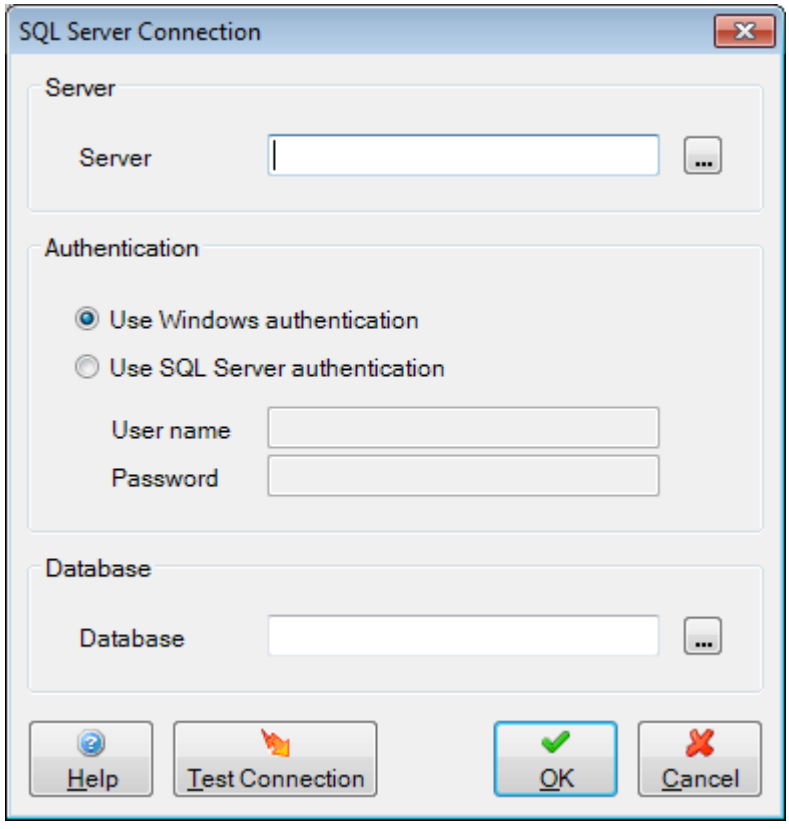
Code variable – The name by which the connection will be known in the code generated to perform the map. The connection is opened before the map is executed and automatically closed after the mapping finishes. The variable is therefore accessible from any local code snippet.

Connection type – The type of database connection, which can be SQL Server, Microsoft Access or ODBC. When the selection is changed a dialog is displayed for editing connection properties. These properties are shown in the list view. If the properties include a password it will be hidden in the application using asterisks and encrypted before being saved to the map script and the generated code, so it will never appear in plain text to an unauthorised observer. However, please bear in mind that with access to Xe it is possible to write maps with full access to the database.

To show the edit dialog for the currently selected connection type and currently entered properties use the **Edit connection** button. To attempt to make a connection to the database using the currently configured details use the **Test connection** button

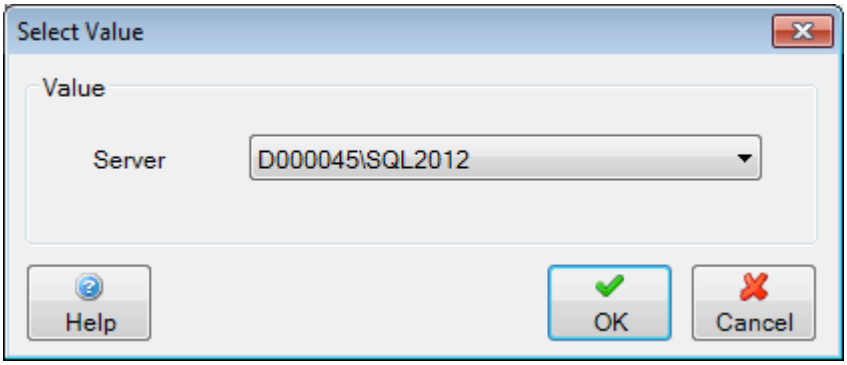
5.14.13.1 SQL Server connection

The details required to connect to a SQL Server database are entered in the following dialog (note that it is also possible to connect to SQL Server using an ODBC connection, if required):



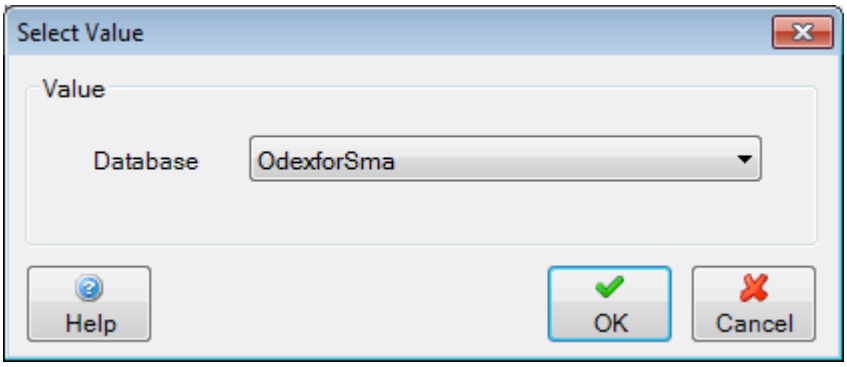
The following information is shown on this dialog:

Server – The name of the SQL Server to connect to. Use the button to display a list of servers visible on your network, or type the name of a server you know to be present.



Authentication – Choose between Windows authentication (which requires that the logged on Windows user have access to the SQL server instance and database) and SQL Server authentication (which requires a user name and password provided by your database administrator).

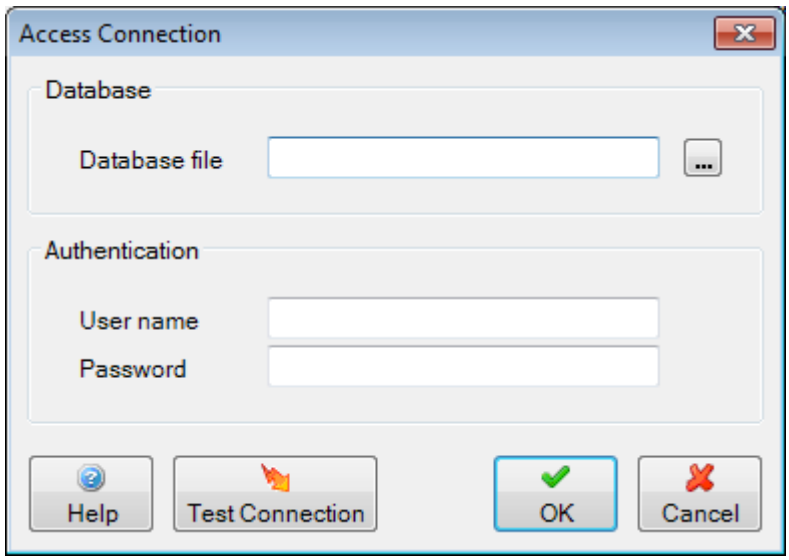
Database – The name of the database to connect to. Use the button to display a list of databases available on the currently selected server, or type the name of a database you know to be present:



Use the **Test connection** button to attempt a connection using the currently selected values (note that a connection will also be attempted if you elect to show the list of databases seen above).

5.14.13.2 Access connection

The details required to connect to a Microsoft Access database are entered in the following dialog (note that it is also possible to connect to Access using an ODBC connection, if required):



The following information is shown on this dialog:

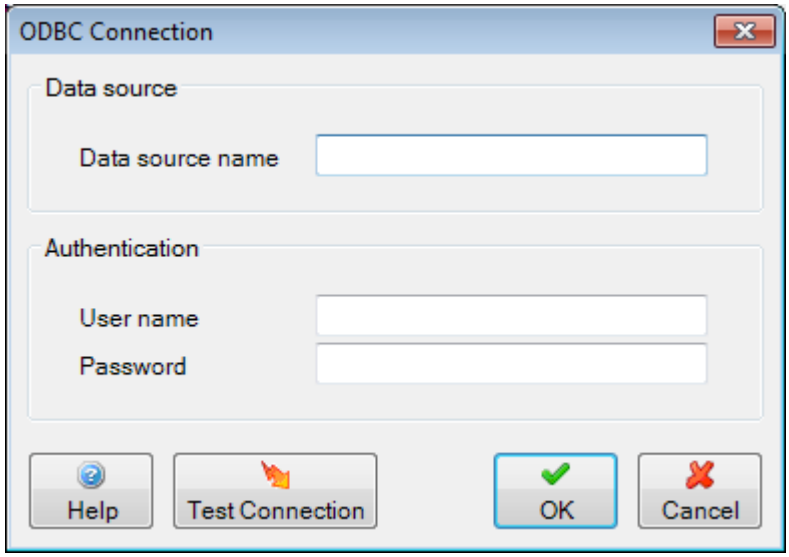
Database – The filename of the database to connect to. Use the button to browse for an Access database file or type the full path of a database you know to be present.

Authentication – If the database is password protected, enter authentication details here.

Use the **Test connection** button to attempt a connection using the currently selected values.

5.14.13.3 ODBC connection

The details required to connect to a data source known to ODBC:



The following information is shown on this dialog:

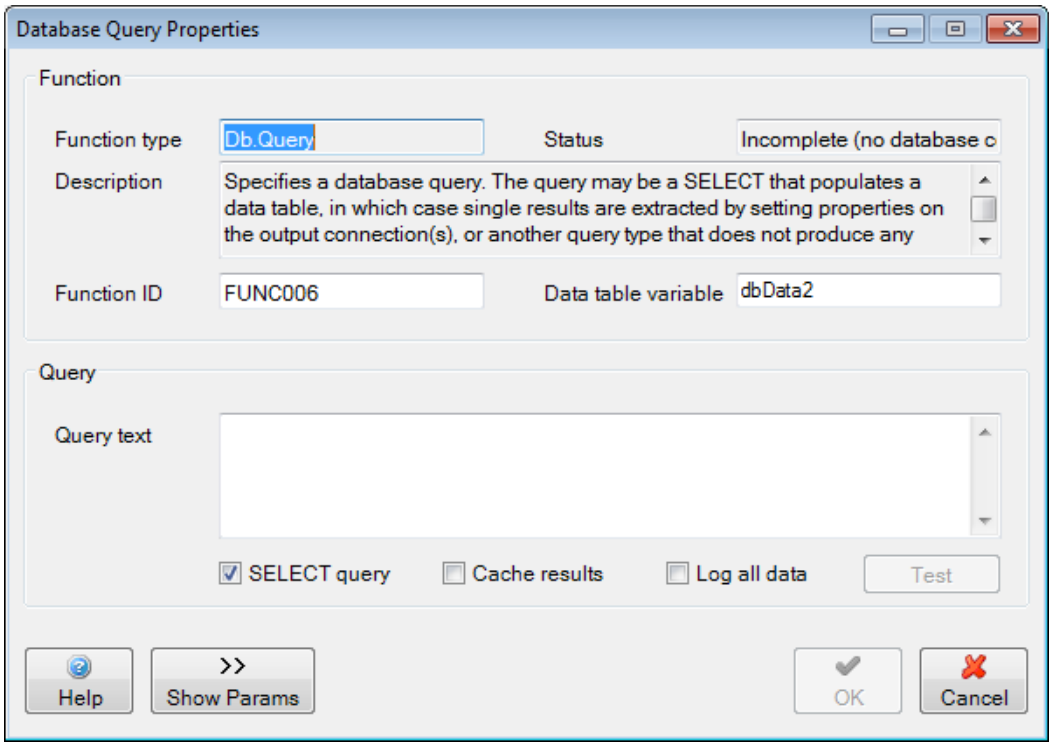
Data source – The data source name (DSN) of the database to connect to. ODBC data sources are configured using the Windows ODBC administrator utility, odbcad32.exe. The value entered here should match the name given to a data source in that utility. To configure a data source go to **Start > Run** and type 'odbcad32',

Authentication – If the database is password protected, enter authentication details here (note that authentication details may alternatively be entered against the data source in the ODBC administrator).

Use the **Test connection** button to attempt a connection using the currently selected values.

5.14.14 Database query function

The function **Db.Query** defines a query to be performed on a database. It is always executed against a database defined by a database connection function. For details about using the **Db.Connect** function see Database connection function. Database functions are not handled in the same way as functions of other types so a separate dialog is used when editing the properties of these functions:



The following information is shown on this dialog:

Function type – The descriptive name of the function (read-only).

Status – The status of the function (read-only). Indicates whether sufficient information has been defined for it to generate a code snippet. Note that a status of OK does not indicate that the query will be successful, or that any results will be returned.

Description – A description of the action performed by the function (read-only).

Function ID – The unique ID by which Xe identifies the function. This value is assigned automatically when the function is created, but it can be changed here if you wish. For example, you may want to provide an identifier that describes the purpose of the function in your map.

Data table variable – If the query is a SELECT then it will fetch results from the database into a data table (an object of the type System.Data.DataTable). A default name for the data table variable is inserted when the function is created. Generally it will be of interest only to advanced users and should not be changed.

Query – The text of the query to be executed is typed into the text box. No syntax checking is performed on the entered text. If the query is a SELECT then the **SELECT query** box should be checked. This indicates that the query should return data and store it in a data table. If this option is selected then other options are enabled:

- **Cache results** – Specifies that where the query is used to map values to more than one target it should be executed only once and the results cached. Use this option to reduce execution time for the map, but only if you are certain that the results of the query will be the same each time.
- **Log all data** – Specifies that all of the data brought back by a SELECT query will be logged. By default, only minimal logging of database operations is carried out.
- **Test** – The test button allows you to test a query. Click the button to display the Test Query dialog.

Parameters – This section allows you to configure inputs to the function. It is functionally identical to the parameters section of the standard function dialog (see Function properties) with the exception of the **Db.Connect** button. The query function requires that a database connection is specified. This is done by defining a **Db.Connect** function and connecting it to the query function. As the connection represents a special parameter it is not shown with the others. To view or edit the select connection function, click the button. The following dialog is displayed:

The dialog shows the currently selected connection function and allows you to select a different one from the drop-down list.

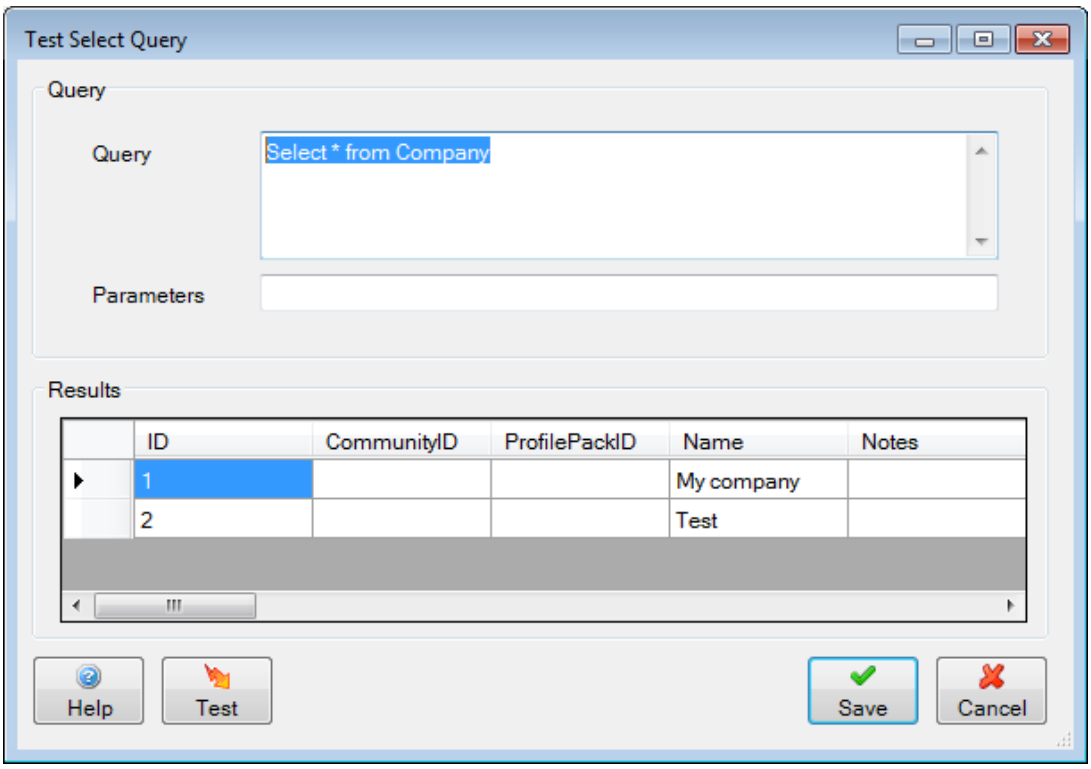
Other connections to a query function represent user parameters that you might want to incorporate into database queries. These parameters are addressed in the query text using placeholders in the form `%P1%`, `%P2%` &c. For instance, to look up a code in a database given another code from the source document, first create a connection from the source node to the query function, then write SQL in this form:

```
SELECT BuyerCode FROM Parties WHERE EdiCode = '%P1%'
```

The value of the first parameter will be substituted for `%P1%` when the map is executed. Take care when writing your SQL. Remember that only the value of the source node will be inserted, and include single quotes around text values where necessary.

5.14.14.1 Test Query

When you click the test button on the Database query function dialog, the query test dialog will be displayed:



The following information is shown on this dialog:

Query – The query text from the **Db.Query** function is displayed here. You can edit this text if required.

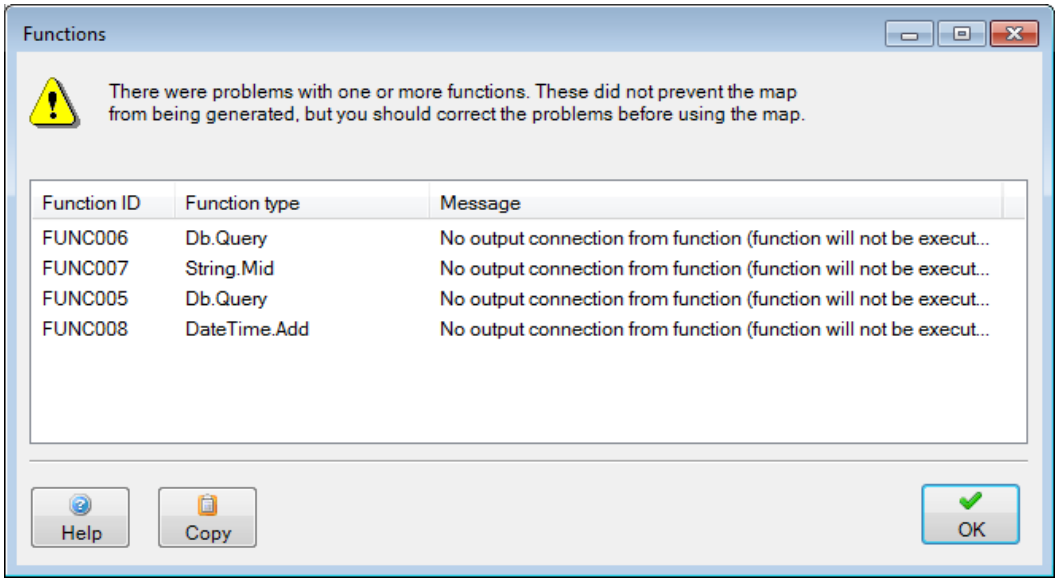
Parameters – If the query text includes parameters (`%P1%`, `%P2%` &c) then you should enter a comma separated list of values which will be substituted for the parameters when the query is executed. In the example above, the value 5 is substituted for `%P1%` and 7 for `%P2%`.

Results –When the dialog is first displayed, Xe will attempt to execute the query and the results, if any, will be displayed in this grid. In the event that there are no results, the grid will remain blank. If there is an error executing the query then a dialog box will be displayed that contains an error description. You can edit the query text and/or parameter string at any time and re-run the query using the **Test** button. The grid will be updated with the new set of results.

If you edit the query text then choosing **Save** to close the dialog will result in the new text being carried back to the **Db.Query** function properties dialog. If you choose **Cancel** then your changes will be discarded. **Save** also has the effect of preserving the parameter string you entered.

5.14.15 Function problems

When the map script is generated Xe checks the built-in functions for problems and, if so configured, displays a warning dialog:



Two types of problem are checked for:

- Functions with no output connections - functions without outputs do not result in any code being generated, so do not prevent the map script from being written, but they may indicate that something has been overlooked.
- Functions for which code could not be generated - functions in error do not prevent the map script from being written, but will result in the map failing when executed. The most common cause of errors is failure to provide enough input parameters to a function.

Choose **OK** to dismiss the dialog and continue using Xe. Choose **Copy** to copy the list of errors to the clipboard.

5.14.16 Available functions

The following sections provide details about the built-in functions:

5.14.16.1 String.Concatenate

The function has these properties:

Property	Value
Display name	String.Concatenate
Script name	STR-CONCAT
Description	Concatenates two or more input strings to create a single output string
Category	String functions
Implicit output type	System.String
Maximum parameters	10
Minimum parameters	2

It takes these parameters:

#	Description	Data type	Required
0	Source string	System.String	Y
1	Source string	System.String	Y

2..9	Source string	System.String	N

5.14.16.2 String.Left

The function has these properties:

Property	Value
Display name	String.Left
Script name	STR-LEFT
Description	Extracts a substring by taking the specified number of characters from the left-hand side of a source string
Category	String functions
Implicit output type	System.String
Maximum parameters	2
Minimum parameters	2
Notes	If there are insufficient characters in the source string to extract the required number then the whole string is returned

It takes these parameters:

#	Description	Data type	Required
0	Source string	System.String	Y
1	Number of characters to extract	System.Int32	Y

5.14.16.3 String.Right

The function has these properties:

Property	Value
Display name	String.Right
Script name	STR-RIGHT
Description	Extracts a substring by taking the specified number of characters from the right-hand side of a source string
Category	String functions
Implicit output type	System.String
Maximum parameters	2
Minimum parameters	2
Notes	If there are insufficient characters in the source string to extract the required number then the whole string is returned

It takes these parameters:

#	Description	Data type	Required
0	Source string	System.String	Y
1	Number of characters to extract	System.Int32	Y

5.14.16.4 String.Mid

The function has these properties:

Property	Value
Display name	String.Mid
Script name	STR-MID
Description	Extracts a substring by taking the specified number of characters from the specified starting index in a source string
Category	String functions
Implicit output type	System.String
Maximum parameters	3
Minimum parameters	3
Notes	If there are insufficient characters in the source string after the start point to extract the required number then the remainder of the string after the start point is returned

It takes these parameters:

#	Description	Data type	Required
0	Source string	System.String	Y
1	Index of first character	System.Int32	Y
2	Number of characters to extract	System.Int32	Y

5.14.16.5 String.Length

The function has these properties:

Property	Value
Display name	String.Length
Script name	STR-LENGTH
Description	Returns the number of characters in a source string
Category	String functions
Implicit output type	System.Int32
Maximum parameters	1
Minimum parameters	1

It takes these parameters:

#	Description	Data type	Required
0	Source string	System.String	Y

5.14.16.6 String.TrimLeft

The function has these properties:

Property	Value
Display name	String.TrimLeft
Script name	STR-TRIMLEFT
Description	Trims characters from the start of a source string. By default, whitespace characters are trimmed, but you may optionally specify a string of one or more other characters to be trimmed instead.
Category	String functions
Implicit output type	System.String
Maximum parameters	2
Minimum parameters	1

It takes these parameters:

#	Description	Data type	Required
0	Source string	System.String	Y
1	Character(s) to be trimmed	System.String	N

5.14.16.7 String.TrimRight

The function has these properties:

Property	Value
Display name	String.TrimRight
Script name	STR-TRIMRIGHT
Description	Trims characters from the end of a source string. By default, whitespace characters are trimmed, but you may optionally specify a string of one or more other characters to be trimmed instead.
Category	String functions
Implicit output type	System.String
Maximum parameters	2
Minimum parameters	1

It takes these parameters:

#	Description	Data type	Required
0	Source string	System.String	Y

1	Character(s) to be trimmed	System.String	N

5.14.16.8 String.ToLower

The function has these properties:

Property	Value
Display name	String.ToLower
Script name	STR-TOLOWER
Description	Converts all of the alphabetic characters in a source string to lowercase.
Category	String functions
Implicit output type	System.String
Maximum parameters	1
Minimum parameters	1

It takes these parameters:

#	Description	Data type	Required
0	Source string	System.String	Y

5.14.16.9 String.ToUpper

The function has these properties:

Property	Value
Display name	String.ToUpper
Script name	STR-TOUPPER
Description	Converts all of the alphabetic characters in a source string to uppercase.
Category	String functions
Implicit output type	System.String
Maximum parameters	1
Minimum parameters	1

It takes these parameters:

#	Description	Data type	Required
0	Source string	System.String	Y

5.14.16.10 String.Find

The function has these properties:

Property	Value
Display name	String.Find

Script name	STR-FIND
Description	Searches for a substring in a source string and returns the zero-based index position of the first occurrence of it or –1 if no instance is found.
Category	String functions
Implicit output type	System.Int32
Maximum parameters	2
Minimum parameters	2

It takes these parameters:

#	Description	Data type	Required
0	Source string	System.String	Y
1	Substring to find	System.String	Y

5.14.16.11 String.FindRx

The function has these properties:

Property	Value
Display name	String.FindRx
Script name	STR-FINDRX
Description	Searches for a substring, given by a regular expression pattern, in a source string, and returns the zero-based index position of the first occurrence of it or –1 if no instance is found.
Category	String functions
Implicit output type	System.Int32
Maximum parameters	2
Minimum parameters	2

It takes these parameters:

#	Description	Data type	Required
0	Source string	System.String	Y
1	Regular expression pattern to find	System.String	Y

5.14.16.12 String.Replace

The function has these properties:

Property	Value
Display name	String.Replace
Script name	STR-REPLACE
Description	Replaces all occurrences of a substring in a source string.

Category	String functions
Implicit output type	System.String
Maximum parameters	3
Minimum parameters	3

It takes these parameters:

#	Description	Data type	Required
0	Source string	System.String	Y
1	Substring to replace	System.String	Y
2	Substring to insert	System.String	Y

5.14.16.13 String.ReplaceRx

The function has these properties:

Property	Value
Display name	String.ReplaceRx
Script name	STR-REPLACERX
Description	Replaces all occurrences of a substring, given by a regular expression pattern, in a source string.
Category	String functions
Implicit output type	System.String
Maximum parameters	3
Minimum parameters	3

It takes these parameters:

#	Description	Data type	Required
0	Source string	System.String	Y
1	Regex pattern to replace	System.String	Y
2	Substring to insert	System.String	Y

5.14.16.14 String.Compare

The function has these properties:

Property	Value
Display name	String.Compare
Script name	STR-COMPARE
Description	Performs an alphanumeric comparison between two strings. The value returned is positive if the first string is ‘greater than’ the second, negative if the firs string is ‘less than’ the second, and zero if the strings are identical.

Category	String functions
Implicit output type	System.Int32
Maximum parameters	2
Minimum parameters	2

It takes these parameters:

#	Description	Data type	Required
0	First string to compare	System.String	Y
1	Second string to compare	System.String	Y

5.14.16.15 Logic.Equal

The function has these properties:

Property	Value
Display name	Logic.Equal
Script name	LOG-EQUAL
Description	Performs a comparison between a supplied value (l-val) and one or more additional values (r-val). It returns TRUE if there is a match with any one of them, otherwise it returns FALSE.
Category	Logic functions
Implicit output type	System.Boolean
Maximum parameters	10
Minimum parameters	2

It takes these parameters:

#	Description	Data type	Required
0	First value to compare (l-val)	var	Y
1	Second value to compare (r-val)	var	Y
2..9	Additional values to compare (r-val)	var	N

5.14.16.16 Logic.NotEqual

The function has these properties:

Property	Value
Display name	Logic.NotEqual
Script name	LOG-NEQUAL
Description	Performs a comparison between a supplied value (l-val) and one or more additional values (r-val). It returns FALSE if there is a match with any one of them, otherwise it returns TRUE.
Category	Logic functions

Implicit output type	System.Boolean
Maximum parameters	10
Minimum parameters	2

It takes these parameters:

#	Description	Data type	Required
0	First value to compare (l-val)	var	Y
1	Second value to compare (r-val)	var	Y
2..9	Additional values to compare (r-val)	var	N

5.14.16.17 Logic.GreaterThan

The function has these properties:

Property	Value
Display name	Logic.GreaterThan
Script name	LOG-GREATER
Description	Performs a comparison between a supplied value (l-val) and one other value (r-val). It returns TRUE if the first is greater than the second, otherwise it returns FALSE.
Category	Logic functions
Implicit output type	System.Boolean
Maximum parameters	2
Minimum parameters	2

It takes these parameters:

#	Description	Data type	Required
0	First value to compare (l-val)	var	Y
1	Second value to compare (r-val)	var	Y

5.14.16.18 Logic.LessThan

The function has these properties:

Property	Value
Display name	Logic.LessThan
Script name	LOG-LESS
Description	Performs a comparison between a supplied value (l-val) and one other value (r-val). It returns TRUE if the first is less than the second, otherwise it returns FALSE.
Category	Logic functions
Implicit output type	System.Boolean

Maximum parameters	2
Minimum parameters	2

It takes these parameters:

#	Description	Data type	Required
0	First value to compare (l-val)	var	Y
1	Second value to compare (r-val)	var	Y

5.14.16.19 Logic.GreaterThanOrEqual

The function has these properties:

Property	Value
Display name	Logic.GreaterThanOrEqual
Script name	LOG-GREATEREQ
Description	Performs a comparison between a supplied value (l-val) and one other value (r-val). It returns TRUE if the first is greater than or equal to the second, otherwise it returns FALSE.
Category	Logic functions
Implicit output type	System.Boolean
Maximum parameters	2
Minimum parameters	2

It takes these parameters:

#	Description	Data type	Required
0	First value to compare (l-val)	var	Y
1	Second value to compare (r-val)	var	Y

5.14.16.20 Logic.LessThanOrEqual

The function has these properties:

Property	Value
Display name	Logic.LessThanOrEqual
Script name	LOG-LESSEQ
Description	Performs a comparison between a supplied value (l-val) and one other value (r-val). It returns TRUE if the first is less than or equal to the second, otherwise it returns FALSE.
Category	Logic functions
Implicit output type	System.Boolean
Maximum parameters	2
Minimum parameters	2

It takes these parameters:

#	Description	Data type	Required
0	First value to compare (l-val)	var	Y
1	Second value to compare (r-val)	var	Y

5.14.16.21 Logic.Range

The function has these properties:

Property	Value
Display name	Logic.Range
Script name	LOG-RANGE
Description	Performs a comparison between a supplied value and two limits. If the value is in the range given by the limits (greater than or equal to the first, less than or equal to the second) then the function returns TRUE, otherwise it returns FALSE.
Category	Logic functions
Implicit output type	System.Boolean
Maximum parameters	3
Minimum parameters	3

It takes these parameters:

#	Description	Data type	Required
0	Value to compare	var	Y
1	Lower limit of range	var	Y
2	Upper limit of range	var	Y

5.14.16.22 Logic.Or

The function has these properties:

Property	Value
Display name	Logic.Or
Script name	LOG-OR
Description	Implements the Boolean OR operator. It combines two or more Boolean values to produce a single output. If at least one of the inputs is TRUE the function returns TRUE, otherwise it returns FALSE
Category	Logic functions
Implicit output type	System.Boolean
Maximum parameters	10

Minimum parameters	2

It takes these parameters:

#	Description	Data type	Required
0	Boolean value	System.Boolean	Y
1	Boolean value	System.Boolean	Y
2..9	Boolean value	System.Boolean	N

5.14.16.23 Logic.And

The function has these properties:

Property	Value
Display name	Logic.And
Script name	LOG-AND
Description	Implements the Boolean AND operator. It combines two or more Boolean values to produce a single output. If all of the inputs is TRUE the function returns TRUE, otherwise it returns FALSE
Category	Logic functions
Implicit output type	System.Boolean
Maximum parameters	10
Minimum parameters	2

It takes these parameters:

#	Description	Data type	Required
0	Boolean value	System.Boolean	Y
1	Boolean value	System.Boolean	Y
2..9	Boolean value	System.Boolean	N

5.14.16.24 Maths.Add

The function has these properties:

Property	Value
Display name	Maths.Add
Script name	MTH-ADD
Description	Takes two or more numerical values and returns their sum.
Category	Mathematical functions
Implicit output type	var
Maximum parameters	10

Minimum parameters	2

It takes these parameters:

#	Description	Data type	Required
0	Numerical value	var	Y
1	Numerical value	var	Y
2..9	Numerical value	var	N

5.14.16.25 Maths.Subtract

The function has these properties:

Property	Value
Display name	Maths.Subtract
Script name	MTH-SUBTRACT
Description	Subtracts one or more numerical values from another and returns the result.
Category	Mathematical functions
Implicit output type	var
Maximum parameters	10
Minimum parameters	2

It takes these parameters:

#	Description	Data type	Required
0	Numerical value	var	Y
1	Numerical value	var	Y
2..9	Numerical value	var	N

5.14.16.26 Maths.Divide

The function has these properties:

Property	Value
Display name	Maths.Divide
Script name	MTH-DIVIDE
Description	Divides one numerical value by another and returns the result.
Category	Mathematical functions
Implicit output type	var
Maximum parameters	2
Minimum parameters	2

It takes these parameters:

#	Description	Data type	Required
0	Numerical value	var	Y
1	Numerical value	var	Y

5.14.16.27 Maths.Multiply

The function has these properties:

Property	Value
Display name	Maths.Multiply
Script name	MTH-MULTIPLY
Description	Multiplies one numerical value by another and returns the result.
Category	Mathematical functions
Implicit output type	var
Maximum parameters	2
Minimum parameters	2

It takes these parameters:

#	Description	Data type	Required
0	Numerical value	var	Y
1	Numerical value	var	Y

5.14.16.28 Maths.Modulus

The function has these properties:

Property	Value
Display name	Maths.Modulus
Script name	MTH-MODULUS
Description	Divides one numerical value by another and returns the remainder.
Category	Mathematical functions
Implicit output type	var
Maximum parameters	2
Minimum parameters	2

It takes these parameters:

#	Description	Data type	Required
0	Numerical value	var	Y
1	Numerical value	var	Y

5.14.16.29 Maths.Max

The function has these properties:

Property	Value
Display name	Maths.Max
Script name	MTH-MAX
Description	Returns the largest of two or more numerical values
Category	Mathematical functions
Implicit output type	var
Maximum parameters	10
Minimum parameters	2

It takes these parameters:

#	Description	Data type	Required
0	Numerical value	var	Y
1	Numerical value	var	Y
2..9	Numerical value	var	N

5.14.16.30 Maths.Min

The function has these properties:

Property	Value
Display name	Maths.Min
Script name	MTH-MIN
Description	Returns the smallest of two or more numerical values
Category	Mathematical functions
Implicit output type	var
Maximum parameters	10
Minimum parameters	2

It takes these parameters:

#	Description	Data type	Required
0	Numerical value	var	Y
1	Numerical value	var	Y
2..9	Numerical value	var	N

5.14.16.31 Maths.Round

The function has these properties:

Property	Value

Display name	Maths.Round
Script name	MTH-ROUND
Description	Performs rounding on a decimal value. By default, the value is rounded to the nearest integer. Alternatively a second parameter can be specified giving the number of significant digits required
Category	Mathematical functions
Implicit output type	var
Maximum parameters	2
Minimum parameters	1

It takes these parameters:

#	Description	Data type	Required
0	Value to be rounded	var	Y
1	Significant digits required	var	N

5.14.16.32 Maths.Sum

The function has these properties:

Property	Value
Display name	Maths.Sum
Script name	MTH-SUM
Description	Returns the sum of the values of all occurrences of a node in the source message
Category	Mathematical functions
Implicit output type	var
Maximum parameters	1
Minimum parameters	1
Notes	Only a source node may be specified as an input to this function

It takes these parameters:

#	Description	Data type	Required
0	Source node	-	Y

5.14.16.33 Maths.Count

The function has these properties:

Property	Value
Display name	Maths.Count
Script name	MTH-COUNT

Description	Returns the number of occurrences of a node in the source message
Category	Mathematical functions
Implicit output type	var
Maximum parameters	1
Minimum parameters	1
Notes	Only a source node may be specified as an input to this function

It takes these parameters:

#	Description	Data type	Required
0	Source node	-	Y

5.14.16.34 Maths.Average

The function has these properties:

Property	Value
Display name	Maths.Average
Script name	MTH-AVERAGE
Description	Returns the arithmetic mean of the values of all occurrences of a node in the source message
Category	Mathematical functions
Implicit output type	var
Maximum parameters	1
Minimum parameters	1
Notes	Only a source node may be specified as an input to this function

It takes these parameters:

#	Description	Data type	Required
0	Source node	-	Y

5.14.16.35 DateTime.Convert

The function has these properties:

Property	Value
Display name	DateTime.Convert
Script name	DT-CONVERT
Description	Converts a date-time string on one format to a date-time string in another format
Category	Date and time functions

Implicit output type	System.String
Maximum parameters	1
Minimum parameters	1
Notes	The following must be specified in the date-time function properties dialog: date-time format of source string; date-time format of target string

It takes these parameters:

#	Description	Data type	Required
0	Date-time string	System.String	Y

5.14.16.36 DateTime.Add

The function has these properties:

Property	Value
Display name	DateTime.Add
Script name	DT-ADD
Description	Adds an interval to a date-time value and returns a new date-time string in a specified format
Category	Date and time functions
Implicit output type	System.String
Maximum parameters	2
Minimum parameters	2
Notes	The following must be specified in the date-time function properties dialog: date-time format of source string; date-time format of target string; string format of interval

It takes these parameters:

#	Description	Data type	Required
0	Date-time string	System.String	Y
1	Interval to add	System.String	Y

5.14.16.37 DateTime.Subtract

The function has these properties:

Property	Value
Display name	DateTime.Subtract
Script name	DT-SUBTRACT
Description	Subtracts an interval from a date-time value and returns a new date-time string in a specified format

Category	Date and time functions
Implicit output type	System.String
Maximum parameters	2
Minimum parameters	2
Notes	The following must be specified in the date-time function properties dialog: date-time format of source string; date-time format of target string; string format of interval

It takes these parameters:

#	Description	Data type	Required
0	Date-time string	System.String	Y
1	Interval to subtract	System.String	Y

5.14.16.38 DateTime.Week

The function has these properties:

Property	Value
Display name	DateTime.Week
Script name	DT-WEEK
Description	Converts a date-time string in a given format to a week number
Category	Date and time functions
Implicit output type	System.Int32
Maximum parameters	1
Minimum parameters	1
Notes	The following must be specified in the date-time function properties dialog: date-time format of source string; week start rule; week start day

It takes these parameters:

#	Description	Data type	Required
0	Date-time string	System.String	Y

5.14.16.39 DateTime.WeekToDate

The function has these properties:

Property	Value
Display name	DateTime.WeekToDate
Script name	DT-WEEKDATE
Description	Converts a week number to a date. The input provided is a string in the format YYWW or

	YYYYWW. The week and year are extracted and used to calculate the date of a given day in that week number of that year.
Category	Date and time functions
Implicit output type	System.String
Maximum parameters	1
Minimum parameters	1
Notes	The following must be specified in the date-time function properties dialog: week-year format of source string; date-time format of target string; week start rule; week start day; day of week required

It takes these parameters:

#	Description	Data type	Required
0	Date-time string	System.String	Y

5.14.16.40 DateTime.MonthEnd

The function has these properties:

Property	Value
Display name	DateTime.MonthEnd
Script name	DT-MONTHEND
Description	Converts a date-time string into another date-time string representing the last day of the month in which the first falls
Category	Date and time functions
Implicit output type	System.String
Maximum parameters	1
Minimum parameters	1
Notes	The following must be specified in the date-time function properties dialog: date-time format of source string; date-time format of target string

It takes these parameters:

#	Description	Data type	Required
0	Date-time string	System.String	Y

5.14.16.41 Function.Source

The function has these properties:

Property	Value
Display name	Function.Source

Script name	ADV-SOURCE
Description	Allows any source of data to be displayed and used in a map; for instance, this function could be used to show a map connection to a target node where the source was a constant or literal value, which in turn would permit the map connection to be conditioned by the result of a logic function. A node from the source tree view may optionally be specified as a second parameter; the node is then used as a trigger, but its value is ignored; supply this parameter when code cannot be generated without it
Category	Advanced functions
Implicit output type	System.String
Maximum parameters	2
Minimum parameters	1

It takes these parameters:

#	Description	Data type	Required
0	Source value	System.String	Y
1	Trigger (value ignored)	-	N

5.14.16.42 Db.Connect

The function has these properties:

Property	Value
Display name	Db.Connect
Script name	DB-CONNECT
Description	Defines a connection to a database for use with Db.Query
Category	Advanced functions
Implicit output type	N/A
Maximum parameters	0
Minimum parameters	0

It takes no parameters

5.14.16.43 Db.Query

The function has these properties:

Property	Value
Display name	Db.Query
Script name	DB-QUERY
Description	Defines a query to be performed against a database
Category	Advanced functions

Implicit output type	System.String for SELECT queries & NULL for other query types
Maximum parameters	1 from Db.Connect & 10 user parameters
Minimum parameters	1 from Db.Connect & 0 user parameters

It takes these parameters:

#	Description	Data type	Required
-	Database connection	Db.Connect	Y
0..9	User parameters, the values of which are referenced in the query text using placeholders %P1%, %P2% &c.	System.String	N

5.15 Code snippet objects

The following sections describe objects that are available in code snippets by default, and which provide specific services to the user.

5.15.1 Static variables

Variables declared in code (whether global or local) can only be relied upon to keep their values within the scope of a single map. To store values that can be used across more than one map, an object of the class `Static` (named `STATIC`), is provided for use in code snippets. You could, for instance, initialise a static variable named `count` in the global INITMAP section, as follows:

```
STATIC.SetInt32("count", 0);
```

This line of code associates the value zero with an Int32 given by the variable name `count` (note that the `STATIC` object cannot be addressed in the `INITIALISE` statement). It can later be accessed and manipulated, for example in a PRE code snippet to be performed before mapping from a source node to a target node:

```
int iCount = STATIC.GetInt32("count");
iCount++;
STATIC.SetInt32("count", iCount);
```

The variable name `count` does not refer to a code snippet variable, it has meaning only as an argument to `Get` and `Set` functions on the `STATIC` object.

5.15.1.1 Get and Set Functions

The `STATIC` object has a number of functions for getting and setting variable values. There are `Get` and `Set` functions for the `.NET` built-in value types, and for `System.String` and `System.Object`, as listed below.

```
GetObject
GetString
GetDateTime
GetBool
GetChar
GetByte
GetInt16
GetInt32
GetInt64
GetFloat
GetDouble
```

```
SetObject
SetString
SetDateTime
SetBool
SetChar
SetByte
SetInt16
SetInt32
SetInt64
SetFloat
SetDouble
```

The `STATIC` object's functions share the following behaviour:

- If a `Get` function is called and no value has previously been set for the given variable name, then a default value is set and returned (the defaults are: zero for numeric types, false for Booleans, empty strings, null objects). It follows that static variables do not need to be 'declared' or initialised before use.
- If a `Get` function is called and the type of the stored value does not match the function call, then a default value is returned (e.g. call SetString("str", "abcdef"), then GetInt32("str") will return zero; these functions do not throw exceptions)
- The `Set` functions automatically overwrite existing values for a given variable name, even if the types differ.

5.15.2 Logging

A static class named `XE` provides the ability to write messages to the Xe log. For example, when mapping from a source node to a target node you can write an entry (of fixed text) to the log, using a PRE code snippet:

```
XE.Log("ABC0000I", "Map ABC010 to DEF020");
```

This would result in an entry in the Xe log like this:

```
MAP1004I    Resolved map DLL to c:\aaa\map.dll
ABC0000I    Map ABC010 to DEF020
BAS2501T    StreamWriter: Open file, 'c:\aaa\out.edi'
```

There are three variations on the log function: `Log()`, `Trace()` and `DeepTrace()`. Each takes two string parameters, 'code' and 'message'. The `Log` function always writes messages to the log (assuming logging enabled), the `Trace` function only writes messages if trace-level logging is enabled, the `DeepTrace` function only writes messages if deep-trace-level logging is enabled.

5.15.3 Source

When writing a local (i.e. not global) code snippet, the value `SRC` can be used to access the value of the source node, avoiding the need to use local code variables and `SETVAL` parameters to get values into and out of the code snippet. For example:

```
if (SRC == "GHI") SRC="JKL";
```

The code snippet has the effect of changing the value that is mapped (to the target node) to `JKL`, only if it was originally `GHI`. The value `SRC` is a string variable. It is only available in `PRE` or `POST` code where a source node is used as the source type.

5.15.4 Get values

The `XE` object also has a number of `Get___()` functions that can be used to fetch a value from the source file, using a path expression, directly from the code snippet. For example, in a PRE code snippet to be performed before mapping from a source node to a target node:

```
SRC += XE.GetString("./PARENT/CHILD[#REF=HHH030]/#VALUE"));
```

The path expression may be relative or absolute. If relative, the starting context node is the same as it would be if the path expression were used on an `IF` parameter on the `MAP` statement. The returned type depends upon the `Get` function used.

In each case the value retrieved from the source DOM (which is a string) is converted to the type given by the function call. If the value cannot be converted the default behaviour is to log the error and continue. This can be changed by using an overloaded version of each function that allows error behaviour to be specified; the integer parameter `onerror` may have one of these values:

`ERROR_LOG`	Return default value, log and continue
`ERROR_SKIP`	Return default value, continue without logging
`ERROR_STOP`	Log error and stop processing.

So, for instance, to stop mapping when an error occurs use the following PRE code snippet:

```
SRC += XE.GetString("./PARENT/CHILD[#REF=HHH030]/#VALUE"),
          ERROR_STOP);
```

The functions available are as follows:

GetBool
```
public static System.Boolean GetBool(string path, int onerror)
public static System.Boolean GetBool(string path)
```

GetByte
```
public static System.Byte GetByte(string path, int onerror)
public static System.Byte GetByte(string path)
```

GetChar
```
public static System.Char GetChar(string path, int onerror)
public static System.Char GetChar(string path)
```

GetInt16
```
public static System.Int16 GetInt16(string path, int onerror)
public static System.Int16 GetInt16(string path)
```

GetInt32
```
public static System.Int32 GetInt32(string path, int onerror)
public static System.Int32 GetInt32(string path)
```

GetInt64
```
public static System.Int64 GetInt64(string path, int onerror)
public static System.Int64 GetInt64(string path)
```

GetString
```
public static System.String GetString(string path, int onerror)
public static System.String GetString(string path)
```

GetSingle
```
public static System.Single GetSingle(string path, int onerror)
public static System.Single GetSingle(string path)
```

GetDouble

```
public static System.Double GetDouble(string path, int onerror)
public static System.Double GetDouble(string path)
```

GetDateTime

```
public static System.DateTime GetDateTime(string path, string format,
    int onerror)
public static System.DateTime GetDateTime(string path,
    string format)
```

Fetches a date-time value, in a format given by the format string, and converts it to a System.DateTime struct. Any valid .NET format string may be used.

GetDateTimeHHMM

```
public static System.DateTime GetDateTimeHHMM(string path,
    int onerror)
public static System.DateTime GetDateTimeHHMM(string path)
```

Fetches a time value in the format HHMM (equivalent to the format string ‘HHmm’) and converts to a System.DateTime struct.

GetDateTimeHHMMSS

```
public static System.DateTime GetDateTimeHHMMSS(string path,
    int onerror)
public static System.DateTime GetDateTimeHHMMSS(string path)
```

Fetches a time value in the format HHMMSS (equivalent to the format string ‘HHmmss’) and converts to a System.DateTime struct.

GetDateTimeYYMMDD

```
public static System.DateTime GetDateTimeYYMMDD(string path,
    int onerror)
public static System.DateTime GetDateTimeYYMMDD(string path)
```

Fetches a date value in the format YYMMDD (equivalent to the format string ‘yyMMdd’) and converts to a System.DateTime struct.

GetDateTimeCCYYMMDD

```
public static System.DateTime GetDateTimeCCYYMMDD(string path,
    int onerror)
public static System.DateTime GetDateTimeCCYYMMDD(string path)
```

Fetches a date value in the format CCYYMMDD (equivalent to the format string ‘yyyyMMdd’) and converts to a System.DateTime struct.

GetDateTimeYYMMDDHHMM

```
public static System.DateTime GetDateTimeYYMMDDHHMM(string path,
    int onerror)
public static System.DateTime GetDateTimeYYMMDDHHMM(string path)
```

Fetches a date-time value in the format YYMMDDHHMM (equivalent to the format string ‘yyMMddHHmm’) and converts to a System.DateTime struct.

GetDateTimeYYMMDDHHMMSS

```
public static System.DateTime GetDateTimeYYMMDDHHMMSS(string path,
    int onerror)
public static System.DateTime GetDateTimeYYMMDDHHMMSS(string path)
```

Fetches a date-time value in the format YYMMDDHHMMSS (equivalent to the format string ‘yyMMddHHmmss’) and converts to a System.DateTime struct.

GetDateTimeCCYYMMDDHHMM

```
public static System.DateTime GetDateTimeCCYYMMDDHHMM(string path,
    int onerror)
public static System.DateTime GetDateTimeCCYYMMDDHHMM(string path)
```

Fetches a date-time value in the format CCYYMMDDHHMM (equivalent to the format string ‘yyyyMMddHHmm’) and converts to a System.DateTime struct.

GetDateTimeCCYYMMDDHHMMSS

```
public static System.DateTime GetDateTimeCCYYMMDDHHMMSS(string path,
    int onerror)
```

```
public static System.DateTime GetDateTimeCCYYMMDDHHMMSS(string path)
```

Fetches a date-time value in the format CCYYMMDDHHMMSS (equivalent to the format string 'yyyyMMddHHmmss') and converts to a System.DateTime struct.

5.15.5 Service segment values

Service segments cannot be addressed using path expressions. Instead unique IDs or references are used to identify each element value. The following method of the XE object can be used in a code snippet to return a value from a service segment (the references that can be used are listed in the section of this document that describes EDI definition fields in the index).

```
public static string GetServSegValue(string ref)
```

So, for instance, to fetch the sender ID from the UNB segment and store in a string:

```
string str = XE.GetString("UNB-SENDER-ID");
```

5.15.6 Counter values

The XE object has the following functions to get at and set values of counters defined in the Xe Index:

```
public static int GetCounterValue(string id)
public static void SetCounterValue(string id, int count)
public static void IncrementCounterValue(string id)
public static string GetMaskedCounterValue(string id)
```

In each case, the id parameter is the ID of the counter as defined by you in a counter entity in the Xe Index. You can define any number of counters in the index and use each one for a different purpose if desired. These counters can be accessed via the Mapper GUI using the Xe class.

The functions work as follows:

GetCounterValue

Returns the current value of the counter; an integer.

SetCounterValue

Sets the current value of the counter; an integer.

IncrementCounterValue

Increments the current value of the counter; the amount of the increment is that defined against the counter in the index, defaulting to 1 if none is specified.

GetMaskedCounterValue

Returns the current value of the counter as a string with a mask applied. For instance, if the mask defined in the index is ABCD%cnt:5% and the value of the counter is 777, the function will return "ABCD00777".

N.B. If you want to use a counter to maintain a unique Interchange Control Reference for each of your trading partners, you would need to provide a stream entity for each trading partner which, in turn, refers to a unique counter entity you have provided. Having provided the required stream attributes and properties, the index will automatically populate the appropriate EDI field with the counter value. You cannot access in the Mapper GUI counters that are referred to in streams.

5.15.7 Lookups

The XE object has a Lookup() function that can be used to convert a value using a lookup table defined in the index:

```
public static string Lookup (string tbl, bool rtol, string val)
```

The parameters are:

- A table reference – this is the reference assigned to the table in the index.
- A flag indicating the lookup direction – use false to lookup left-to-right and true to lookup right-to-left.
- The value to be looked up.

The function returns the converted value (a string).

5.15.8 Result

The mapper maintains a flag that indicates whether a map was completed successfully or not (the System.Boolean value 'OK'). The flag is available to you in your ENDMAP code snippet. To check the map outcome code this:

```
if (OK == true)
   // Map succeeded - do something
else
   // Map failed - do something else
```

5.15.9 Exit

The XE object has an Exit() function that can be used to exit a map immediately. The function is called with one of these codes:

```
EXIT_OK          Map succeeded, normal termination
EXIT_FAIL        Map failed, terminates in error
EXIT_SKIP        Does not produce a target document,
                 but does not terminate in error
```

Note that the 'skip' option is not compatible with:

- The index mechanism to flush part of a target document during a map.
- The 'single target document' flag in the index.

5.15.10 New target

The XE object has a NewTarget() function that can be used to force the creation of a new XML document during a map. The default situation is that one map creates one XML target document. To create a new XML document at any point during the map, call this function.

```
if (test == true)          // Create new target if Boolean is set
  XE.NewTarget();
```

5.15.11 XML document properties

The XE object has functions that can be used to map specific properties of a target XML document. The SetXmlEncoding() function is used to specify the encoding that will be written to the prolog of the target document. The SetXmlDoctype() function is used to map a literal !DOCTYPE node to the target. The SetXmlVersion() function is used to specify the version number to include in the prolog:

```
XE.SetXmlEncoding("ISO-8859-1");
XE.SetXmlDoctype("<!DOCTYPE message SYSTEM \"sample.dtd\">");
XE.SetXmlVersion("1.1");
```

5.15.12 Map comments

The `XE` object has a `MapComment ()` function that can be used to insert a comment into an XML target document either before or after a specified element:

```
void MapComment(string path, string value, bool before);
```

The path is the full path of the target element with which the comment is to be associated. The value is the literal value of the comment. The final Boolean parameter indicates whether the comment is written before or after the element given by the path. So a code snippet of:

```
XE.MapComment("//Invoice/ID", "some text", false);
```

Might produce XML output of:

```
<Invoice>
  <ID>12345</ID>
  <!-- some text -->
  <More>
    ...
```

The full path needs to be qualified if the element name in the target XML is qualified. So:

```
XE.MapComment("//Invoice/dip:ID", "some text", false);
```

Will produce:

```
<Invoice>
  <dip:ID>12345</dip:ID>
  <!-- some text -->
  <More>
    ...
```

Note that the comment will be associated with the last instance of the specified element to be mapped to the target, so the comment cannot be mapped before the element is. Typically this means that the POST code snippet of the map connection that creates the element will be used to map the comment.

5.15.13 Get source nodes

The `XE` object has functions to support fetching one or more nodes from the source document, given a path expression. These are used when more complex processing is required than can be carried out using the `Get___()` functions to return single values and include the ability to loop over a collection of source nodes.

GetNode

The following functions are available when working with multiple nodes:

```
public static HANDLE GetNodeList(string path)
public static HANDLE GetNodeList(NODE node, string path)
public static HANDLE GetNodeList(HANDLE handle, string path)

public static NODE GetNextNode(HANDLE handle)
public static NODE GetNode(HANDLE handle, int pos)

public static int GetNodeCount(HANDLE handle)
public static void ReleaseNodeList(HANDLE handle)
```

Call `GetNodeList()` to create a list of source nodes given a relative or absolute path expression. The type returned is a `HANDLE`, which is used by Xe to reference the list and which is passed as a parameter to the other functions. There are

three overloads of this function. The first assumes that the context node for relative path expressions is the current context node in the mapper. The second allows a NODE object representing another DOM node to be supplied, in which case that node will be the context node for the path expression. The third allows a handle to another node list to be supplied, in which case the context node for the path expression will be the current node in the node list represented by the handle.

Once a node list has been initialised, nodes can be accessed using the GetNextNode() function (strictly read-only forward iteration) which takes the handle to the node list as a parameter and returns an object of the class NODE (described below). Alternatively, an overload of the GetNode() function can be used to obtain a single node from the list given a handle to the list and a (zero-based) index position. GetNextNode() advances the iterator associated with the list handle and so affects the context node in any nested call to GetNodeList() that uses the handle. Calling GetNode() does not advance the iterator and therefore does not affect the current context node.

A count of the total number of nodes in the list can be obtained at any time using the GetNodeCount() function. When the node list is no longer needed, call the ReleaseNodeList() function to free the memory associated with it (if this function is not called, the list will be released automatically when the map finishes).

GetNode

The following functions are used to return a single node, given a relative or absolute path expression:

```
public static NODE GetNode(string path)
public static NODE GetNode(NODE node, string path)
public static NODE GetNode(HANDLE handle, string path)
```

There are three overloads of this function. The first assumes that the context node for relative path expressions is the current context node in the mapper. . The second allows a NODE object representing another DOM node to be supplied, in which case that node will be the context node for the path expression. The third allows a handle to another node list to be supplied, in which case the context node for the path expression will be the current node current node in the node list represented by the handle. The type returned is an object of the class NODE (described below).

GetNodeValue

The following functions are used to return a value from a single node, given a relative or absolute path expression:

```
public static string GetNodeValue (NODE node, string path)
public static string GetNodeValue(HANDLE handle, string path)
```

There are two overloads of this function. In the first an object of the type NODE is used to indicate the context node for the path expression. In the second a HANDLE to a node list is used to indicate the context node.

NODE class

The functions described above return objects of the class NODE to represent source nodes. The class has three properties that expose data about the source node:

```
public string Name  { get; }
public string Ref   { get; }
public string Value { get; }
```

These return the node name, node reference and data value respectively.

Example

The following code example indicates how these functions are used:

```
// Get children of the root node named "ABC"
HANDLE hLoop = XE.GetNodeList("//CHILD[#NAME=ABC]");
NODE node1 = XE.GetNextNode(hLoop);

// Loop over "ABC" nodes
while(node1 != null)
{
    // Do something with "ABC" node (node1)

    // Get children of the "ABC" node with the reference "DEF0010"
    HANDLE hLoop2 = XE.GetNodeList(hLoop, "./CHILD[#REF=DEF0010]");
    NODE node2 = XE.GetNextNode(hLoop2);

    while(node2 != null)
    {
        // Do something with "DEF0010" node (node2)

        // Get a child of the "DEF0010" node named "777"
        NODE node3 = XE.GetNode(hLoop2, "./CHILD[#NAME=777]");

        // Do something with "777" node (node3)

        // Get next "DEF0010" node
        node2 = XE.GetNextNode(hLoop2);
    }

    // Get the third node in the loop over the "DEF0010" nodes
    NODE node4 = XE.GetNode(hLoop2, 3);

    // Do something with the third "DEF0010" node (node4)

    // Get next "ABC" node
    node1 = XE.GetNextNode(hLoop);
}
```

5.15.14 Parameters

The XE class has functions to support accessing values of external parameters specified when the map is executed. These functions are implemented:

GetParameter

```
public static string GetParameter(string name);
```

Returns the value associated with a named parameter. If the parameter was not supplied when the map was invoked then the parameter name is returned.

ReplaceParameters

```
public static string ReplaceParameters(string str);
```

Returns the source string with any parameter names it contains replaced by their defined values. For instance, if the string passed in is “Rupert is %ABC%” and the value of parameter %ABC% is given at runtime as “a bear”, the string returned from the function will be “Rupert is a bear”. If the value of %ABC% is given as “an evil media baron” then a different string will be returned.

5.15.15 ODEX

A static class named `ODEX` provides the ability to communicate with ODEX Enterprise, if you are running the map as part of an ODEX workflow. These functions are supported:

GetWorkflowFileUserData

```
public static string GetWorkflowFileUserData();
public static string GetWorkflowFileUserData(int onerror);
```

Returns the user data field associated with the workflow file in ODEX. An overload allows you to indicate how to handle errors; it takes one of the constants `ERROR_LOG`, `ERROR_SKIP`, and `ERROR_STOP` defined above.

SetWorkflowFileUserData

```
public static void SetWorkflowFileUserData(string strData);
public static void SetWorkflowFileUserData(string strData, int onerror);
```

Sets the value of the user data field associated with the workflow file in ODEX. An overload allows you to indicate how to handle errors; it takes one of the constants `ERROR_LOG`, `ERROR_SKIP`, and `ERROR_STOP` defined above.

6 Definition Editor

6.1 Introduction

Xe now includes a definition editor for creating and manipulating the definition files used in the mapper. The file types supported are:

- **EDF (EDI Definition File)** – For defining EDI messages.
- **HDF (In-house Definition File)** – For defining flat files, CSV, IDOC and TRA documents.
- **XDF (XML Definition File)** – For defining XML documents.

The definition editor can be used to create definitions from scratch, or to amend or augment existing definitions. You are most likely to have to create an HDF from scratch. This is because EDFs can usually be generated from the Data Dictionary or adapted from existing EDFs, and XDFs can always be generated from XML instance documents. The definition editor has been designed to support rapid creation of definition files, especially HDFs, where records and fields can be added with single mouse clicks or key presses.

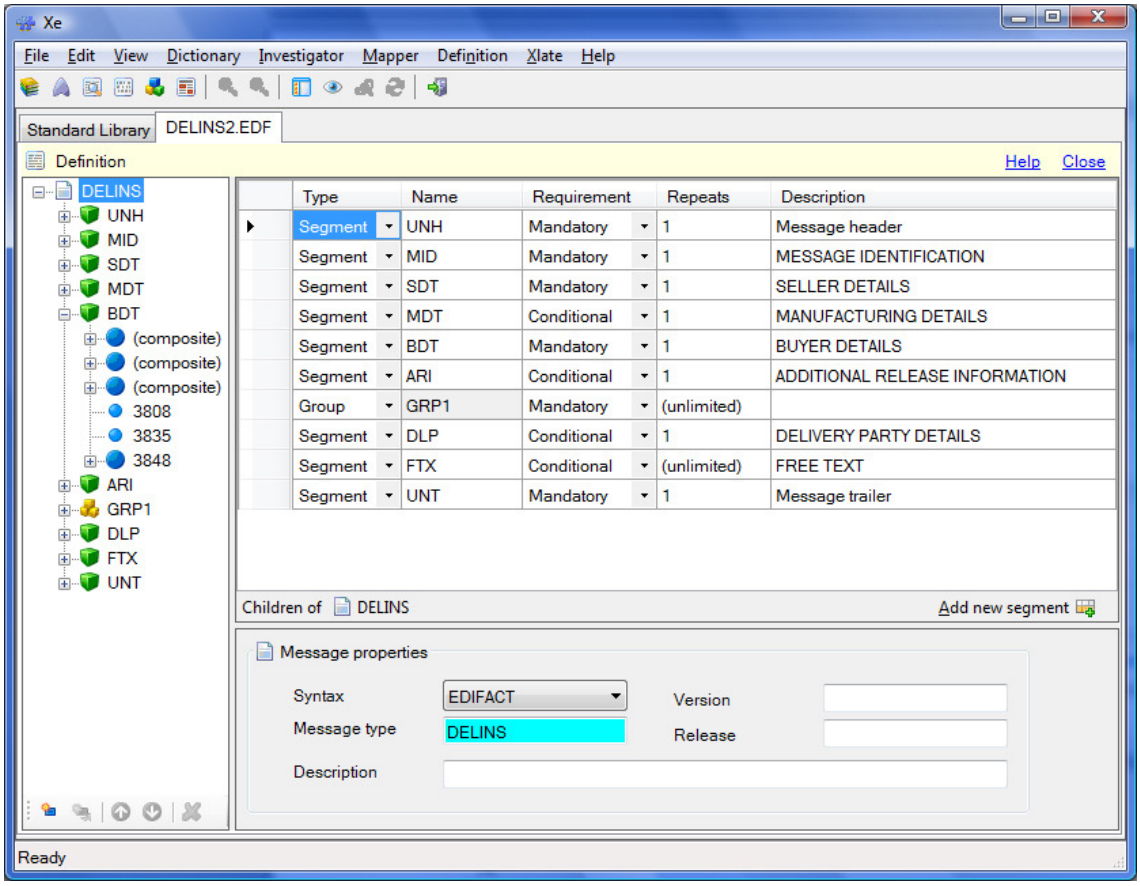
6.1.1 Definition files and Xlate

If you are familiar with Xlate (the predecessor of the Xe mapping engine) you will know that the EDF and HDF file formats used in Xe come from Xlate. These definitions have been extended in some respected to support the newer features of the Xe mapper not available in Xlate. There are also a number of features of the definition files required by Xlate that are not needed in Xe.

The definition editor supports only those features required by Xe. While it is possible to load in a definition including Xlate-only features, those features will not be written out when the definition file is saved. If you load a file that contains such featured you will see a warning message to this effect. If you want to preserve those features, you should close the definition editor without saving the file.

6.2 The definition editor

This section outlines the main features of the definition editor. The image below shows the editor GUI. It is shown with an EDI definition in it, but the principles described are of general application:



The three main parts of the editor are as follows:

6.2.1 Definition hierarchy

The left-hand part of the editor is occupied by a tree view that contains the full hierarchy of the message or document. When you select an item in the tree its child entities are displayed in the grid to the right (see Child items) and the properties of the item are displayed in the panel below (see Item properties).

The **Definition** menu item contains options for how items are displayed in the tree. Check the **Show descriptions** menu item to include descriptions of each entity in the tree. Check the **Single expand** menu item restrict the tree to expanding only one node at any level.

Actions

As well as using the tree view to display and navigate the hierarchy, you can use it to manipulate the definition. Right-clicking an item in the tree will cause a pop-up menu to be displayed. Depending on the item selected, the following options are available:

- **Add** – Add a child entity. Depending on the item selected in the tree, multiple Add items may be included on the menu, one for each permitted child type. New entities are appended as the last child of the selected item.
- **Insert** – Insert a sibling entity. Depending on the item selected in the tree, multiple Insert items may be included on the menu, one for each permitted sibling type. New entities are inserted as the next sibling of the selected item.
- **Move up** – The selected item is moved up in the tree view.
- **Move down** – The selected item is moved down in the tree view.

- **Cut** – Add the selected item (and any descendants) to the clipboard and remove from the tree view
- **Copy** – Add the selected item (and any descendants) to the clipboard and leave in the tree view
- **Paste as child** – If there is a suitable item on the clipboard, append it as the last child of the selected item.
- **Paste as sibling** – If there is a suitable item on the clipboard, insert it as the next sibling of the selected item.
- **Delete** – Remove the selected item (and any descendants) from the tree view.

Some of these options are also available from the small toolbar at the bottom of the tree view. These are: Add (default child item), Insert (default child item), Move up, Move down, Delete. In each case, the button will be enabled only when the associated action is available for the currently selected tree view item.

6.2.2 Child items

To the right of the tree view (at the top of the screen) is a grid containing rows representing the immediate child entities of whatever is selected in the tree view. Each row contains information about one child of the selected item. The columns shown vary according to the type of entity selected in the tree. The status bar at the bottom of the grid includes text indicating what child entities are being shown (in the example above, the text “Children of DELINS” indicates that it is the children of the message definition itself that are being displayed).

The cells of the grid, which exist at the intersection of row and column each contain one property of one entity. In most cases the first column (and therefore the first cell in each row) represents the entity type (for instance, if the tree view selection is an EDI Segment, the permitted child entity types are Data Element and Composite).

In most cases, cell values can be edited *in situ*. Cells are edited by either entering a text value, of choosing a value from a drop-down list. Some properties are read-only and these are shown in cells with a greyed-out background. Also, editing of the entity type cell is disabled if changes are made that prevent the required type conversion. For instance, when you first create a child entity of an EDI Segment you can freely switch its type between Element and Composite, but once the Composite has child Elements its type cannot be changed.

New rows representing new entitles can be added to the grid using the Add button on the status bar at the bottom of the grid. The entity type added will be the default child entity for the parent item (the default child of an EDI Message is a Segment, the default child of a Segment is an Element). Items can also be added by pressing the **Insert** key when the grid is focussed.

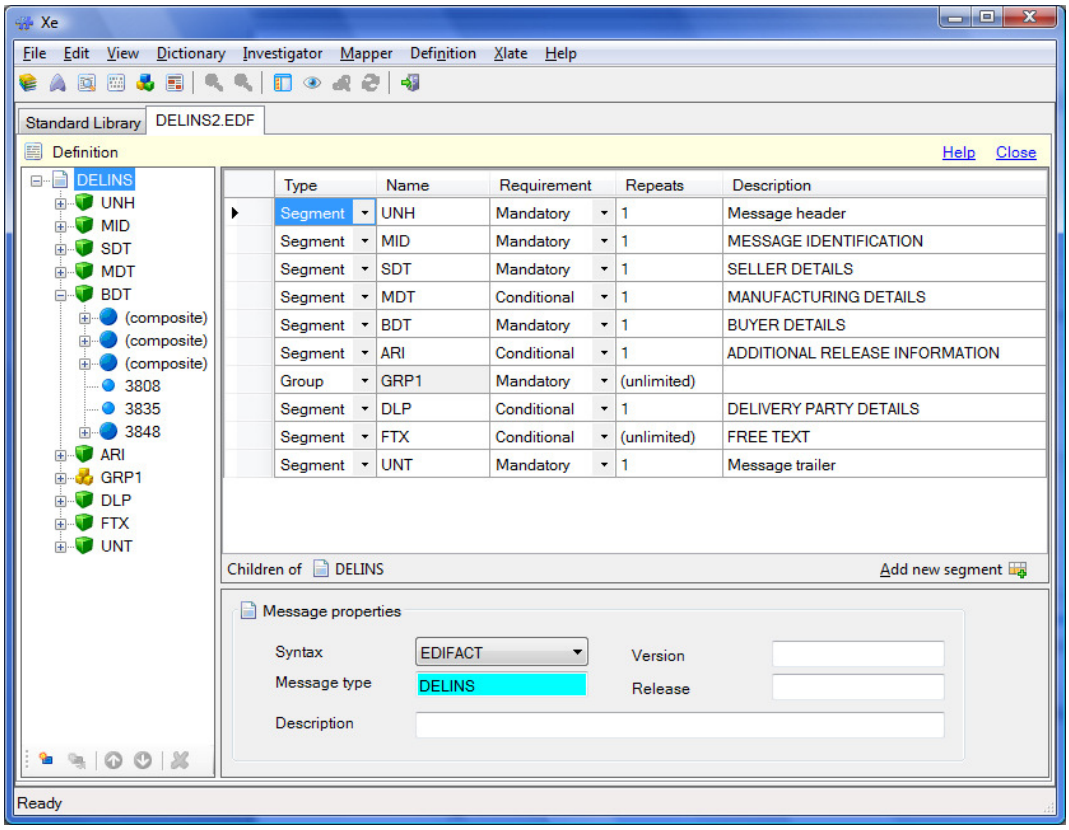
6.2.3 Item properties

To the right of the tree view (at the bottom of the screen) is a panel showing the properties of whatever was last selected in the tree view or grid. The panel changes according to the entity type that is selected for editing. All of the editable properties of the entity are shown in the panel, including some that are

also shown in the grid, and can be edited directly. Details of what appears in the panel for each entity type are given in later sections.

6.3 Create and edit EDI definitions

Definitions are created and opened in the editor using the Index Explorer. An EDI definition appears in the editor like this:



Note that the message header and trailer service segments are displayed in the tree view according to the selected syntax (in this case UNH and UNT). These are changed if the syntax of the message is changed. They cannot be edited and are not written to the EDF file when the definition is saved. In fact, no service segments are written to the file. This is because the Xe mapper does not use service segment definitions from EDFs when reading EDI documents.

The following sections list the entities and properties shown in the grid view, according to what is selected in the tree, and the properties editable in the panel according to the selection in the tree or grid.

6.3.1 EDI entities in the grid

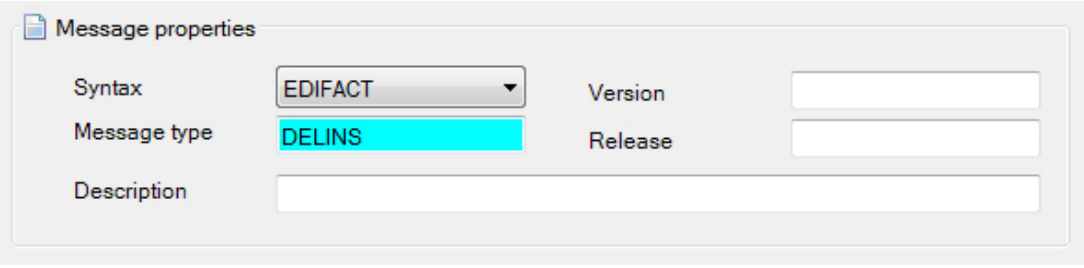
The following table lists the entities and properties (grid columns) shown in the child item grid, according to the element selected in the tree view.

Tree item	Child (grid) items	Grid column	Description
Message, Group	Segment, Group	Type	Entity type drop-down (segment or group)
		Name	Entity name (segment code or group name; group names are read-only)

		Requirement	Whether the entity is mandatory, conditional or optional (within its parent context)
		Repeats	The maximum repeat count for the entity (use 0 or 'unlimited' to indicate unbounded repeats)
		Description	A free text description of the entity (for information only)
Segment	Composite, Element	Type	Entity type drop-down (composite or element)
		Name	Entity name (data tag)
		Requirement	Whether the entity is mandatory, conditional or optional (within its parent context)
		Field format	The expected data format (alphabetic, numeric, alphanumeric &c, for elements only)
		Description	A free text description of the entity (for information only)
Composite	Element	Type	Entity type drop-down (element only)
		Name	Element name (data tag)
		Requirement	Whether the element is mandatory, conditional or optional (within its parent context)
		Field format	The expected data format (alphabetic, numeric, alphanumeric &c)
		Description	A free text description of the element (for information only)
Element	As elements have no children, the grid shows the siblings of the element		

6.3.2 Message properties

The panel shown when the message is selected in the tree view is as follows:

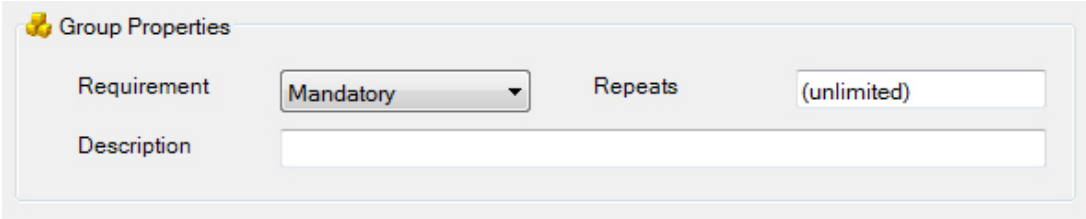


Use the **Syntax** drop-down list to select the message syntax. If the syntax is changed then the read-only message header and trailer service segments in the tree view are switched accordingly. If the syntax is set to VDA then you cannot include composite data elements in the definition as they are not supported in VDA.

Enter the message or transaction set type in the text box. This field is mandatory (though it is not used in the mapping process). You may also enter version and release information and a free-form text description if required.

6.3.3 Group properties

The panel shown when a group is selected in the tree view or grid is as follows:



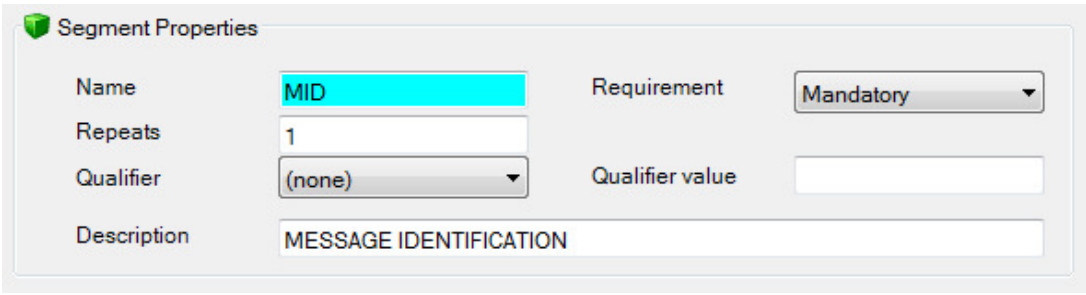
Use the **Requirement** drop-down list to select the group requirement (mandatory, conditional, optional).

Enter the maximum repeat count in the **Repeats** text box (use 0 or 'unlimited' to indicate unbounded repeats are permitted).

You may also enter a free-form text **Description** for the group.

6.3.4 Segment properties

The panel shown when a segment is selected in the tree view or grid is as follows:



Enter the mandatory segment code or tag in the **Name** text box.

Use the **Requirement** drop-down list to select the segment requirement (mandatory, conditional, optional).

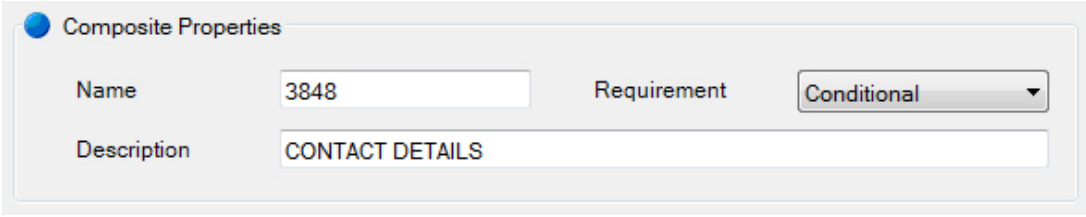
Enter the maximum repeat count in the **Repeats** text box (use 0 or 'unlimited' to indicate unbounded repeats are permitted).

If you want to specify a qualified segment instance then use the **Qualifier** drop-down list to select the element or sub-element that contains the qualifier, and enter the **Qualifier value** in the text box.

You may also enter a free-form text **Description** for the segment.

6.3.5 Composite properties

The panel shown when a composite is selected in the tree view or grid is as follows:



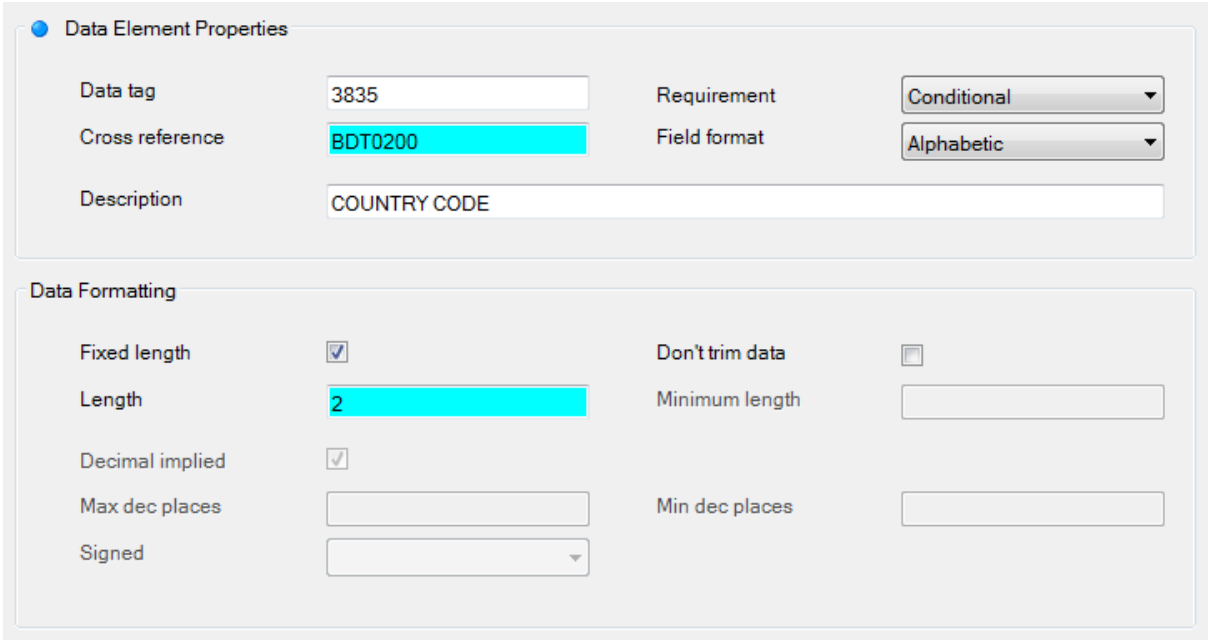
Enter the optional composite data-tag in the **Name** text box.

Use the **Requirement** drop-down list to select the composite requirement (mandatory, conditional, optional).

You may also enter a free-form text **Description** for the segment.

6.3.6 Element properties

The panel shown when an element is selected in the tree view or grid is as follows:



Enter the optional element **Data tag** in the text box.

Use the **Requirement** drop-down list to select the element requirement (mandatory, conditional, optional).

The **Cross reference** field is mandatory and is used in the mapper uniquely to identify a source or target EDI element. When you add elements, cross-references are generated automatically and it should not often be necessary to change them. If you want to re-name all cross-references sequentially within segments, choose the menu command **Definition >> Reset cross-references**.

Use the **Field format** drop-down list to specify the expected data format (alphabetic, numeric, alphanumeric).

You may also enter a free-form text **Description** for the element.

The **Data Formatting** section is used to specify additional information about how data is expected to be formatted when using the definition to load EDI, and how data is to be written out when using the definition to write EDI.

Use the **Fixed length** check box to indicate whether the field is a fixed length or can vary up to the maximum. If the field is fixed length then one **Length** text box is enabled into which you must enter the required field length. The **Don't trim data** flag may be set to indicate that EDI data should not have whitespace trimmed from the end on mapping.

If the field is not fixed length then **Maximum length** and **Minimum length** text boxes are shown. The maximum field length must be entered and must be greater than zero. The minimum field length is optional.

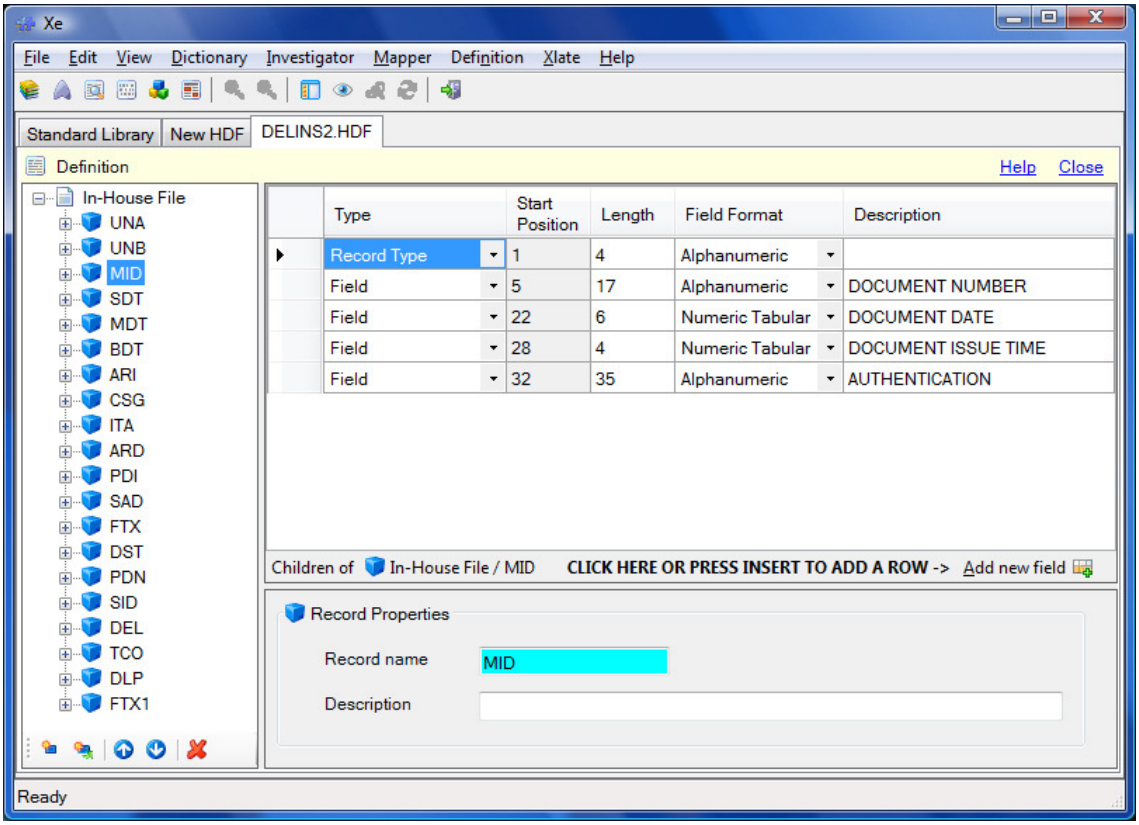
If the field type is numeric then use the **Decimal implied** check box to indicate whether a decimal point is implied or explicit. If the decimal point is implied then

one **Decimal places** text box is enabled into which you may enter the required number of implied decimal places. If the value is zero or missing then no decimal places are implied. If the decimal point is explicit then **Max dec places** and **Min dec places** text boxes are shown. Both values are optional.

If the field type is numeric then use the **Signed** drop-down list to indicate whether the value is signed (leading sign), unsigned or has a trailing sign.

6.4 Create and edit in-house definitions

Definitions are created and opened in the editor using the Index Explorer. An In-house definition appears in the editor like this:



The following sections list the entities and properties shown in the grid view, according to what is selected in the tree, and the properties editable in the panel according to the selection in the tree or grid.

6.4.1 In-house entities in the grid

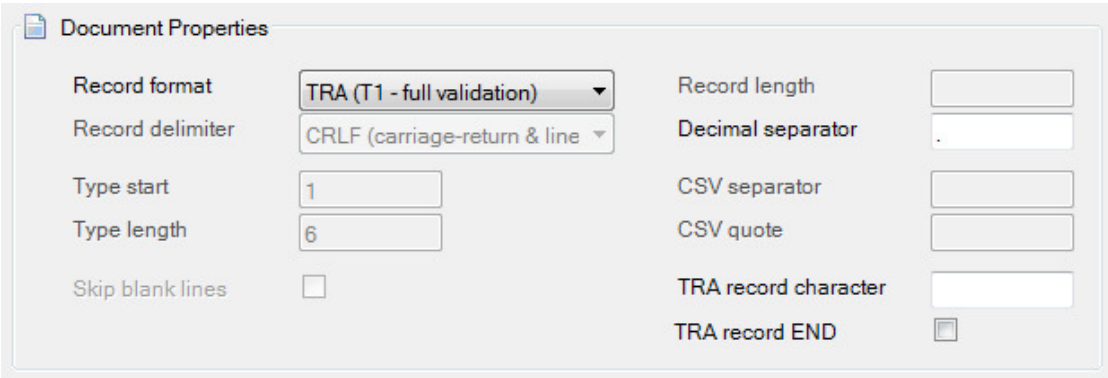
The following table lists the entities and properties (grid columns) shown in the child item grid, according to the element selected in the tree view. Note that the 'section' entity type is relevant only to the TRA file format and cannot be used with other in-house file types.

Tree item	Child (grid) items	Grid column	Description
Document, Group	Record, Group	Type	Entity type drop-down (record or group)
		Record type	Record type name (the value in the record type field, records only)

		Description	A free text description of the entity (for information; records only)
Record (non-TRA formats)	Field	Type	The field type (may be Field for a normal data field, Filler for blank space in the record, or one of the type variants to indicate it is the type field: Type, Typer, Typex, Typef)
		Start position	The start position in the record of the field (read-only)
		Length	The length of the field (or maximum length is the format is CSV)
		Field format	The expected data format (alphabetic, numeric, alphanumeric &c)
		Description	A free text description of the element (for information only)
Record, (TRA format)	Section, Field	Type	Entity type drop-down (section or field)
		Name	Entity name (data tag)
		Length	The maximum length of the field (field only)
		Field format	The expected data format (alphabetic, numeric, alphanumeric &c; field only)
		Description	A free text description of the entity (for information only)
Section, (TRA format)	Field	Type	Entity type drop-down (field only)
		Name	Entity name (data tag)
		Length	The maximum length of the field
		Field format	The expected data format (alphabetic, numeric, alphanumeric &c)
		Description	A free text description of the field (for information only)
Field	As fields have no children, the grid shows the siblings of the field		

6.4.2 Document properties

The panel shown when the document is selected in the tree view is as follows:



Use the **Record format** drop-down list to select the record format. The types available are the same as when creating a new HDF (see Create and edit in-

house definitions). You should take care when changing the format for an HDF with any substantive content as the format specified is fundamental to Xe being able to interpret in-house files.

Where the record format is a delimited type, the **Record delimiter** drop-down can be used to specify the delimiter and the **Skip blank lines** check box can be used to specify that the in-house file is permitted to contain blank lines, which Xe will ignore.

The **Type start** and **Type length** text boxes are used to specify where in a record Xe should look for the record type. The start position is unity-based (so if the record type field is at the start of the record then the start position is 1). The values specified here apply to the whole file. If values are set and the start position is 1 then each time a new record is added a TYPE field will be added too. These values can be overridden for individual records by specifying a TYPE field with a different start position and/or length. Whether the file format is CSV, the TYPE field is identified by field number, so the **Type start** text box becomes **Type field** and the **Type start** text box is disabled. These values are not relevant to TRA formats.

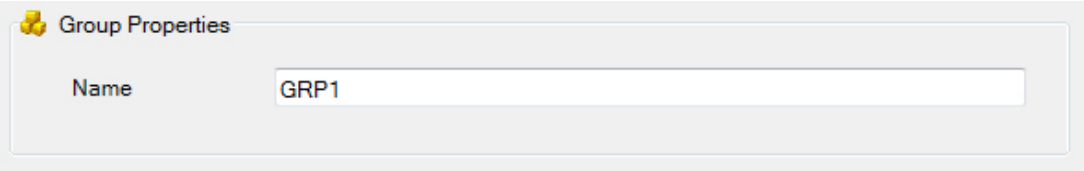
For fixed format files, the **Record length** text box is enabled. Use this to specify the length of each record. The length of each record specified in the definition, calculated by adding up its field lengths, must match the length specified here.

For CSV format files, the **CSV separator** and **CSV quote** text boxes are enabled. Use these to specify the character used as the separator and the character used as the quote in a CSV file.

For TRA format files, the **TRA record character** text box and **TRA record END** check box are enabled. The former is used to specify the specify the character used to indicate the start and end of a TRA file record (the default is '/' as in '/REC-TYPE'). The latter is used to indicate whether a TRA format file uses END record fields (as in '/END').

6.4.3 Group properties

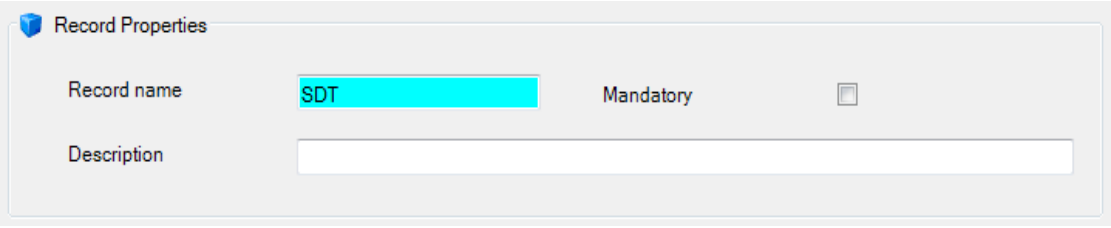
The panel shown when a group is selected in the tree view or grid is as follows:



The **Name** text box shows the group name, which is generated by the editor and is read-only. The HDF group has no editable properties.

6.4.4 Record properties

The panel shown when a record is selected in the tree view or grid is as follows:



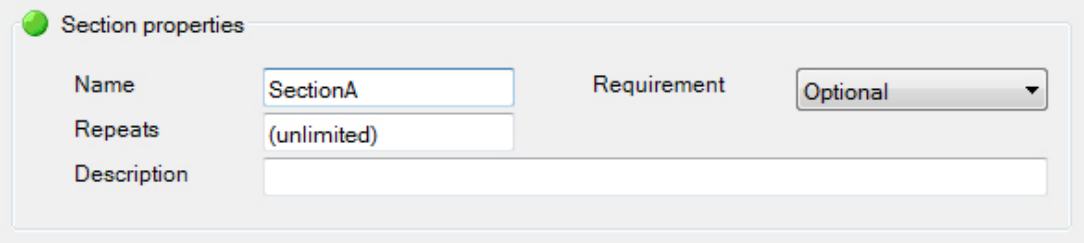
The **Record name** property is the value that appears in the record's TYPE field and is used to match the record in the in-house file with the record definition in the HDF. Record type names must be unique within an in-house definition.

The **Mandatory** check box can be used to indicate that the record is required. If a mandatory item is missing from the source file then a validation error will occur on loading.

Use the **Description** text box to enter a free-form description of the record.

6.4.5 Section properties

The panel shown when a section is selected in the tree view or grid is as follows:



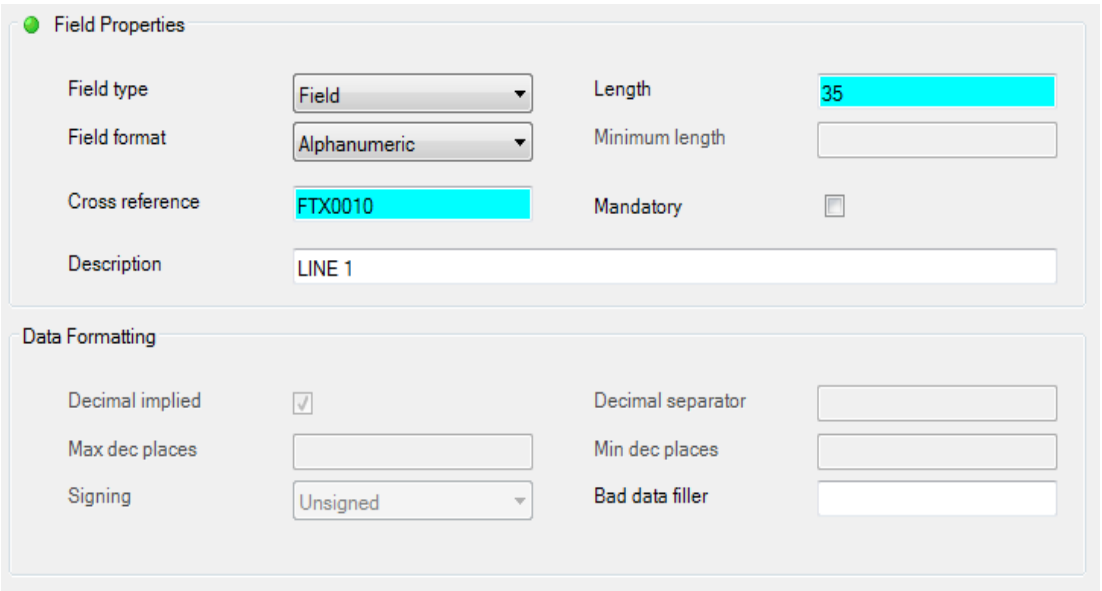
Sections are used in HDFs that represent TRA files to group elements together within a record. The section Name can be used to identify the purpose of the section, but is not mandatory and is not used in mapping.

Use the **Requirement** drop-down list to specify whether the section is mandatory or optional. Use the **Repeats** text box to specify the maximum number of repeats, where a value of 0 or 'unlimited' means the maximum repeat count is unbounded.

Use the **Description** text box to enter a free-form description of the section.

6.4.6 Field properties

The panel shown when a field is selected in the tree view or grid is as follows:



Use the **Field type** drop-down list to specify the field type. The permitted types are:

- **Field** – a normal field containing data

- **Filler** – a field containing spaces for padding that are ignored when processing the in-house file.
- **Record type** – a record type field (corresponds to TYPE in the HDF)
- **Record type filler** – a record type field that is written as filler when writing an in-house file (corresponds to TYPEF in the HDF)
- **Record type right** – a record type field that is right justified when writing an in-house file (corresponds to TYPER in the HDF)
- **Record type omit** – a record type field that is not written out when writing an in-house file (corresponds to TYPEX in the HDF)

Use the **Field format** drop-down list to specify the expected data format (alphabetic, numeric, alphanumeric). If you select a numeric format then the data formatting section is enabled so that more details can be specified.

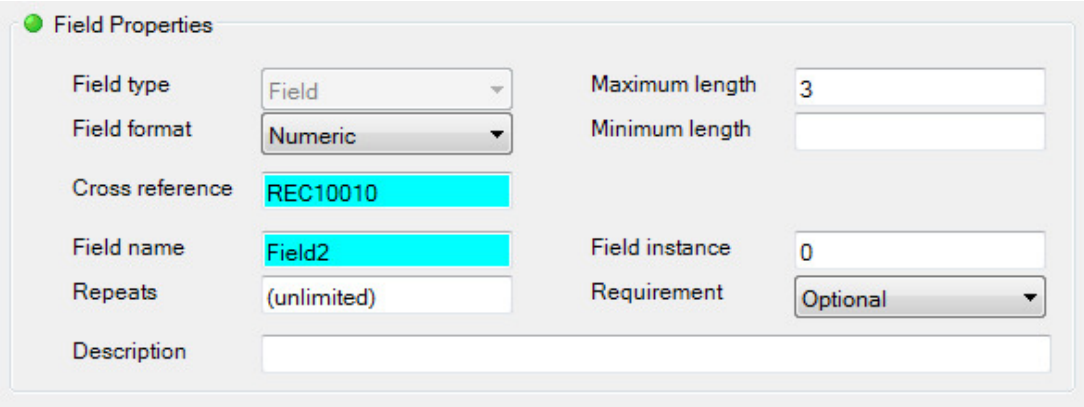
The **Cross reference** field is mandatory and is used in the mapper uniquely to identify a source or target field. When you add fields, cross-references are generated automatically and it should not often be necessary to change them. If you want to re-name all cross-references sequentially within records, choose the menu command **Definition >> Reset cross-references**.

If the field is fixed length (Delimited, Fixed and Variable record formats) then one **Length** text box is enabled into which you must enter the required field length. If the field is not fixed length then **Maximum length** and **Minimum length** text boxes are shown. The maximum field length must be entered and must be greater than zero. The minimum field length is optional.

The **Mandatory** check box can be used to indicate that the field is required. If a mandatory item is missing from the source file then a validation error will occur on loading.

Use the **Description** text box to enter a free-form description of the field.

There are additional properties in the field panel when the record type is TRA.



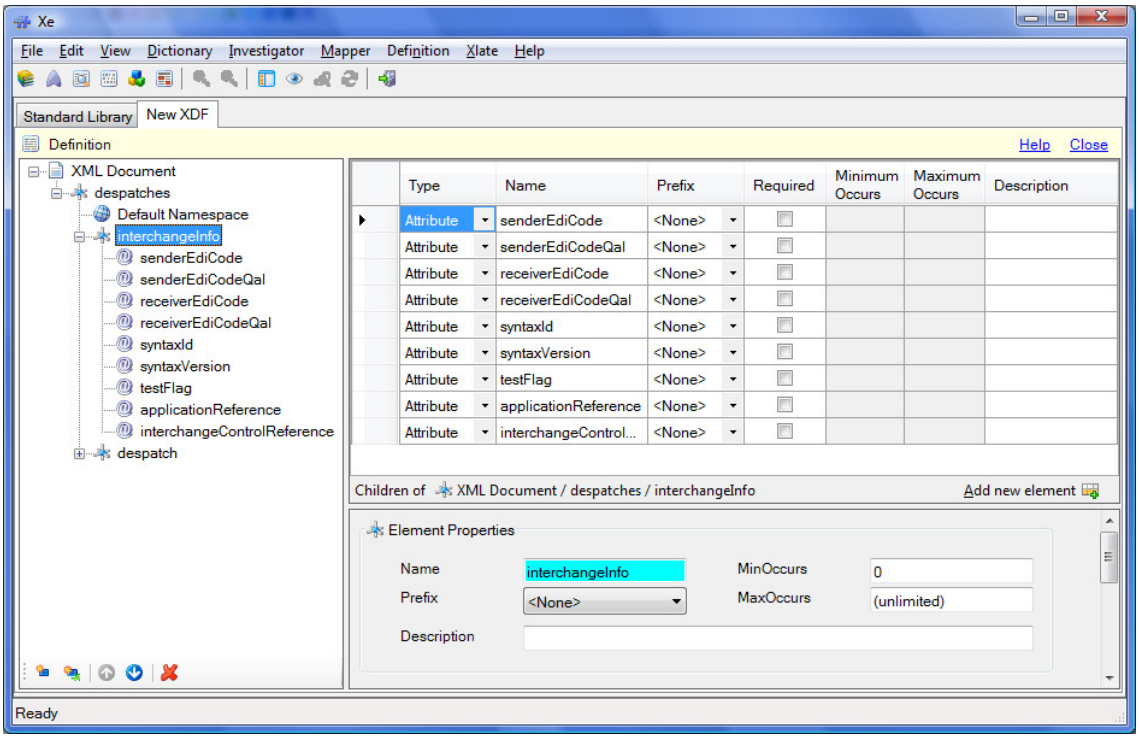
The **Field name** corresponds to the name used in the TRA file and must be specified `(field_name = value)`.

The **Field instance** is used with the record format T1 that specifies strict validation of the TRA file. It indicates an instance or level number that constrains the way in which fields repeat within sections (for instance if field instances in a section are 1, 2, 2, 3, 3, 4, then there must be an instance of field 1, there can be only one instance of each field 2 and field 3 and a field with a lower instance number cannot come after a field with a higher instance number unless it is a 1, indicating a new instance of the section).

Use the **Requirement** drop-down list to specify whether the section is mandatory or optional. Use the **Repeats** text box to specify the maximum number of repeats, where a value of 0 or 'unlimited' means the maximum repeat count is unbounded.

6.5 Create and edit XML definitions

Definitions are created and opened in the editor using the Index Explorer. An XML definition appears in the editor like this:



The following sections list the entities and properties shown in the grid view, according to what is selected in the tree, and the properties editable in the panel according to the selection in the tree or grid.

6.5.1 XML entities in the grid

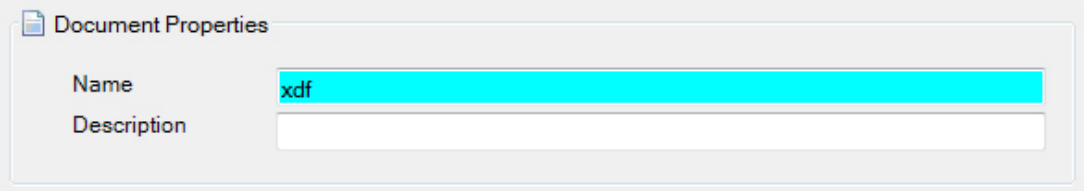
The following table lists the entities and properties (grid columns) shown in the child item grid, according to the element selected in the tree view.

Tree item	Child (grid) items	Grid column	Description
Document	Element	Type	Entity type drop-down (element only)
		Name	Element name (unqualified)
		Prefix	Element prefix (name qualifier corresponds to a URI given in a namespace declaration)
		Required	Not used for elements
		MinOccurs	The minimum number of occurrences expected
		MaxOccurs	The maximum number of occurrences permitted

		Description	A free text description of the element
Element	Element, Attribute	Type	Entity type drop-down (element or attribute)
		Name	Entity name (unqualified)
		Prefix	Entity prefix (name qualifier corresponds to a URI given in a namespace declaration)
		Required	Whether the attribute is required to be present in the XML document (attributes only)
		MinOccurs	The minimum number of occurrences expected (elements only)
		MaxOccurs	The maximum number of occurrences permitted (elements only)
		Description	A free text description of the entity
Attribute, namespace declaration	As attributes and namespace declarations have no children, the grid shows the children of the entity's parent element. Namespace declarations are shown in the tree view, but not the grid.		

6.5.2 Document properties

The panel shown when the document is selected in the tree view is as follows:

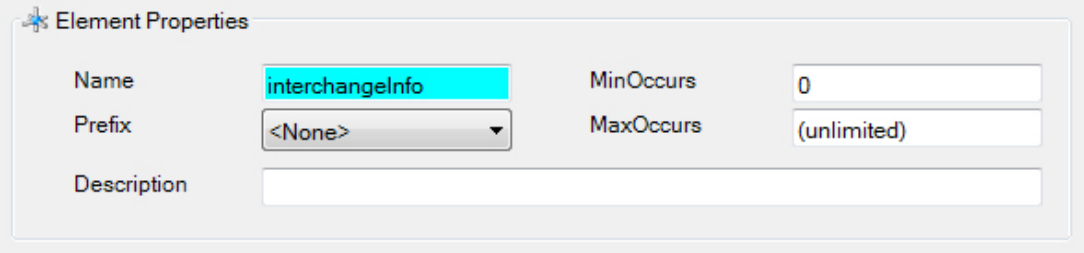


Use the **Name** text box to enter the name of the document.

Use the **Description** text box to enter a free-form description of the document.

6.5.3 Element properties

The panel shown when an element is selected in the tree view or grid is as follows:



Use the **Name** text box to enter the name of the element (unqualified; that is, without a namespace prefix). Use the **Prefix** drop-down list to choose a prefix and associate the element with a namespace. The prefixes available are those defined using namespace declarations that are in scope for the element.

Use the **MinOccurs** text box to enter the minimum number of occurrences of the element that are expected. Use the **MaxOccurs** text box to enter the

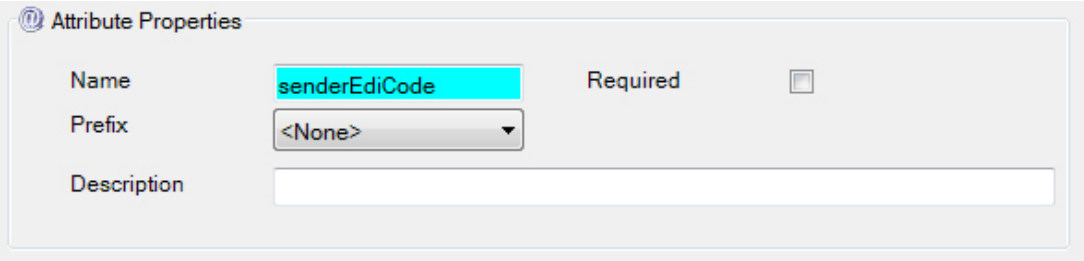
maximum number of occurrences of the element that are permitted (where a value of 0 or 'unlimited' means that the maximum repeat count is unbounded).

Use the **Description** text box to enter a free-form description of the element.

Use the Data formatting section to specify optional formatting information for the element.

6.5.4 Attribute properties

The panel shown when an attribute is selected in the tree view or grid is as follows:



Use the **Name** text box to enter the name of the attribute (unqualified; that is, without a namespace prefix). Use the **Prefix** drop-down list to choose a prefix and associate the attribute with a namespace. The prefixes available are those defined using namespace declarations that are in scope for the attribute.

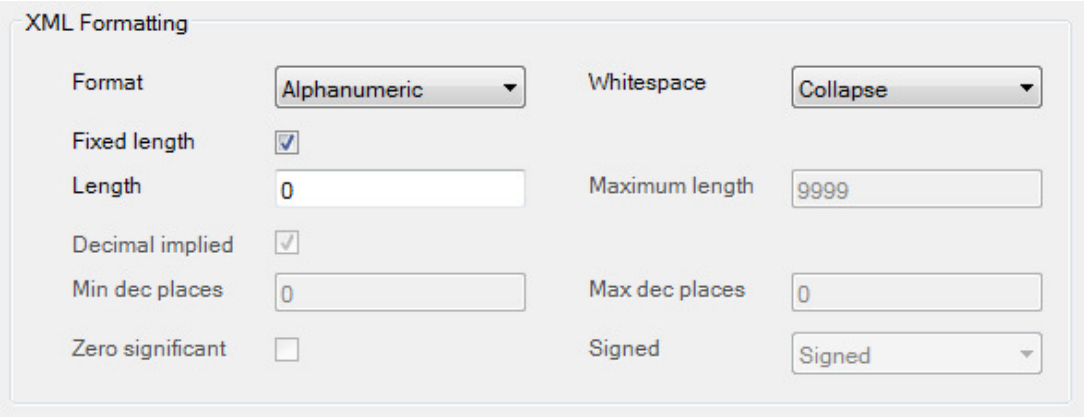
Use the **Required** check box to indicate whether the attribute is mandatory.

Use the **Description** text box to enter a free-form description of the attribute.

Use the Data formatting section to specify optional formatting information for the attribute.

6.5.5 Data formatting

The panel section shown for specifying data formatting for an element or attribute is as follows:



Use the **Field format** drop-down list to specify the expected data format (alphabetic, numeric, alphanumeric).

Use the **Whitespace** drop-down list to specify whitespace processing for the field (preserve, collapse or replace).

Use the **Fixed length** check box to indicate whether the field is a fixed length or can vary up to the maximum. If the field is fixed length then one **Length** text box is enabled into which you must enter the required maximum field length.

If the field is not fixed length then **Maximum length** and **Minimum length** text boxes are shown. The maximum field length must be entered and must be greater than zero. The minimum field length is optional.

If the field type is numeric then use the **Decimal implied** check box to indicate whether a decimal point is implied or explicit. If the decimal point is implied then one **Decimal places** text box is enabled into which you may enter the required number of implied decimal places. If the value is zero or missing then no decimal places are implied. If the decimal point is explicit then **Max dec places** and **Min dec places** text boxes are shown. Both values are optional.

If the field type is numeric then use the **Signed** drop-down list to indicate whether the value is signed (leading sign), unsigned or has a trailing sign.

6.5.6 Namespace properties

The panel shown when a namespace declaration is selected in the tree view is as follows:

Use this panel to associate a **Prefix** with a **Namespace** URI. Once a namespace declaration has been added the prefix will be available in the drop-down lists for elements and attributes.

7 Xe Reports

7.1 Introduction

Xe now includes a reporting framework with which you can quickly and easily create designs for custom printed reports and transport labels, map data from any document type to build report instances, and view and print them using Xe or a standalone report application.

The reporting framework includes these elements:

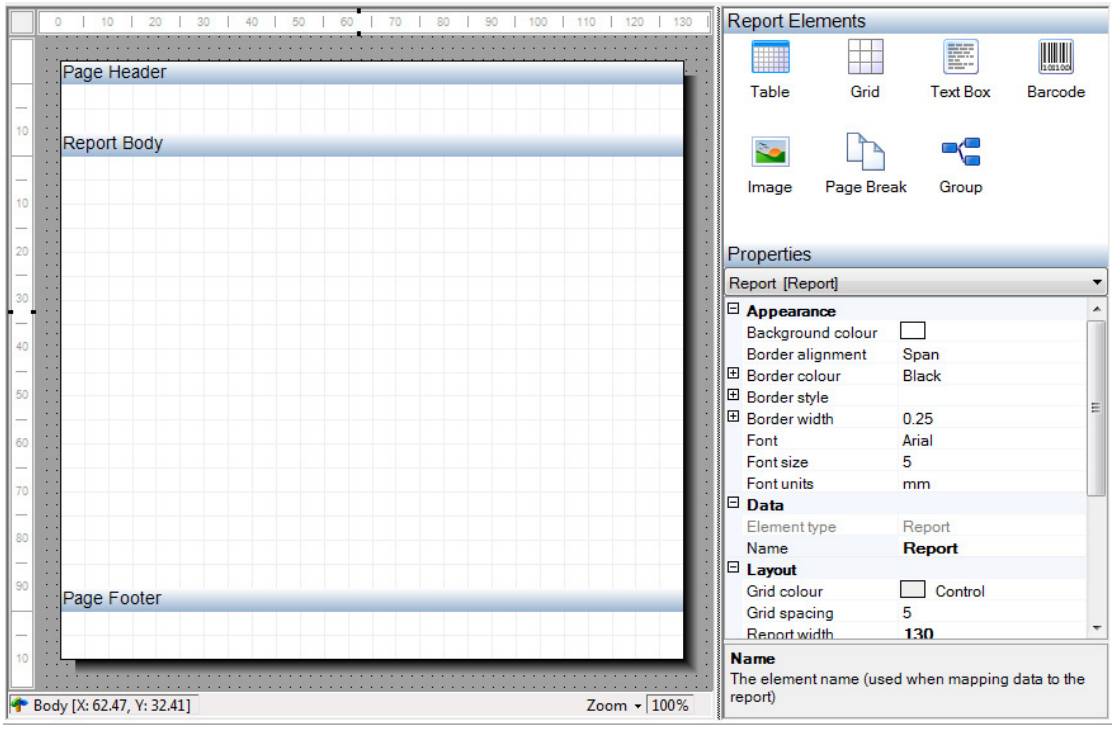
- **Report designer** – a visual designer for reports and labels, including layout, report element toolbox and property editor. The report designer is used to create Report Definition Files (RDF) that include all of the layout and style information you specify in the designer.
- **Mapping** – The Xe mapper now includes support for mapping data from any supported business document type to a report design (given by an RDF) to create report instance files (RPT files). A single report definition can be re-used many times to produce reports from different source data documents, including different document types if you wish.
- **View/print** – The Xe application includes a report viewer that allows you to view and print report instances (RPT) files. A separate report viewer (RPT.exe) is also available, which allows reports to be viewed or printed without using the main Xe application.

There are several stages involved in creating an Xe report

- **Design report** – Use the report designer to specify the layout of the desired report, element styles and fixed data.
- **Create source definition** – Create or locate a definition file for the source data document. You can do this by exporting a definition from the Data Dictionary, creating an XML definition from an XML instance file in the Mapper, building a definition from scratch in the Definition Editor, or using a definition for the source type that you already have for use in other maps.
- **Create the map** – Use the Xe mapper to create a new map project and map data from the source document definition to the report definition.
- **Test and refine** – Use the Xe application to test your report, by providing a source document and running the map, or getting the designer to create a report instance using ‘fake’ data it creates. Fine tune your report as needed.

7.2 Report designer

New report definitions are created and opened using the Index Explorer. The report designer is shown in the following image.



The key features of the designer are set out in the following sections:

7.2.1 Report layout

The larger part of the designer is taken up with the report layout, which is where you will position report elements during the design process. The layout is divided into three sections. The page header and page footer sections are where you will place report elements to be rendered on each printed page, at the top and bottom respectively. The report body section is where the substantive content of the report is placed.

Rulers are displayed along the top and left-hand edges of the layout and a position indicator is shown on each ruler according to the current position of the cursor over the layout.

The status bar at the bottom of the layout includes an information panel on the left hand side. If the cursor is not over a report element that you have placed on the layout then this panel shows the report section over which the cursor is hovering and the current cursor position (relative to the report section). If the cursor is over an element then the element type is displayed along with position and size information about the element.

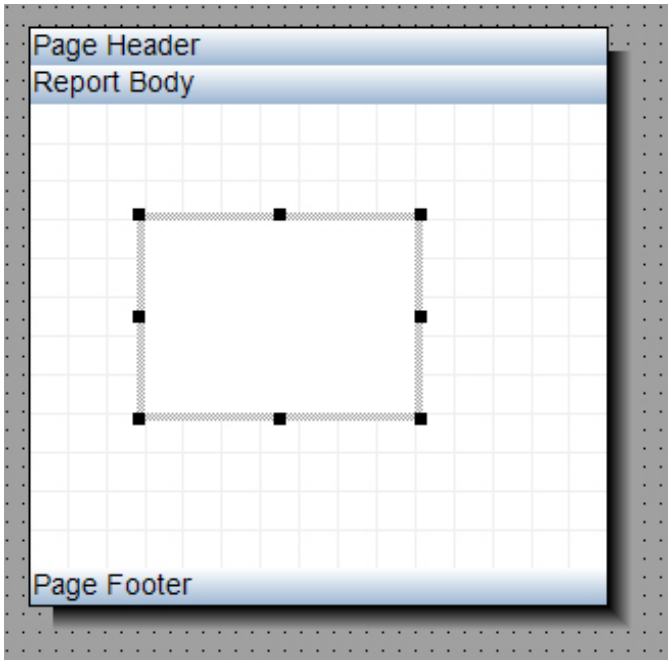
The status bar also includes a zoom indicator showing the current zoom factor. Clicking the **Zoom** button will pop up a menu with options to alter the zoom factor

Actions

The report layout section of the designer supports the following actions.

- Resize the report and report sections by dragging. Dragging the left or right hand side of the report with the mouse will increase or decrease the report width. Dragging the top of the report or the report body banner will resize the header section. Dragging the bottom of the report will resize the footer section. Dragging the page footer banner will resize the report body section.
- Manipulate elements. Report elements are placed on the layout by dragging from the report element toolbox. Once element have been placed on the layout they can be selected by clicking with the mouse. Once selected they

are drawn with highlighting and resize nodes around the outsize, and they can be moved or resized by dragging the edges or the nodes. Elements cannot be moved or resized so that their edges are outside the bounds of the report sections in which they exist. The image below shows the resize nodes and highlight around a selected text box.



- Scroll and zoom the report. Hold down the **Ctrl** key and use the **+** and **–** keys to zoom in and out. Hold down the **Ctrl** key and use the **arrow** keys to scroll the report within the layout when the scrollbars are shown.
- Pop-up menus. Right any element on the report layout to display a pop-up menu of options for manipulating it. Standard menu options include:
    - **Delete** – remove the selected element from the layout.
    - **Copy** – create a copy of the selected element.
    - **Paste** – insert the last copied element into the layout.
    - **Copy formatting** – copy the style information from the selected element (fonts, borders, colours &c).
    - **Paste formatting** – apply the last copied element styles to the selected element.
    - **Bring to front** – where report elements overlap, draws the selected element on top of any others.
    - **Send to back** – where report elements overlap, draws the selected element behind any others.
- Undo and redo. Hold down the **Ctrl** key and use the **Z** key to undo the last edit action performed in the report. Hold down the **Ctrl** key and use the **Y** key to redo the last undone edit action.

7.2.2 Report elements

The top right-hand section of the designer is the report element toolbox. The items shown here are report elements that can be placed on the report layout as part of the design process. The elements that you can use are as follows:

- **Table** – Used to display data in rows and columns. Typically a table will be used to show a variable number of data rows, with column headers and possibly a table footer. In the report instance the table is resized to accommodate all of the data rows required. Tables have sub-elements such as rows, columns and cells and may have 'inner tables' with their own rows, columns and cells. Tables cannot be used in the page header and page footer sections.
- **Grid** – Used to display text, images and/or barcodes in precise relative positions. Typically a grid will be used to define the layout for a transport label. The grid and the rows and cells within it are never resized in the report instance. Grids are conceptually made up of rows and cells, though only the cells are exposed as sub-elements. Grids cannot be used in the page header and page footer sections.
- **Text box** – Used to display a single block of text anywhere on the report. Optionally resizes itself to accommodate content.
- **Barcode** – Used to display a barcode anywhere on the report.
- **Image** – Used to display an image anywhere on the report.
- **Page break** – Used to indicate a new page (or new physical page) should be issued at a given point in the report body. Page breaks cannot be used in the page header and page footer sections.
- **Group** – Used to group together other report elements so that they can repeat. Groups are shown as paired group start and end lines on the designer. Elements between the line can be repeated any number of times within the report instance, depending on the data content mapped. Groups cannot be used in the page header and page footer sections.

Actions

The report element toolbox section of the designer supports the following actions.

- Drag report elements. Using the mouse, new report elements can be dragged from the toolbox and dropped on the layout. When dragging a new element onto the layout the cursor displays a cross to indicate that the element can be dropped or a forbidden image (no entry) to indicate that it cannot. Elements can only be dropped if there is room for them in the report section, given the intended drop site. Elements are created with a fixed default size, but this will be reduced to allow them to be dropped at a given location in the report section and not go outside the bounds of the section. However, element sizes cannot be reduced below certain fixed minima, and if a new element's size cannot be reduced to fit at a given location then you will not be allowed to drop the element there.

7.2.3 Element properties

The bottom right-hand section of the designer is the property editor. All of the report elements (including the report and report sections, and the sub-elements such as table and grid cells) have a number of properties that can be viewed and edited in this section. There are three parts to the property section.

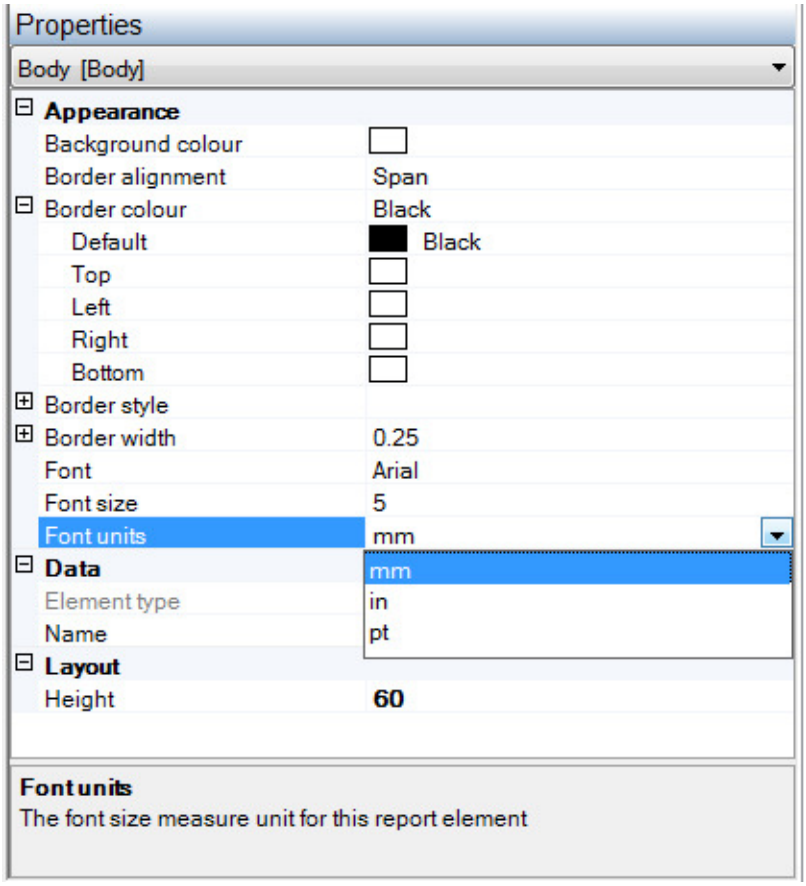
- **Element list** – At the top of the section is a drop-down list of elements and sub-elements defined in the report (including the report and the header, footer and body sections).

- **Property grid** – Displays the properties of the currently selected report element. Properties are broken into types or categories (such as Appearance, Layout and so forth). Some properties are compound, in that they are made up of other properties. For instance, an element's position property includes two values: X and Y coordinates. This is displayed with a small plus sign to the left of the property name.
- **Property description** – At the bottom of the property section is an area of text that displays a description of the currently selected property. The text provides a hint as to the effect of changing the property.

Actions

The property section of the designer supports the following actions.

- Select an element in the drop-down list to display its properties and select it in the layout.
- Edit properties in the grid and view the effects in the layout. The way in which properties are edited depends on the type. Some properties, such as the **Text** property in a text box are edited by simply highlighting the required property grid cell and typing in the text. Some properties present a list of values (for instance True and False) or a small control for selecting a value, indicated by a small drop-down arrow appearing in the cell when the property is highlighted for editing. Others display a dialog for editing, indicated by a small button with three dots. Compound properties with multiple sub-properties can be edited either by using the plus sign by the property name to display the sub-properties and editing them individually, or sometimes by entering an acceptable value for the compound property (for instance, the value "5,5" is acceptable for a position). Property values will be validated and invalid values will be changed back to the last valid value. Finally, some properties cannot be edited and are there for information only, in which case the name and value appear greyed out. The image shows a property grid with an expanded compound property (Border colour) and a drop-down list for selecting a value (Font units)



The properties of each element type are described in the following sections. For the most part, report design consists of adding, moving and sizing report elements on the layout, and setting their properties. Once the meanings of the properties are understood the use of the elements should be clear. There are specific sections later on in the guide, however, that describe the Table element and the Grid element in more detail.

7.2.4 Common properties

These properties are common to many or all report elements.

Data section

Property	Type	Description
Element type	Read-only (text)	The type of the selected element (includes the report itself, page headers and footers, report body and sub-elements, such as table and grid cells)
Name	Text	The element name. You should give elements meaningful names so that it is clear what data map to them. This is particularly import when creating tables or grids with many cells
Static	Drop-down list (Boolean)	Specifies whether the element contains fixed values. If the flag is set to True then it will not be possible to map data to this field using the Xe Mapper
Visible	Drop-down list (Boolean)	Specifies whether the element is visible in the printed report. This flag can be set to True or

		False in the Xe Mapper and therefore can be used to turn element on or off in a given report instance

Appearance section

Property	Type	Description
Background colour	Drop-down colour picker	The element background colour
Border alignment	Drop-down list	Indicates where the border is drawn in relation to the edges of the element. The values are Inside (drawn wholly within the element), Outside (drawn wholly outside the element) and Span (drawn half inside and half outside the element edges)
Border colour	Compound	The element border colour or colours. Specify a default colour that applies to all borders, and/or different values for one or more specific borders.
Border colour (default)	Drop-down colour picker	Specify the default border colour that applies to all borders unless overridden
Border colour (top)	Drop-down colour picker	Specify the colour for the top border
Border colour (left)	Drop-down colour picker	Specify the colour for the left border
Border colour (right)	Drop-down colour picker	Specify the colour for the right border
Border colour (bottom)	Drop-down colour picker	Specify the colour for the bottom border
Border style	Compound	The element border line style or styles. Specify a default line style that applies to all borders, and/or different values for one or more specific borders.
Border style (default)	Drop-down list	Specify the default border line style that applies to all borders unless overridden
Border style (top)	Drop-down list	Specify the line style for the top border
Border style (left)	Drop-down list	Specify the line style for the left border
Border style (right)	Drop-down list	Specify the line style for the right border
Border style (bottom)	Drop-down list	Specify the line style for the bottom border
Border width	Number (decimal)	The element border width or widths. Specify a default line width that applies to all borders, and/or different values for one or more specific borders.
Border width	Number	Specify the default border width that applies to

(default)	(decimal)	all borders unless overridden
Border width (top)	Number (decimal)	Specify the width for the top border
Border width (left)	Number (decimal)	Specify the width for the left border
Border width (right)	Number (decimal)	Specify the width for the right border
Border width (bottom)	Number (decimal)	Specify the width for the bottom border
Font	Dialog	Use the dialog to pick the font (family) and styles (bold, italic, underline, strikeout) for any text rendered in the element
Font size	Number (decimal)	Specify the font size for any text rendered in the element.
Font units	Drop-down list	Select the units in which the font size is measured (inches, millimetres or points)
Padding	Compound	The amount of padding, in millimetres, to be used between the content of an element and its borders.
Padding (default)	Number (decimal)	Specify the default padding that applies to all borders unless overridden
Padding (top)	Number (decimal)	Specify the padding between the top border and the content
Padding (left)	Number (decimal)	Specify the padding between the left border and the content
Padding (right)	Number (decimal)	Specify the padding between the right border and the content
Padding (bottom)	Number (decimal)	Specify the padding between the bottom border and the content

Layout section

Property	Type	Description
Position	Compound	The position of a report element relative to the section it is in
Position (X)	Number (decimal)	The X coordinate of the element's position (in millimetres)
Position (Y)	Number (decimal)	The Y coordinate of the element's position (in millimetres)
Size	Compound	The size of a report element
Size (width)	Number (decimal)	The width of the element in (millimetres)
Size (height)	Number (decimal)	The height of the element in (millimetres)

7.2.5 Report properties

These properties (or their meanings) are specific to the report itself

Layout section

Property	Type	Description
Grid colour	Drop-down colour picker	Specifies the colour in which the positioning grid in the layout should be rendered
Grid spacing	Number (decimal)	Specifies the space between lines (in millimetres) of the positioning grid in the layout
Report width	Number (decimal)	The width of the report in the designer (in millimetres). The width specified here governs the size you see in the layout and therefore the space available for elements. If he report is to fit on one page horizontally then the report width plus left and right margins must not be greater than the page width.
Show grid	Drop-down list (Boolean)	Specifies whether to display the positioning grid in the layout
Snap to grid/mm	Drop-down list	Specifies the 'snap to' behaviour of report elements. If set to None then elements can be dragged (move and resize) to any position or dimension. Alternatively you can specify that elements should 'snap' to the nearest millimetre or the positioning grid lines, in which case sizing and positioning of elements using the mouse is constrained accordingly (note that it is still possible to specify position and size values that are not aligned to the grid or nearest mm using the property grid)

Printing section

Printing options are stored with the report design so that report instances can be printed automatically with these settings. These values can be overridden, however, in the report viewer (discussed later).

Property	Type	Description
Duplex	Drop-down list (Boolean)	Indicates whether the report should be printed using duplexing if the printer supports is
Fit-to-page	Drop-down list	Specifies how the report should be rendered if it spans more than one page horizontally. A value of None indicates that the report should be printed across multiple pages horizontally. A value of ToWidth indicates the report should be zoomed to fit on one page horizontally. A value of ToWidthPreserveBarcodes has the same effect, except that the report engine will attempt to maintain the dimensions of any barcodes in grids. The ToWidth options are intended for use only where the difference between the report width and page size is small (for

		instance, fitting a transport label with fixed sizes onto the printed page where the printer's printable area would otherwise be too small)
Orientation	Drop-down list	Specifies whether the report is designed to be printed in portrait or landscape mode.
Page height	Number (decimal)	The height (in millimetres) of the page onto which the report will be printed (by default)
Page margins	Compound	The sizes of the margins between the edge of the page and the report contents
Page margins (left)	Number (decimal)	The sizes of the margins between the left-hand edge of the page and the report contents
Page margins (right)	Number (decimal)	The sizes of the margins between the right-hand edge of the page and the report contents
Page margins (top)	Number (decimal)	The sizes of the margins between the top edge of the page and the report contents
Page margins (bottom)	Number (decimal)	The sizes of the margins between the bottom edge of the page and the report contents
Page width	Number (decimal)	The width (in millimetres) of the page onto which the report will be printed (by default)

7.2.6 Report section properties

These properties (or their meanings) are specific to the report sections (page header and footer, report body).

Layout section

Property	Type	Description
Height	Number (decimal)	The height of the section (in millimetres) in the designer layout. The heights of the header and footer sections translate directly to the printed report. This is not true of the body section, which typically contains elements whose size will vary according to the data in a particular instance of the report.

7.2.7 Table properties

These properties (or their meanings) are specific to the table element.

Data section

Property	Type	Description
Rows can repeat	Drop-down list (Boolean)	Indicates whether rows are permitted to repeat within the table. The default value is True as the table is designed to display multiple instances of data rows. If the value is set to False then only one instance of the row will be shown in a table.

7.2.8 Table column properties

These properties (or their meanings) are specific to the table column element.

Data section

Property	Type	Description
Column index	Read-only (integer)	The 1-based index of the column in the table
Sorting	Drop-down list	If the data rows mapped to the table are to be sorted according to the values in this column, then this property indicates the sort type and direction

Layout section

Property	Type	Description
Width	Number (decimal)	The width of the column (in millimetres)

7.2.9 Table row properties

These properties (or their meanings) are specific to the table row element.

Data section

Property	Type	Description
Row index	Read-only (integer)	The 1-based index of the row in the table
Parent	Read-only (text)	Indicates whether the row is part of the table body, table header or table footer.

Layout section

Property	Type	Description
Height	Number (decimal)	The height of the row (in millimetres). Bear in mind that if the cells in the row are permitted to shrink or grow according to content then the size of the row as rendered in the report instance may be different from that given here

7.2.10 Table cell properties

These properties (or their meanings) are specific to the table cell element.

Data section

Property	Type	Description
Barcode	Dialog	Use the dialog to specify the type of barcode to be shown in the cell and its characteristics. Note that if a barcode is specified then the value of the Text property (or any text mapped to the cell by the Xe Mapper) will be rendered as the barcode data content. If an image and barcode are both specified the image will not be shown.
Change field	Drop-down list (Boolean)	Indicates whether the cell value should be rendered only if it is different from the value in the previous instance (previous row)
Draw line	Compound	Specifies characteristics of a horizontal line to be drawn in the cell. The line is drawn one or both sides of any text (depending on the content alignment chosen) but never over the text.
Draw line (colour)	Drop-down colour picker	The colour of the line to draw through the cell
Draw line (style)	Drop-down list	The style of the line to draw through the cell
Draw line (width)	Number (decimal)	The width of the line to draw through the cell
Image	Dialog	Use the dialog to select an image that will be rendered to the cell. Images can be imported into a report using this dialog and are then available for use throughout the report. Images imported into the report are copied and stored with the report definition. This means that if an image is changed it needs to be re-imported to update the report definition with the changes. It also means that using large images or many different images in a report will result in a large definition file and large report instance files. If an image and barcode are both specified the image will not be shown. If an image and text are both specified then the text will be rendered over the top of the image.
Image mode	Drop-down list	Specifies how the image is to be sized and positioned within the cell. A value of Zoom will increase or decrease the image size to fit in the cell while maintaining the aspect ration. A value of Stretch will increase or decrease the image size to fill the cell ignoring the aspect ratio. Other values will result in the image being placed in the specified location (e.g. centre, bottom left, left middle) without resizing. If barcode settings are specified then the image mode specified will be applied to the barcode and no image will be shown.
Parent	Read-only (text)	Indicates whether the cell belongs to a row that is part of the table body, table header or table footer.
Text	Text	The text to be rendered to the cell. If barcode

		settings are specified then the value here is used as the barcode's data content and no text is displayed. If text and an image are both specified then both will be shown, with the text being rendered over the top of the image. If the cell is mapped to in the Xe Mapper then mapped data will be rendered in preference to any fixed text entered here. Fixed text (such as column headings) can be protected against mapping by setting the Static property to True. This prevents them from appearing in the mapper.

Format section

Property	Type	Description
Numeric format	Dialog	Use the dialog to specify how numeric data in the cell is to be rendered (format, width and precision)

Layout section

Property	Type	Description
Align (horizontal)	Drop-down list	Indicates the horizontal alignment of text in the cell (images and barcodes are not affected by this property and are aligned according to the Image mode property)
Align (vertical)	Drop-down list	Indicates the vertical alignment of text in the cell (images and barcodes are not affected by this property and are aligned according to the Image mode property)
Auto-size	Drop-down list	Indicates if the cell height should be allowed to grow or shrink (or both, or neither) according to the data content.
Word wrap	Drop-down list (Boolean)	Indicates whether text should be word-wrapped if it is too long for the cell (the alternative is truncation)

7.2.11 Grid cell properties

These properties (or their meanings) are specific to the grid cell element.

Data section

Property	Type	Description
Barcode	Dialog	Use the dialog to specify the type of barcode to be shown in the cell and its characteristics. Note that if a barcode is specified then the value of the Text property (or any text mapped

		to the cell by the Xe Mapper) will be rendered as the barcode data content. If an image and barcode are both specified the image will not be shown.
Caption	Compound	Specifies the text and characteristics of the cell caption. The caption is an extra text value that can be entered into grid cells. Typically it will be used to insert some title text into a transport label cell, where it can be used in conjunction with any other content (text, barcode, image)
Caption (align horizontal)	Drop-down list	Specifies the horizontal alignment of the caption in the cell
Caption (align vertical)	Drop-down list	Specifies the vertical alignment of the caption in the cell
Caption (font)	Dialog	Use the dialog to pick the font (family) and styles (bold, italic, underline, strikeout) for the caption text.
Caption (font colour)	Drop-down colour picker	Specify the font colour for the caption text.
Caption (font size)	Number (decimal)	Specify the font size for the caption text.
Caption (font units)	Drop-down list	Select the units in which the font size is measured (inches, millimetres or points)
Caption (padding)	Compound	The amount of padding, in millimetres, to be used between the caption and the cell borders and other content
Caption (padding (default))	Number (decimal)	Specify the default padding that applies to all borders unless overridden
Caption (padding (top))	Number (decimal)	Specify the padding on the top edge of the caption
Caption (padding (right))	Number (decimal)	Specify the padding on the right edge of the caption
Caption (padding (left))	Number (decimal)	Specify the padding on the left edge of the caption
Caption (padding (bottom))	Number (decimal)	Specify the padding on the bottom edge of the caption
Caption (text)	Text	The text to be rendered in the caption
Caption (width)	Number (decimal)	The horizontal extent in (millimetres) of the rendered caption. Setting this property to a value other than zero allows the caption to be word wrapped to fit the given width. If zero, then the text is not wrapped.
Image	Dialog	Use the dialog to select an image that will be rendered to the cell. Images can be imported into a report using this dialog and are then available for use throughout the report. Images imported into the report are copied and stored with the report definition. This means that if an

		image is changed it needs to be re-imported to update the report definition with the changes. It also means that using large images or many different images in a report will result in a large definition file and large report instance files. If an image and barcode are both specified the image will not be shown. If an image and text are both specified then the text will be rendered over the top of the image.
Image mode	Drop-down list	Specifies how the image is to be sized and positioned within the cell. A value of Zoom will increase or decrease the image size to fit in the cell while maintaining the aspect ratio. A value of Stretch will increase or decrease the image size to fill the cell ignoring the aspect ratio. Other values will result in the image being placed in the specified location (e.g. centre, bottom left, left middle) without resizing. If barcode settings are specified then the image mode specified will be applied to the barcode and no image will be shown.
Text	Text	The text to be rendered to the cell. If barcode settings are specified then the value here is used as the barcode's data content and no text is displayed. If text and an image are both specified then both will be shown, with the text being rendered over the top of the image. If the cell is mapped to in the Xe Mapper then mapped data will be rendered in preference to any fixed text entered here. Fixed text (such as column headings) can be protected against mapping by setting the Static property to True. This prevents them from appearing in the mapper.

Format section

Property	Type	Description
Numeric format	Dialog	Use the dialog to specify how numeric data in the cell is to be rendered (format, width and precision)

Layout section

Property	Type	Description
Align (horizontal)	Drop-down list	Specifies the horizontal alignment of text in the cell (images and barcodes are not affected by this property and are aligned according to the Image mode property)
Align (vertical)	Drop-down list	Specifies the vertical alignment of text in the cell (images and barcodes are not affected by

		this property and are aligned according to the Image mode property)
Align to caption	Drop-down list	Specifies the manner in which grid content (text, image or barcode) is aligned with respect to the caption. If a value other than None is specified then content is rendered according to the other alignment properties (horizontal and vertical alignment for text; image mode for barcodes and images) within the part of the cell 'left over' after the caption alignment has been considered. For example, a value of Full indicates that content is aligned to the caption both horizontally and vertically. This means that if the caption is in the top-left of the cell, the part of the cell available for rendering the main content stretches from the bottom-right of the rectangle occupied by the caption to the bottom-right of the cell. Alignment to caption is not perform in respect of captions that are themselves aligned in the middle of a cell, horizontally or vertically. Alignment to caption options include 'overlap' variants, which are used to specify that the main content of a cell should be allowed to overlap the caption if this is necessary to fit it in the cell.
Height	Number (decimal)	The height of the cell
Width	Number (decimal)	The width of the cell
Word wrap	Drop-down list (Boolean)	Indicates whether text should be word-wrapped if it is too long for the cell (the alternative is truncation)

7.2.12 Text box properties

These properties (or their meanings) are specific to the text box element.

Data section

Property	Type	Description
Draw line	Compound	Specifies characteristics of a horizontal line to be drawn in the text box. The line is drawn one or both sides of any text (depending on the content alignment chosen) but never over the text.
Draw line (colour)	Drop-down colour picker	The colour of the line to draw through the text box.
Draw line (style)	Drop-down list	The style of the line to draw through the text box.
Draw line (width)	Number (decimal)	The width of the line to draw through the text box.
Text	Text	The text to be rendered to the text box. If the

		text box is mapped to in the Xe Mapper then mapped data will be rendered in preference to any fixed text entered here. Fixed text (such as column headings) can be protected against mapping by setting the Static property to True. This prevents them from appearing in the mapper.

Format section

Property	Type	Description
Numeric format	Dialog	Use the dialog to specify how numeric data in the text box is to be rendered (format, width and precision)

Layout section

Property	Type	Description
Align (horizontal)	Drop-down list	Indicates the horizontal alignment of text in the text box.
Align (vertical)	Drop-down list	Indicates the vertical alignment of text in the text box.
Auto-size	Drop-down list	Indicates if the text box height should be allowed to grow or shrink (or both, or neither) according to the data content.
Word wrap	Drop-down list (Boolean)	Indicates whether text should be word-wrapped if it is too long for the text box (the alternative is truncation)

7.2.13 Barcode box properties

These properties (or their meanings) are specific to the barcode box element.

Data section

Property	Type	Description
Barcode	Dialog	Use the dialog to specify the type of barcode to be shown in the box and its characteristics.
Image mode	Drop-down list	Specifies how the barcode is to be sized and positioned within the box. A value of Zoom will increase or decrease the size to fit in the box while maintaining the aspect ratio. A value of Stretch will increase or decrease the barcode size to fill the box ignoring the aspect ratio. Other values will result in the barcode being placed in the specified location (e.g. centre, bottom left, left middle) without resizing.
Text	Text	The data to be rendered to the barcode. If the

		barcode box is mapped to in the Xe Mapper then mapped data will be rendered in preference to any fixed text entered here. A fixed value can be protected against mapping by setting the Static property to True. This prevents the text property from appearing in the mapper.

7.2.14 Image box properties

These properties (or their meanings) are specific to the image box element.

Data section

Property	Type	Description
Image	Dialog	Use the dialog to select an image that will be rendered to the image box. Images can be imported into a report using this dialog and are then available for use throughout the report. Images imported into the report are copied and stored with the report definition. This means that if an image is changed it needs to be re-imported to update the report definition with the changes. It also means that using large images or many different images in a report will result in a large definition file and large report instance files.
Image mode	Drop-down list	Specifies how the image is to be sized and positioned within the image box. A value of Zoom will increase or decrease the image size to fit in the box while maintaining the aspect ratio. A value of Stretch will increase or decrease the image size to fill the box ignoring the aspect ratio. Other values will result in the image being placed in the specified location (e.g. centre, bottom left, left middle) without resizing.

7.2.15 Page break properties

These properties (or their meanings) are specific to the image box element.

Data section

Property	Type	Description
New physical page	Drop-down list (Boolean)	Specifies whether the page break results in a new physical page (ensures that the next page is odd-numbered when printing duplex)

Layout section

Property	Type	Description

Position	Number (decimal)	The vertical offset (in millimetres) of the page break from the top of the report body section.

7.2.16 Group start and group end properties

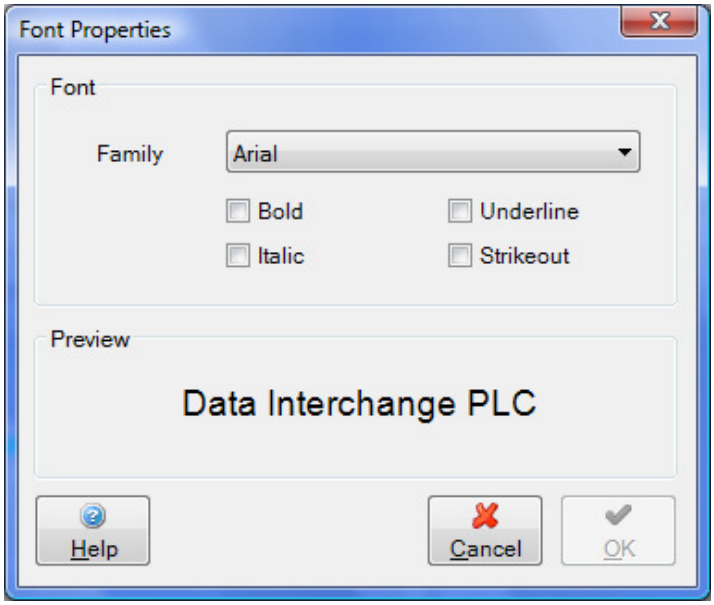
These properties (or their meanings) are specific to the group start and group end elements.

Layout section

Property	Type	Description
New page	Drop-down list	Specifies whether the group start or end results in a new page (or a new physical page) with the option to suppress a new page on the first occurrence
Position	Number (decimal)	The vertical offset (in millimetres) of the group divider from the top of the report body section.

7.2.17 Font properties dialog

The following dialog is used to select a font family and font styles.



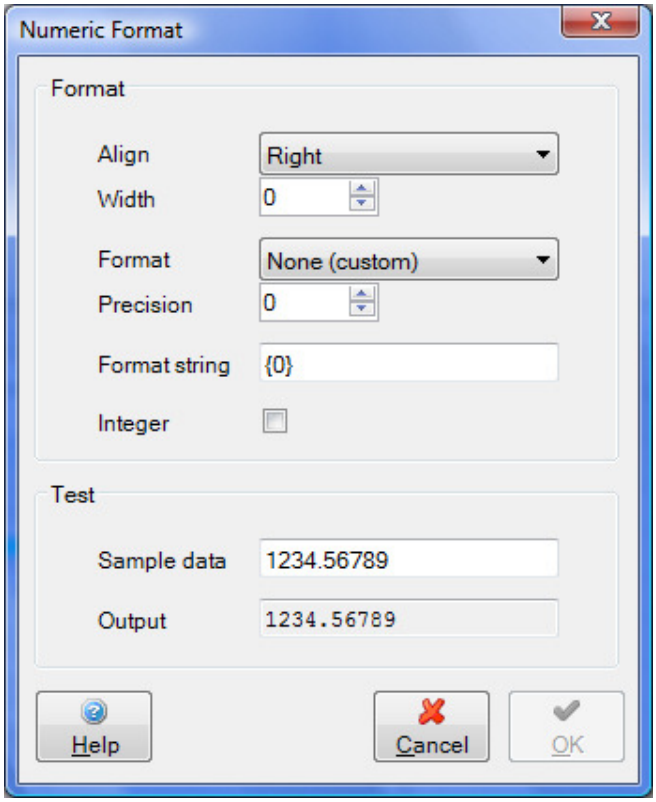
Use the **Family** drop-down list to select a font family (typeface) from the list of fonts installed on your computer. If the reports you produce are to be viewed on other computers then choose only common fonts.

Check or uncheck the boxes to add or remove the **Bold**. **Italic**, **Underline** and **Strikeout** font styles. The preview panel shows some text formatted according to the font family and styles you have chosen.

When you are satisfied with the font selected click **OK**, To exit the dialog without saving your changes click **Cancel**.

7.2.18 Numeric format dialog

The following dialog is used to specify the format of numeric values rendered to a text box or cell.



The dialog is used to specify properties that will produce a format string for writing text values as numbers in the given format. The **format string** produced is a standard .NET format string, and you can enter one directly if you are familiar with the form. Alternatively, set the properties individually and the format string will be generated for you.

The **Align** drop-down list allows you to select how the number is to be aligned (note that this is the alignment within the width you specify, not the alignment within the cell or text box in the report).

The **Width** control is used to specify the minimum width for the rendered numeric string.

The **Format** drop-down list allows you to select from the range of pre-set formats available.

The **Precision** control is used to specify the precision for the rendered numeric string (number of digits or decimal places).

The **Integer** check box is used to specify that the number is to be converted to an integer (rather than a decimal) before formatting, which is required for some format types.

The **Sample data** text box allows you to enter a number to test the property values specified. The number you enter will be formatted according to the current settings and displayed in the **Output** box.

The available formats are as follows:

- **Currency** – Thee number is converted to a string that represents a currency amount using the current regional settings on your computer. The precision specifier indicates the desired number of decimal places.
- **Decimal** – The number is converted to a string of decimal digits (0 to 9), prefixed by a minus sign if the number is negative. The precision specifier indicates the minimum number of digits desired in the resulting string. If

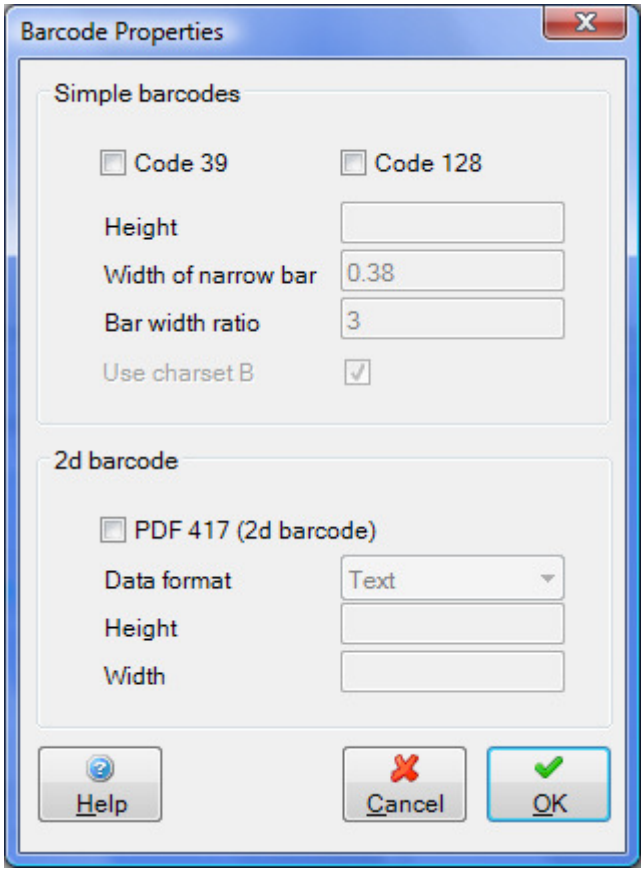
required, the number is padded with zeros to its left to produce the number of digits given by the precision specifier. Applies only to integer types.

- **Scientific** - The number is converted to a string of the form "d.ddde+ddd", where each 'd' indicates a digit. The string starts with a minus sign if the number is negative. The precision specifier indicates the desired number of digits after the decimal point.
- **Fixed point** – The number is converted to a string of the form "ddd.ddd" where each 'd' indicates a digit. The string starts with a minus sign if the number is negative. The precision specifier indicates the desired number of decimal places.
- **General** – The number is converted to the most compact of either fixed point or scientific notation, depending on the type of the number and whether a precision specifier is present.
- **Number** – The number is converted to a string of the form "d,ddd.ddd", where 'd' indicates a digit, ',' indicates a thousand separator between number groups, and '.' indicates a decimal point symbol. The precision specifier indicates the desired number of decimal places.
- **Percent** – The number is converted to a string that represents a percent. The converted number is multiplied by 100 in order to be presented as a percentage. The precision specifier indicates the desired number of decimal places. If the precision specifier is omitted.
- **Hexadecimal** – The number is converted to a string of hexadecimal digits. Applies only to integer types. The precision specifier indicates the minimum number of digits desired in the resulting string.

When you are satisfied with the settings click **OK**, To exit the dialog without saving your changes click **Cancel**.

7.2.19 Barcode properties dialog

The following dialog is used to select a barcode type and specify properties.



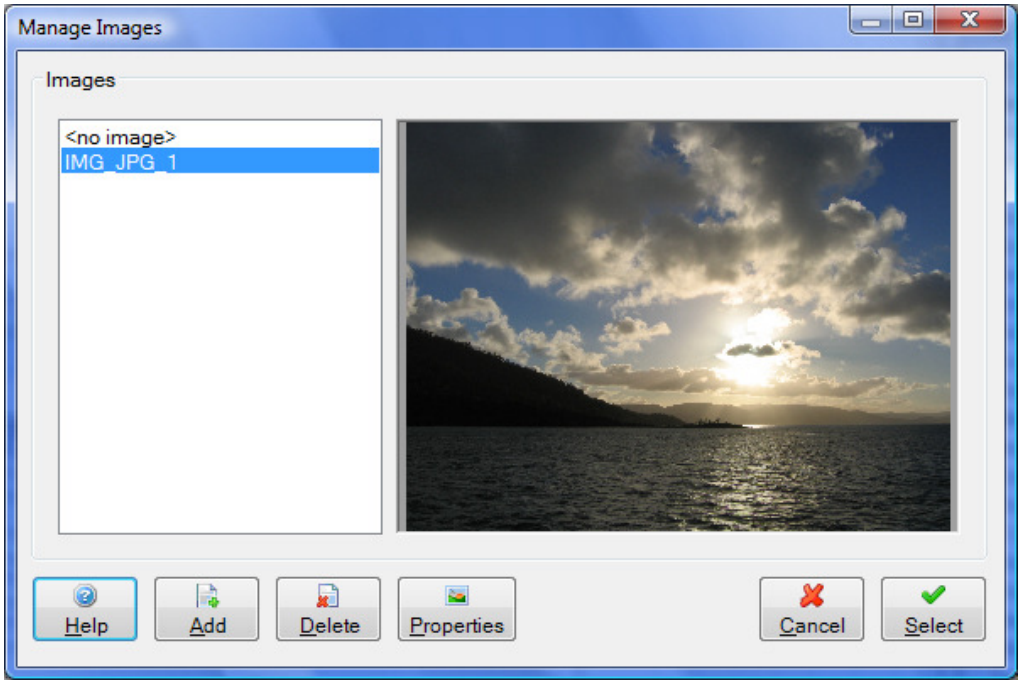
Use the check boxes to select either the Code 39 or Code 128 simple barcodes, or the PDF 417 2d barcode type. If no box is checked it means that you do not want a barcode to appear in the box or cell. When you select a barcode type, other parts of the dialog are enabled for you to specify properties:

- **Height** – Specify the rendered height of the barcode (all types).
- **Width** – Specify the rendered width of the barcode (PDF417 only).
- **Data format** – Specify the data format (text or binary) (PDF417 only).
- **Width of narrow bar** – Specify the width of a narrow data bar in millimetres (Code 39 and Code 128).
- **Bar width ratio** – Specify the ratio of the narrow to wide bar width (Code 39 only).
- **Charset B** – Specify that the extender character set may be used (Code 128 only)

When you are satisfied with the details entered click **OK**, To exit the dialog without saving your changes click **Cancel**.

7.2.20 Image selection dialog

The following dialog is used to manage images imported into a report definition and select images for display in cells and image boxes.



The list on the left hand side contains the names of all images that have been imported into the report definition. Select an image to view a preview.

Click the **Add** button to browse for a new image to import into the definition.

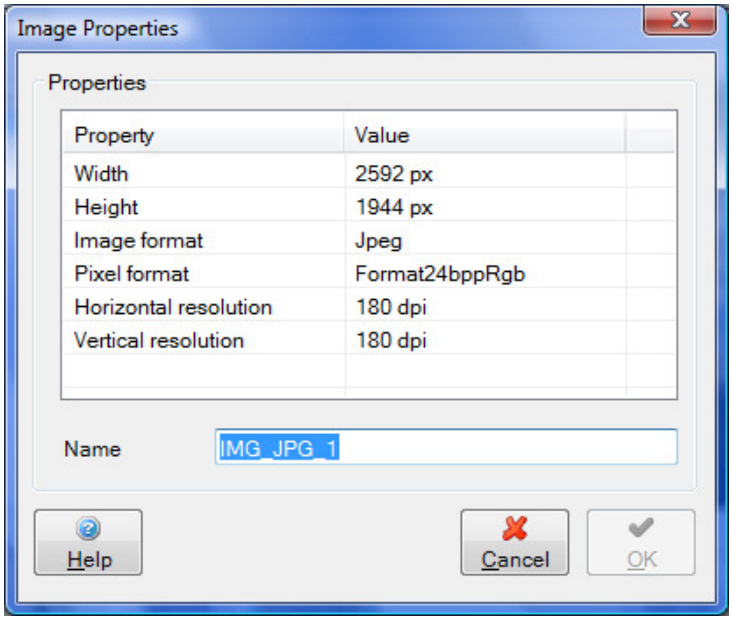
Click the **Delete** button to remove the image selected in the list

Click the **Properties** button to show properties for the selected image and change its internal name.

Click the **Cancel** button to close the dialog without selecting an image.

Click the **OK** button to close the dialog and select the current image (or <no_image> if you want to specify that no image should be displayed).

The dialog shown when the **Properties** button is clicked is shown below



The dialog shows a list of image properties and their values. These values cannot be edited

The internal name by which the image is known within the report definition can be edited by changing it in the text box.

When you are satisfied with the details entered click **OK**, To exit the dialog without saving your changes click **Cancel**.

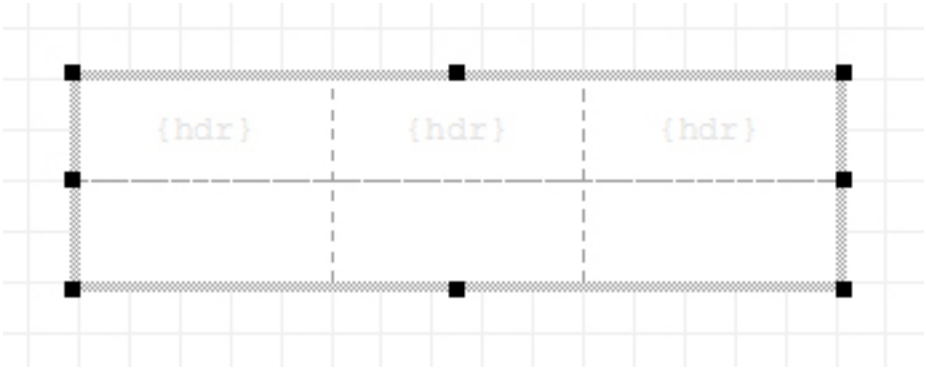
7.2.21 Report preview

While developing a report in the designer you can view a preview of what an instance of the report might look like with data in it, using the report preview facility. Choose **Mapper >> Preview report** from the menu, or use the keyboard shortcut **Ctrl + Shift + P** to display a dialog containing the preview.

7.2.22 Table element

The table element is used to display data in rows and columns. Typically a table will be used to show a variable number of data rows, depending on the source data being mapped to it.

A table comprises zero or more header rows, one or more body (or data) rows and zero or more footer rows arranged across one or more columns. By default, tables are created with one header row and one body row cross three columns. All of the substantive data in a table is entered into cells, which exist at the intersections of rows and columns (though they can be made to span columns as we will see).



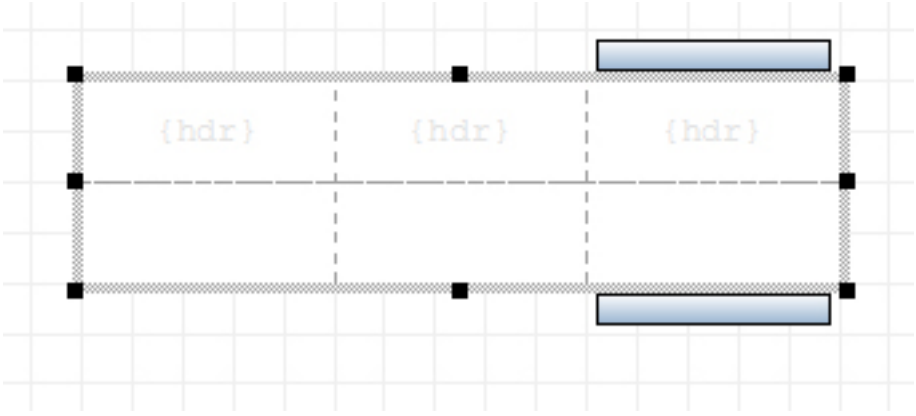
Where there is no content in a header row cell, it contains the greyed out value “{hdr}” (empty footer row cells contain the value “{ftr}”).

7.2.22.1 Editing

You can enter fixed text into any cell by setting the **Text** property, and you can make this immutable from the mapper by setting the **Static** property to **true**. Once the table has been selected in the layout, cells can be selected and their properties edited individually in the property grid. It is also possible to select multiple cells by holding down the **Ctrl** key while clicking on cells. Changes made to cell properties while multiple cells are selected will be applied to all of them.

Cells can also contain barcodes or images. These can be configured by setting the relevant cell properties in the property grid (see Table cell properties).

A row or column can be selected by moving the mouse over the row or column, but just outside the bounds of the table. Doing so causes a selection tab to pop up. Clicking the tab selects the row or column in the grid. The following image shows a table with a column selection tab displayed.



When the table is highlighted, rows and columns can be resized by dragging with the mouse. Let the mouse hover over any internal border between cells (horizontal or vertical) and the cursor will change to indicate that the line can be dragged. Drag the line to resize the column to the left or the row above. Note that other rows and columns retain their sizes when neighbours are resized, so reducing the height of a row reduced the height of the table, and *vice versa*. We will see later on that the grid behaves differently.

7.2.22.2 Menu

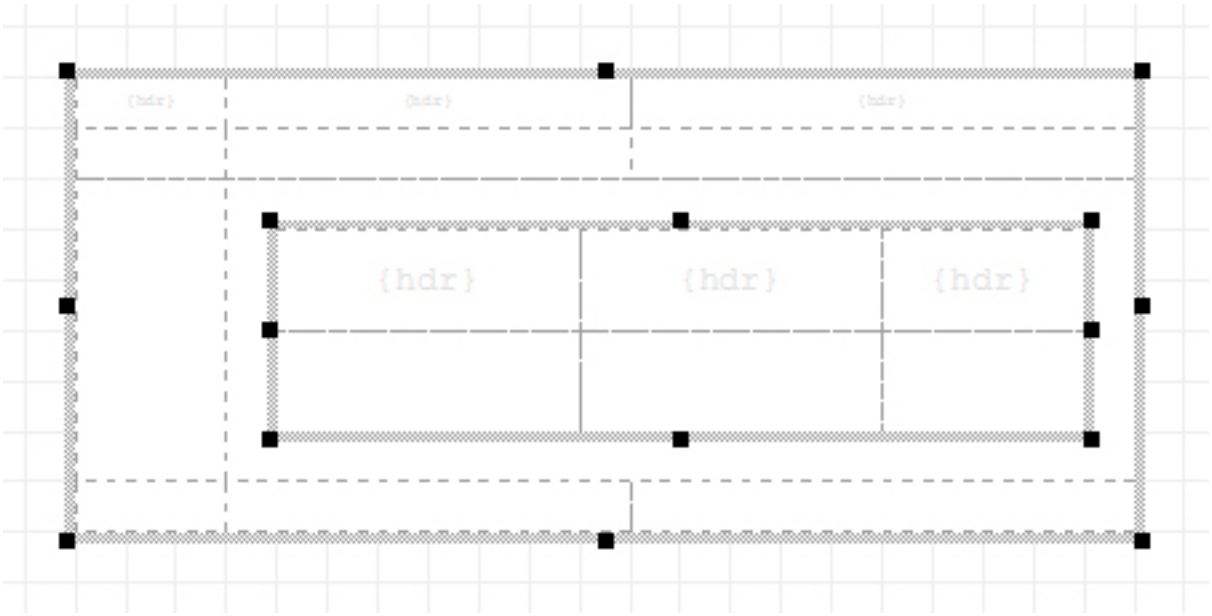
Right-clicking on any cell in the table (while the table is selected) or any row or column selection tab (when displayed) causes a pop-up menu to be shown. The menu items displayed are as follows:

- Select all cells – all cells in the table are selected for editing.
- Insert column to the left – add a new column to the table to the left of the selected column.
- Insert column to the right – add a new column to the table to the right of the selected column.
- Delete column – remove the selected column.
- Equalise column widths – set the widths of each column in the table to the same value without changing the overall table width.
- Insert row above – add a new row to the table above the current one (the row will be of the same type as the current one: header, footer or body).
- Insert row below – add a new row to the table below the current one (the row will be of the same type as the current one: header, footer or body).
- Insert header row – if no header rows are present then this option allows you to add a single header row (further header rows are added by bringing up the menu for the new header row and choosing to “add row above” or “add row below”)
- Insert footer row – if no footer rows are present then this option allows you to add a single footer row (further footer rows are added by bringing up the menu for the new footer row and choosing to “add row above” or “add row below”)
- Join cells – this option is shown when two or more contiguous cells in the same row are selected; it allows you to create a single cell that spans multiple columns.

- Split cell – this option is shown when a single cell that spans multiple columns is selected; it allows you to split the spanning cell into multiple cells, one for each spanned column.
- Insert inner table – inserts an inner table into a cell (see below).
- Clear inner table – removes an inner table from a cell (see below).
- Centre inner table – centres an inner table both horizontally and vertically within a cell.

7.2.22.3 Inner tables

Inner tables can be used to include additional rows of data, across different columns, within the data rows of a standard table. Inner tables have all of the features of standard tables, including header and footer rows, and can contain the same content types (text, images and barcodes). They can be resize and moved within the containing cell in the same way as standard tables are moved and resized within report sections. Inner tables will typically be used to display data that is hierarchically subordinate to data in the main table. The following image shows an inner table in a table in the designer:



Note that the inner table occupies a cell that spans tow rows of the main table, and that it has a header row of its own, which can be used to display column headers for the inner rows. Although the inner table is shown here with gaps between its borders and the borders of the containing cell, it will be more usual to omit these gaps, which will otherwise appear in the report instance.

There can be only one inner table in any main table row, and the report designer will not offer the menu option to insert an inner table into a cell if there is already one in the row.

7.2.22.4 Rendering issues

When a report instance is rendered the table will be resized according to the number of instances of each data row. Rows will also be resized according to the data content of cells if the cells have certain properties set. Specifically, the **Auto-size** property indicates whether a cell can grow or shrink or both, or neither) according to data content. A row can grow only if all cells in the row can grow, and the same applies *mutatis mutandis* to shrinking. Also, cells will not

grow in response to text data content unless their **Word wrap** properties are set to **true**.

Table headers are rendered at the start of each table, and at the start of each subsequent page onto which the table spans. Footers are rendered only once after the last body row. Inner table rows are rendered wholly within outer table rows. One outer table row is added for each instance of the inner table row.

If a table is too wide for the printed page and there is no scaling option set, then the table will be rendered across two or more pages, with only whole columns displayed on each page. If one column is too wide for a printed page then it is an error and the table cannot be rendered.

7.2.22.5 Table placeholders (count and sum)

Placeholders are available for use in tables only that support counting the instances or summing the values of cells in the table. The placeholders are as follows:

%TCNT_cellname% - counts the number of instances so far of the named cell in the current instance of the table.

%TSUM_cellname% - sums the values of the instances so far of the named cell in the current instance of the table.

%TCNTP_cellname% - counts the number of instances so far of the named cell in the current instance of the table on the current page.

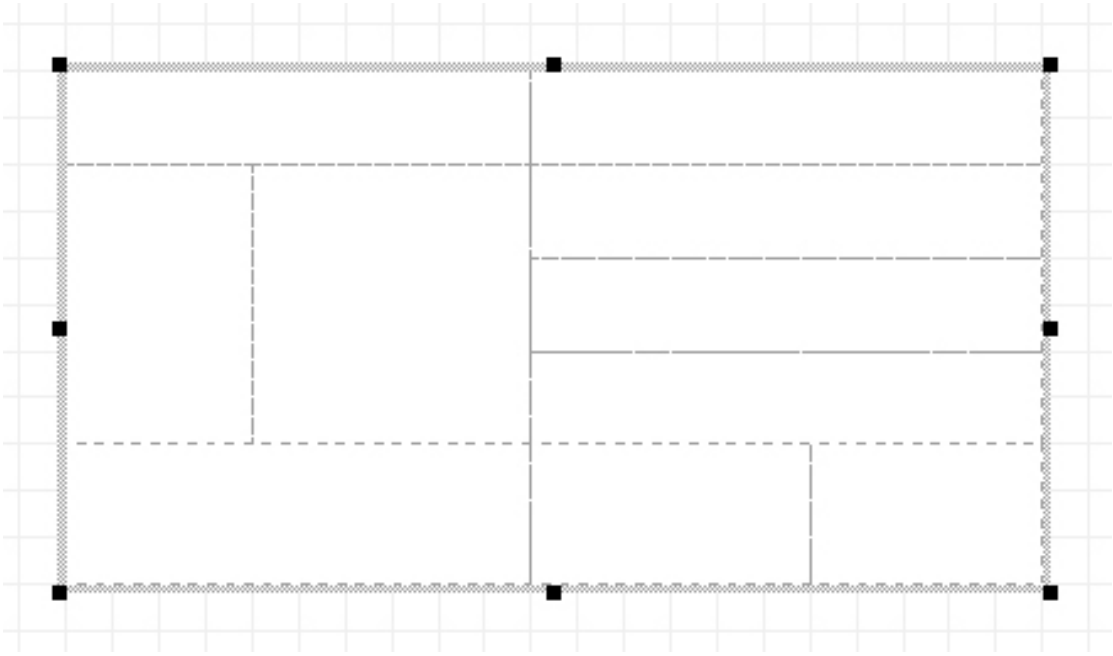
%TSUMP_cellname% - sums the values of the instances so far of the named cell in the current instance of the table on the current page.

The 'cellname' part of the placeholder must be entered to coincide with the name of a cell in the table. If the cell name changes you will need to change the placeholder as well. If using one of the ‘sum’ placeholders, then the values in the cells should all be convertible to a number.

7.2.23 Grid element

The grid element is used to display text, images and/or barcodes in precise relative positions. Typically a grid will be used to define the layout for a transport label.

A grid comprises one or more rows, each of which contains one or more cells, each of which may contain one or more rows &c. By default, grids are created with three rows, each with two cells. All of the substantive data in a grid is entered into cells. Rows exist only as containers for the cells. Careful adding of rows and cells allows you to build an arbitrarily complex grid that will support any transport label or other report structure requirement. The image below shows a sample grid.

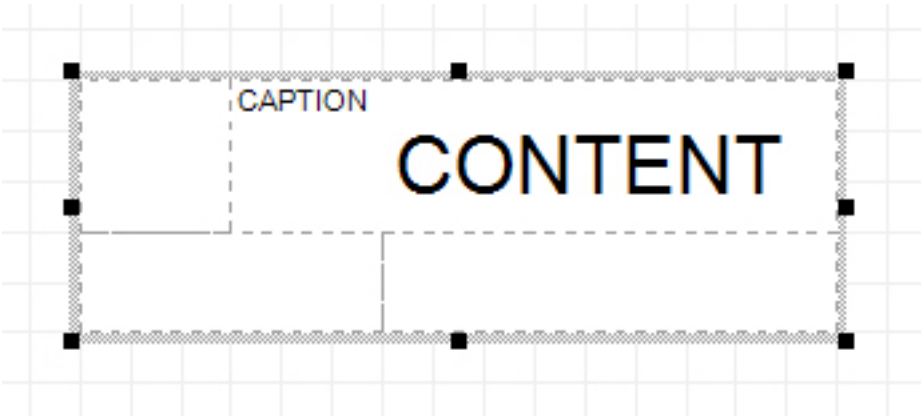


7.2.23.1 Editing

You can enter fixed text into any cell by setting the **Text** property, and you can make this immutable from the mapper by setting the **Static** property to **true**. Once the grid has been selected in the layout, cells can be selected and their properties edited individually in the property grid. It is also possible to select multiple cells by holding down the **Ctrl** key while clicking on cells. Changes made to cell properties while multiple cells are selected will be applied to all of them.

Cells can contain barcodes or images. These can be configured by setting the relevant cell properties in the property grid (see Grid cell properties).

Grid cells can also have captions. Captions are additional text fields that appear as titles in a cell. Typically they are used to name fields on transport labels.



The grid cell element has a compound **Caption** property. The sub-properties of **Caption** include the caption **Text** and a number of properties for formatting and aligning the caption (see Grid cell properties).

When the grid is highlighted, rows and cells can be resized by dragging with the mouse. Let the mouse hover over any internal border between cells (horizontal or vertical) and the cursor will change to indicate that the line can be dragged. Drag the line to resize the cell to the left or the row above. Note that other rows and cells are resized as well so that the overall size of the grid is maintained. This behaviour is designed to support one of the main features of the grid,

which is that is allows precise sizing and positioning of cells and their contents. We have seen that the table behaves differently.

7.2.23.2 Menu

Right-clicking on any cell in the grid (while the grid is selected) causes a pop-up menu to be shown. The menu items displayed are as follows:

- Select all cells – all cells in the grid are selected for editing.
- Insert row above – add a new row to the grid above the current one.
- Add row below – add a new row to the grid below the current one.
- Delete row – remove the current row from the grid.
- Split cell vertically (insert cell) – create a new cell by splitting the current cell into two vertically.
- Split cell horizontally (insert row) – create a new row with a single cell in it by splitting the current cell into two horizontally.
- Delete cell – remove the current cell from the grid.

7.2.23.3 Rendering issues

When a report instance is rendered neither the grid nor any of its rows and cells are resized, except to the extent that they are affected by the scaling property you set against the report. If a grid is too wide for the printed page and scaling is turned off then it is an error and the grid cannot be rendered.

7.2.24 Placeholders

The reporting framework supports several placeholders for inserting dynamic data into a report instance. To make use of a placeholder, either enter its name as fixed text in a text box or cell in the designer, or map its name in the Xe Mapper. The supported placeholders are set out in the following table:

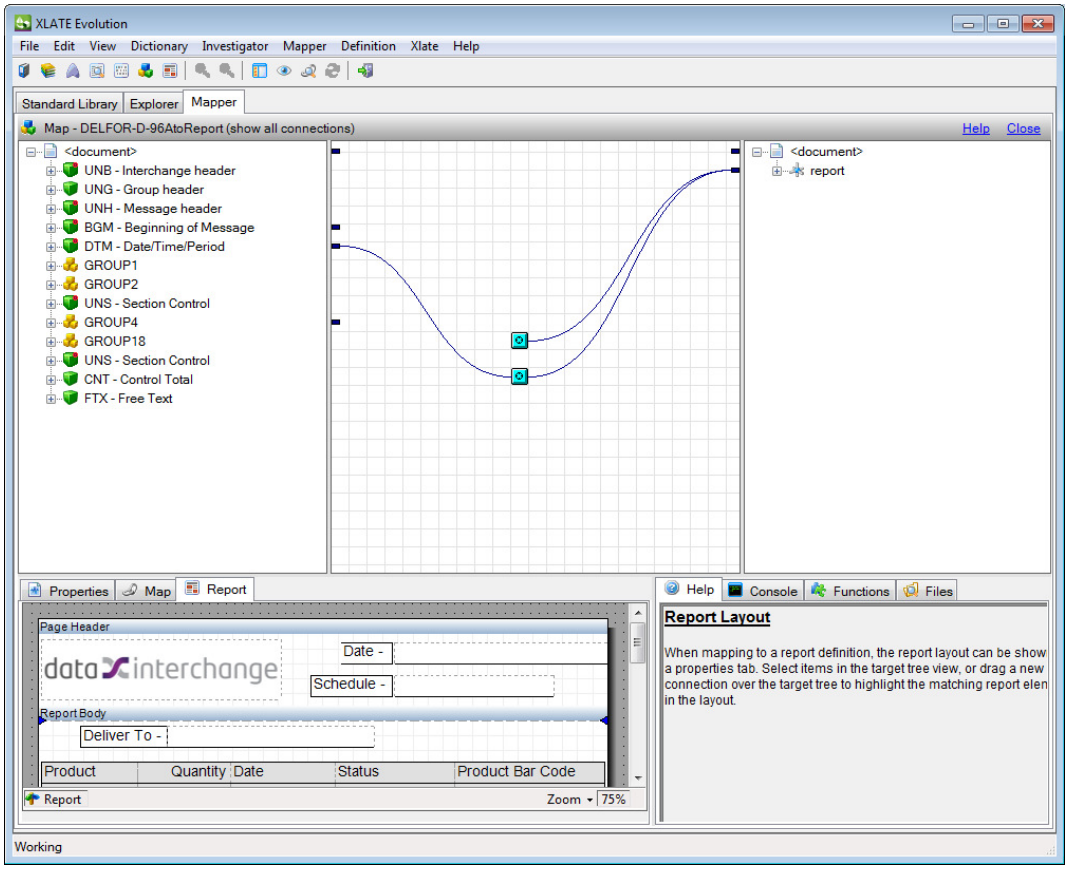
Placeholder	Description
%PGN%	Represents the current page number in the report instance
%PGT%	Represents the total number of pages in the printed report
%PNN%	Represents the current page number an the total number of pages in the form “Page x of n”
%RDT%	Represents the date and time the report was created in the format “05/06/2007 08:09:10”
%CDT%	Represents the date and time the report was rendered (viewed or printed) in the format “05/06/2007 08:09:10”
%RDT_fmt%	Represents the date and time the report was created in a format you specify by including a valid format string in place of “fmt”
%CDT_fmt%	Represents the date and time the report was rendered (viewed or printed) in a format you specify by including a valid format string in place of “fmt”

The “fmt” parameter of the date-time placeholders can be any valid date-time format string. The formats supported are the same as those listed in the Custom date-time formats section for the mapper date-time functions. For example, use “%RDT_D%” to output the report date in the format “05 June 2007”.

7.3 Creating reports in the mapper

Once you have created a report design it can be used to display data from any source document supported in the Xe mapper. The mapper GUI can be used to map data from an arbitrary source document to build an instance of the report.

Use the Index Explorer to create a map with the required source document definition and the report definition as the target. When the mapper is opened it will look something like this:



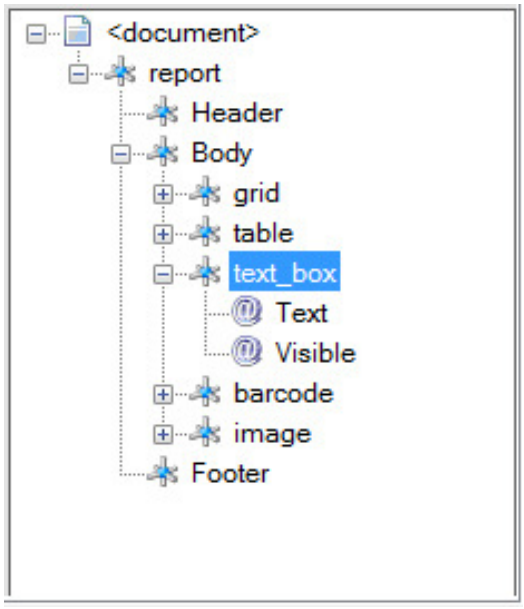
The principal features to note at first glance are:

- The report which is the target of the map is shown in the tree view on the right hand side as a hierarchical document type (in fact, as if it were an XML document). The report appears in the hierarchy as the parent of the header, body and footer sections and the sections appear as the parents of their respective elements.
- You can map to the report definition (in the tree view) in the same way you would map to any other document type (dragging connections, adding literal or constant values &c). Built-in functions and code snippets can be used as normal.
- You can only map to elements in the target that can take data values, or to special flags that appear as attributes. It makes no sense, for instance, to map a value to the **Header** element. More on this later, but note that the text box has a **Text** ‘attribute’ and a **Visible** ‘attribute’. A value mapped to the Text attribute will appear as the text in the text box in the report instance. A value mapped to the Visible attribute will determined whether the element is shown or hidden (values “false”, “no”, “n” and “0” equate to false, other values to true). Mapping to the text box element, creates a connection to the Text attribute instead.

- The report layout is shown in a tab in the properties section of the mapper at the bottom left of the screen. When you highlight an item in the target tree view, or when you hover over a target tree node while dragging a new connection from the source tree view, the element in the report layout that corresponds to the tree view node is highlighted and scrolled into view.

7.3.1 Mapping to a text box

The following image shows the appearance of a text box in the target tree view of the mapper.



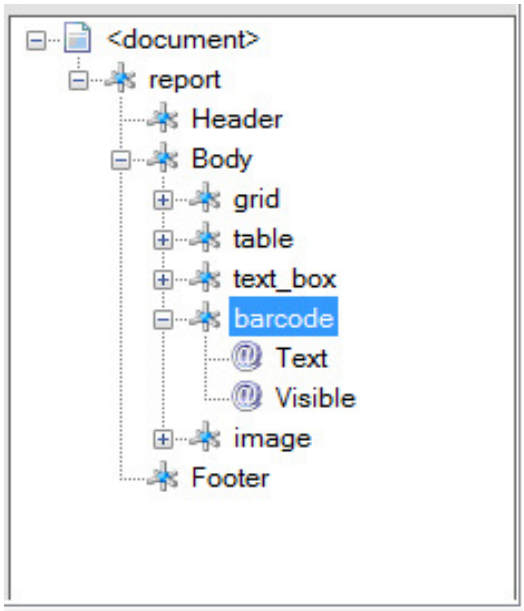
The text box element has two “attributes” in the target tree view: **Text** and **Visible**.

The text box is shown only if its **Static** property is set to **false** in the report designer. Map a value to its **Text** attribute in order to have it rendered to the report. If no value is mapped and the **Text** property of the text box is set in the report designer then that text will be rendered to the report.

The **Visible** attribute can be used to show or hide the text box in the report instance. If the **Visible** property of the text box is set in the report designer then the value of that property will determine its visibility, but it can be overridden by mapping a value to the **Visible** attribute (values of “false”, “no”, “n” and “0” equate to a Boolean value of **false**; anything else is **true**).

7.3.2 Mapping to a barcode box

The following image shows the appearance of a barcode box in the target tree view of the mapper.



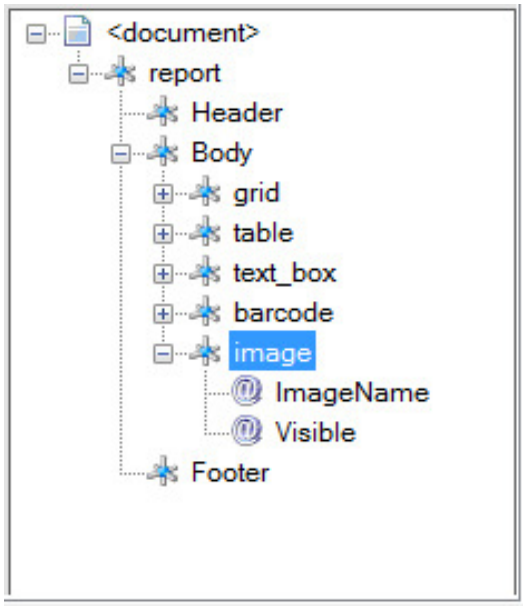
The barcode box element has two “attributes” in the target tree view: **Text** and **Visible**.

The barcode box is shown only if its **Static** property is set to **false** in the report designer. Map a value to its **Text** attribute in order to have it rendered to the report as a barcode. If no value is mapped and the **Text** property of the barcode box is set in the report designer then that text will be rendered to the barcode in the report.

The **Visible** attribute can be used to show or hide the barcode box in the report instance. If the **Visible** property of the barcode box is set in the report designer then the value of that property will determine its visibility, but it can be overridden by mapping a value to the **Visible** attribute (values of “false”, “no”, “n” and “0” equate to a Boolean value of **false**; anything else is **true**).

7.3.3 Mapping to an image box

The following image shows the appearance of an image box in the target tree view of the mapper.



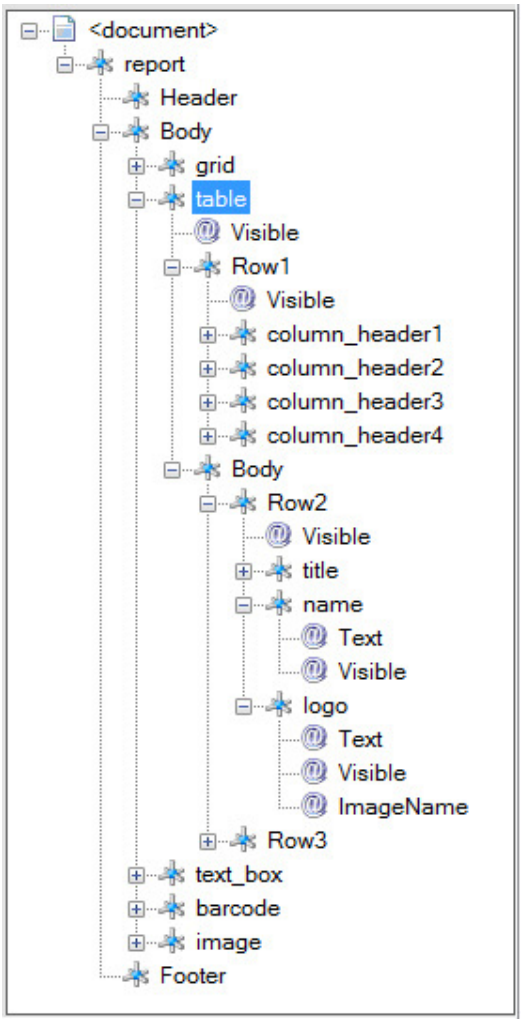
The image box element has two “attributes” in the target tree view: **ImageName** and **Visible**.

The image box is shown only if its **Static** property is set to **false** in the report designer. Normally you will choose an image to show in an image box by selecting it in the report designer, but the **ImageName** attribute allows you to set or override the image to be displayed at map time. Map the name of an image imported into the report definition to this attribute in order to have it rendered to the report.

The **Visible** attribute can be used to show or hide the image box in the report instance. If the **Visible** property of the image box is set in the report designer then the value of that property will determine its visibility, but it can be overridden by mapping a value to the **Visible** attribute (values of “false”, “no”, “n” and “0” equate to a Boolean value of **false**; anything else is **true**).

7.3.4 Mapping to a table

The following image shows the appearance of a table in the target tree view of the mapper.



The table element one “attribute” in the target tree view: **Visible**.

The **Visible** attribute can be used to show or hide the table in the report instance. If the **Visible** property of the table is set in the report designer then the value of that property will determine its visibility, but it can be overridden by mapping a value to the **Visible** attribute (values of “false”, “no”, “n” and “0” equate to a Boolean value of **false**; anything else is **true**).

The table element has child elements representing the content of the table. It has a child element representing a table header row. It also has two grandchild

rows representing table body rows. These are contained in a single body row element. The body row element is required to provide a hierarchy for the map that will support multiple repeating rows in the table body (Row2, Row3, Row2, Row3 &c). Each row has a **Visible** attribute that can be used to turn it on and off in the report instance (as described above for the table itself).

Each row has a number of child elements that represent table cells. The substantive content of the table is mapped to the cells. Cells appear in the tree view only if their **Static** properties are set to **false** in the report designer. Cells have two or three “attributes” in the target tree view: **Text**, **Visible** and (optionally) **ImageName**.

Map a value to the **Text** attribute in order to have it rendered to the cell in the report. If no value is mapped and the **Text** property of the cell is set in the report designer then that text will be rendered to the report.

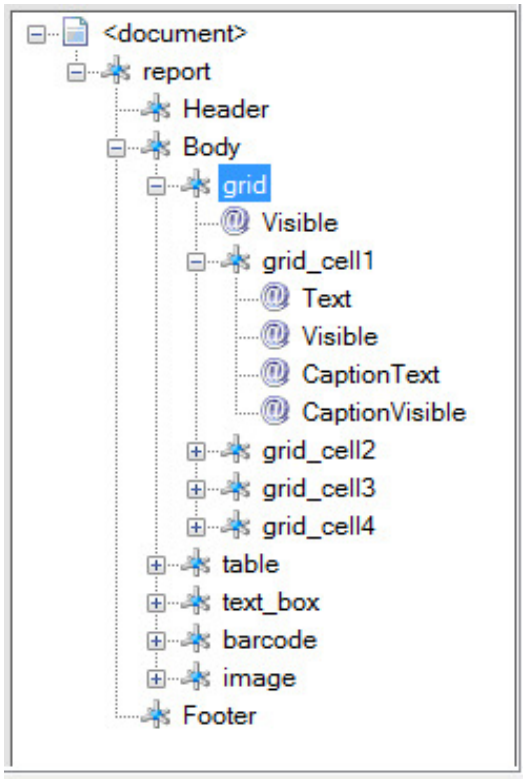
If barcode properties are configured against a table cell then mapping data to the cell has the result that the value mapped is rendered as the barcode data (no text is displayed in the cell).

The **Visible** attribute can be used to show or hide the cell in the report instance (as described above for the table itself).

The **ImageName** attribute appears if an image is specified against the cell in the report designer. Normally you will choose an image to show in a table cell by selecting it in the designer, but the **ImageName** attribute allows you to set or override the image to be displayed at map time. Map the name of an image imported into the report definition to this attribute in order to have it rendered to the cell in the report.

7.3.5 Mapping to a grid

The following image shows the appearance of a grid in the target tree view of the mapper.



The grid element one “attribute” in the target tree view: **Visible**.

The **Visible** attribute can be used to show or hide the grid in the report instance. If the **Visible** property of the grid is set in the report designer then the value of that property will determine its visibility, but it can be overridden by mapping a value to the **Visible** attribute (values of “false”, “no”, “n” and “0” equate to a Boolean value of **false**; anything else is **true**).

The grid element has child elements representing the content of the grid. The child elements represent the grid’s cells. Note that the grid’s rows are not shown in the tree view. The substantive content of the grid is mapped to the cells. Cells appear in the tree view only if their **Static** properties are set to **false** in the report designer. Cells have four or five “attributes” in the target tree view: **Text**, **Visible**, **CaptionText**, **CaptionVisible** and (optionally) **ImageName**.

Map a value to the **Text** attribute in order to have it rendered to the cell in the report. If no value is mapped and the **Text** property of the cell is set in the report designer then that text will be rendered to the report.

If barcode properties are configured against a grid cell then mapping data to the cell has the result that the value mapped is rendered as the barcode data (no text is displayed in the cell).

The **Visible** attribute can be used to show or hide the cell in the report instance (as described above for the grid itself).

Map a value to the **CaptionText** attribute in order to have it rendered to the cell in the report. If no value is mapped and the **Caption Text** property of the cell is set in the report designer then that text will be rendered to the report.

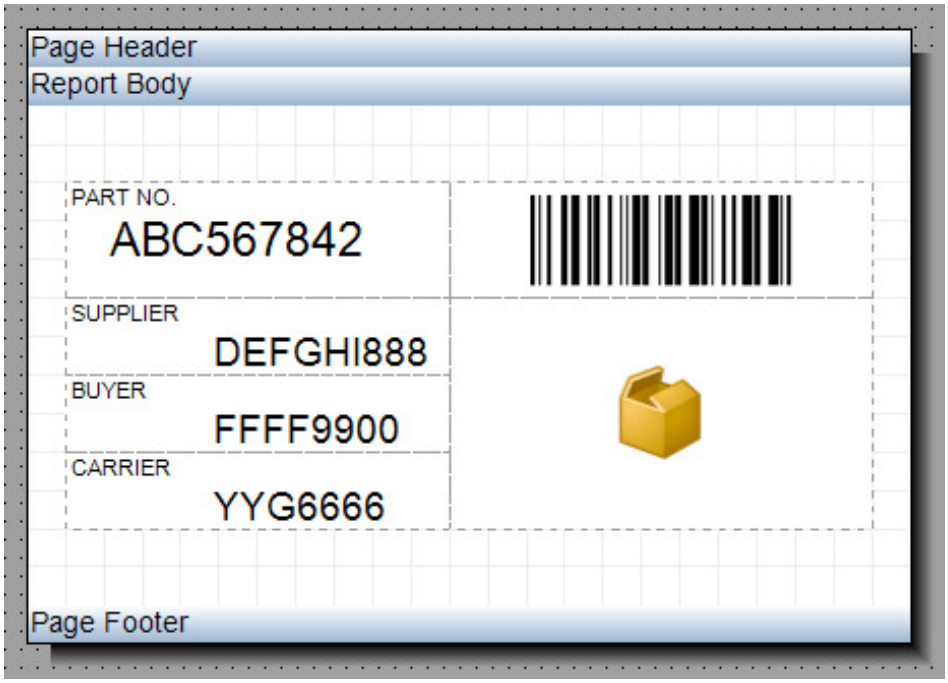
The **CaptionVisible** attribute can be used to show or hide the cell caption in the report instance (as described above for the grid).

The **ImageName** attribute appears if an image is specified against the cell in the report designer. Normally you will choose an image to show in a grid cell by selecting it in the designer, but the **ImageName** attribute allows you to set or override the image to be displayed at map time. Map the name of an image imported into the report definition to this attribute in order to have it rendered to the cell in the report.

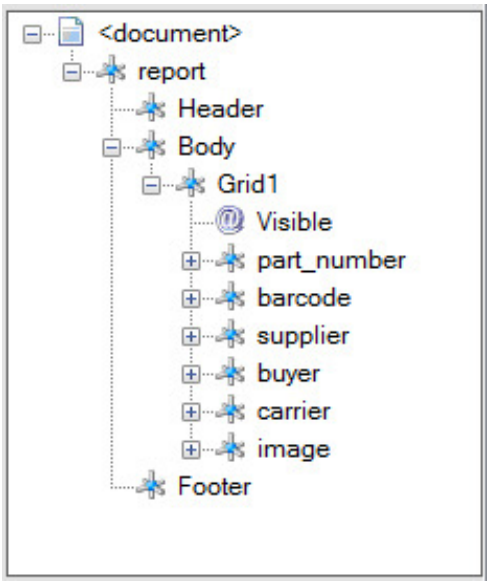
7.3.6 Creating hierarchy using groups

Because the mapper maps report data as if it were mapping to an XML document it needs to map to a hierarchy if certain elements are to repeat. Table rows repeat automatically because the elements shown in the target tree view for a table contain the required hierarchy already (table – body – row – cell).

Where other elements are required to repeat, it is necessary to supply information about the hierarchy to the mapper. This is achieved using the group elements. Consider the following view from the report designer.



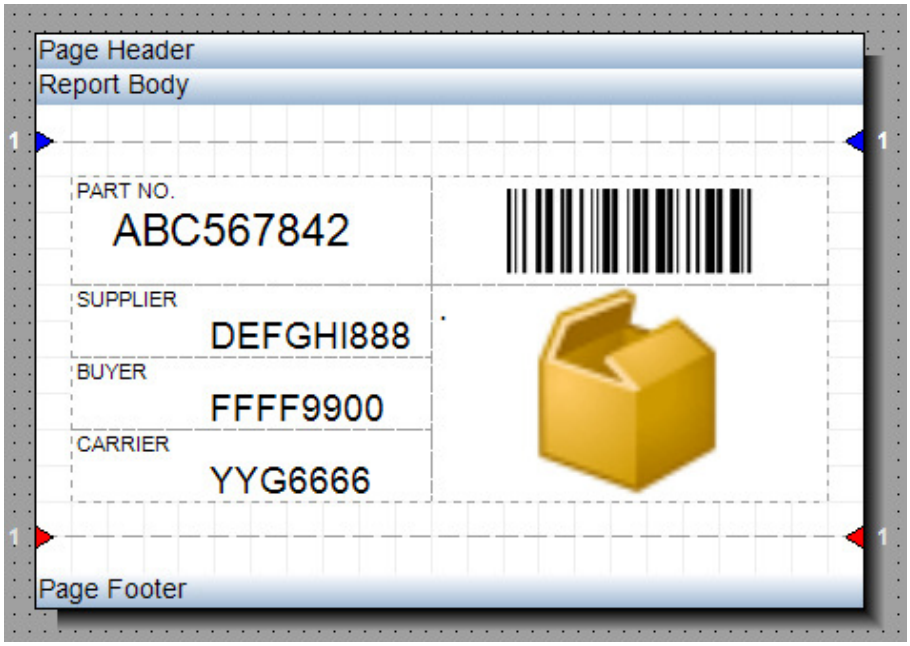
Which translates to the following tree view in the mapper:



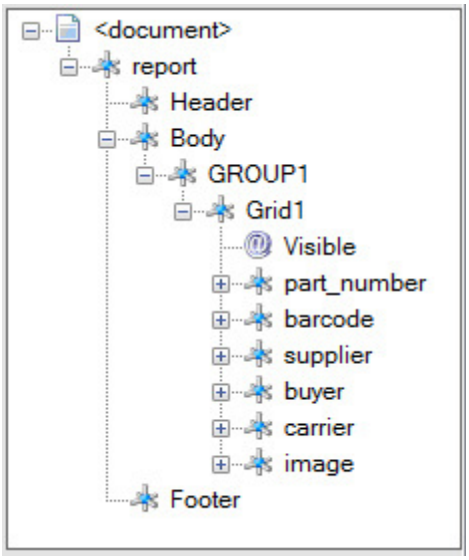
If the source document contains enough data to populate one grid then all is well and a single label will be produced. If however, data are mapped for two or more grids then the result will be repeating body sections in the target data:

```
<document>
  report
    header
    body
      Grid1
        &c
    body
      Grid1
        &c
    body
      Grid1
        &c
    footer
```

Which is not supported by the viewer. If this is what is mapped, only the first instance of the body (and therefore the first instance of the grid) will be rendered to the viewer and/or the printed page. The answer is to use a group which will provide the hierarchy in the target definition needed to permit it to repeat:



Which translates to the following tree view in the mapper:



If the source document contains data for more than one grid the result will be an output like this:

```
<document>
  report
    header
    body
      GROUP1
        Grid1
          &c
      GROUP1
        Grid1
          &c
      GROUP1
        Grid1
          &c
    footer
```

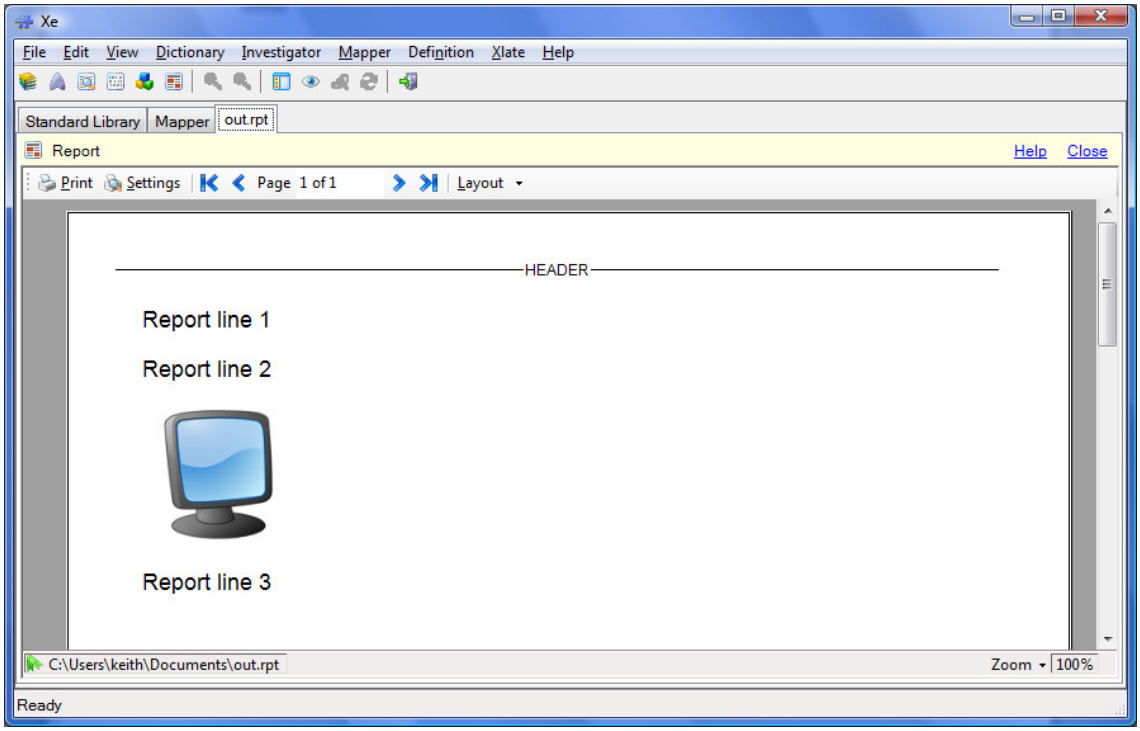
This structure can be rendered correctly by the report viewer. Groups can also be nested to allow any required hierarchy to be built and mapped to.

7.4 Viewing and printing reports

The main Xe GUI includes a report viewer and printer component. When you run Xe maps from the GUI that produce reports, and choose to open the output files, they are opened in the viewer.

7.4.1 Xe report viewer

You can view any report instance file created in the Xe mapper by choosing the menu item **File >> Open** to browse for a report file, which Xe will then open.



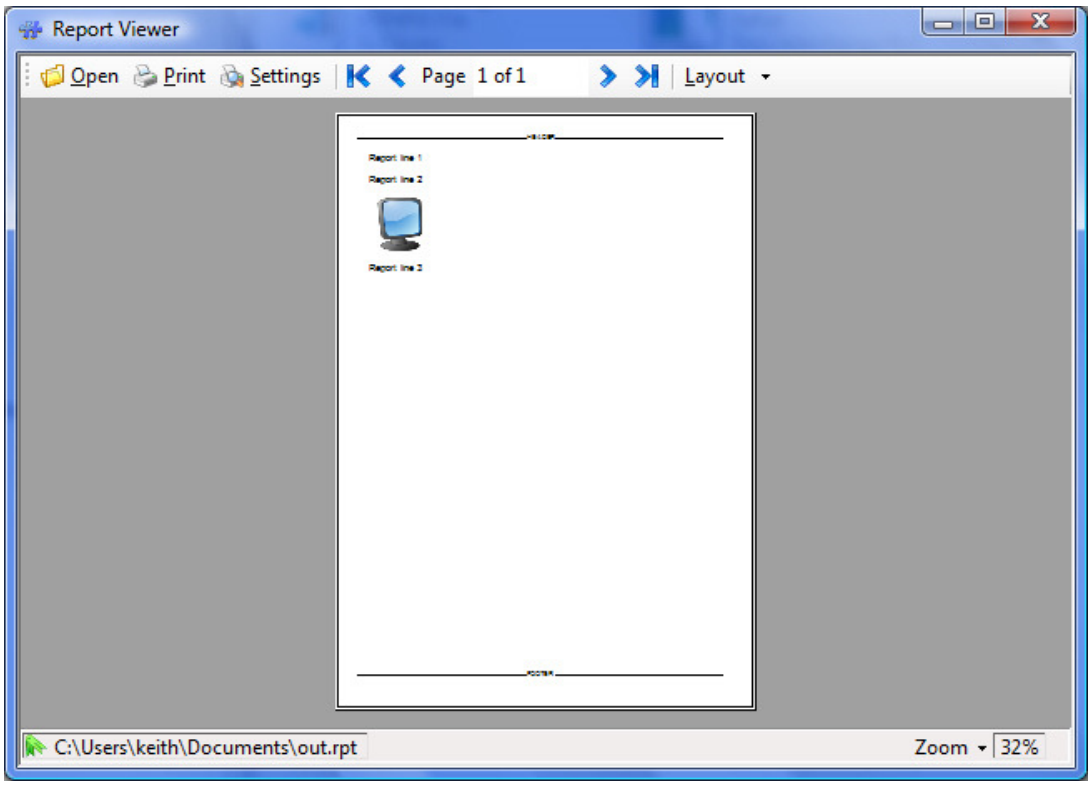
The main features of the report viewer are as follows:

- The filename of the report being displayed is shown in the status bar at the bottom-right of the view.
- The zoom factor is shown in the status bar at the bottom-left of the view. Click the **Zoom** button to show a pop-up menu with options for changing the zoom factor.
- The **Print** button on the toolbar along the top of the viewer window displays a standard print dialog that allows you to choose the printer, pages to print, number of copies and other settings, before printing the report.
- The **Settings** button displays a dialog (Print settings dialog) that can be used to change print settings and redraw the view if necessary.
- There are navigation button and a page number indicator that can be used to scroll through the pages of a multiple page report
- The **Layout** button is used to change the number of report pages shown in the viewer at one time (choose one, two or four pages from the pop-up menu).

7.4.2 Standalone report viewer and printer

If you do not want to use the main Xe GUI for viewing and printing reports, you can use the standalone report viewer and printer application RPT.exe. This application is distributed with Xe and can be found in the directory where Xe

was installed. It is also accessible from the Xe folder in the Start menu, using the **Report Viewer** shortcut.



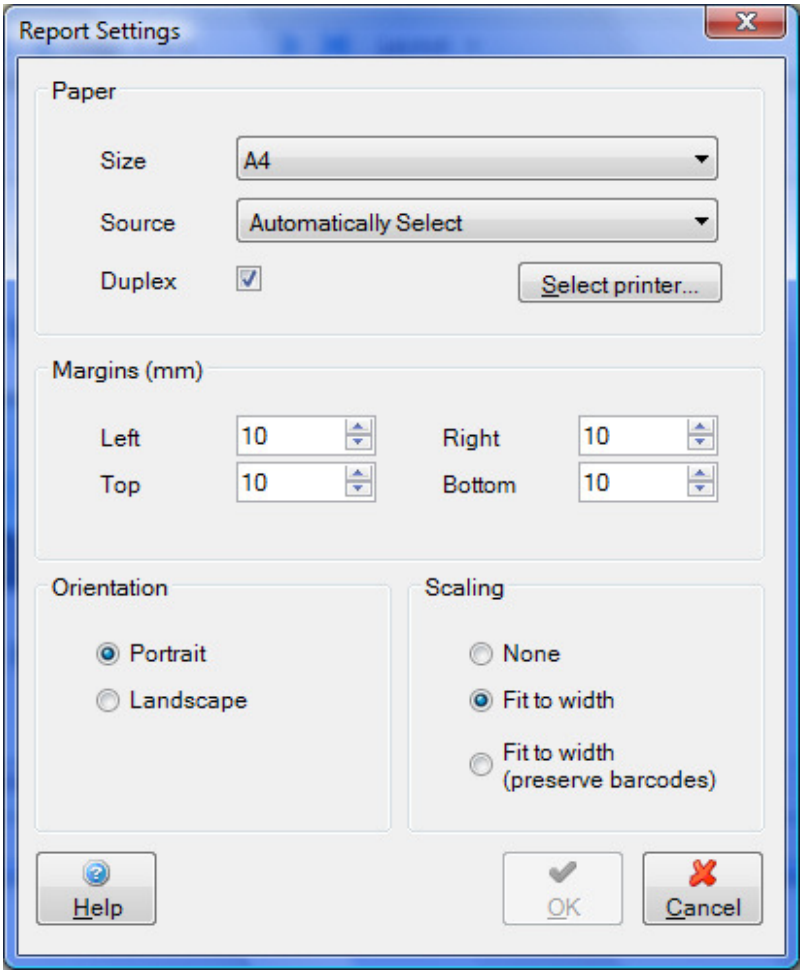
The viewer works in the same way as the viewer built into the main Xe GUI (see Xe report viewer). The only exception is that the toolbar in this application contains an Open button that you use to browse for report files to view.

The application RPT.exe can also be used from the command line. The following commands are available:

```
RPT /?               -  display otions
RPT                  -  show the viewer with no report file
RPT (file)           -  show the viewer with the specified report file
RPT /p (file)        -  silently print the file to the default printer
RPT /p (file) (prn)  -  silently print the file to the named printer
```

7.4.3 Print settings dialog

The print settings dialog is used to set options for rendering the report. The report is laid out for printing on the default printer, using the print settings specified against the report definition in the designer. This dialog is used to override those settings.



Use the **Size** drop-down list to select a paper size from those available on the current printer.

Use the **Source** drop-down list to select a paper source from those available on the current printer.

Check or uncheck the **Duplex** checkbox to indicate whether to use duplexing if supported by the current printer.

Click the **Select Printer** button to a dialog which will allow you to pick another installed printer as the target for the report.

Change the values for the page margins using the **Left**, **Right**, **Top** and **Bottom** margin controls.

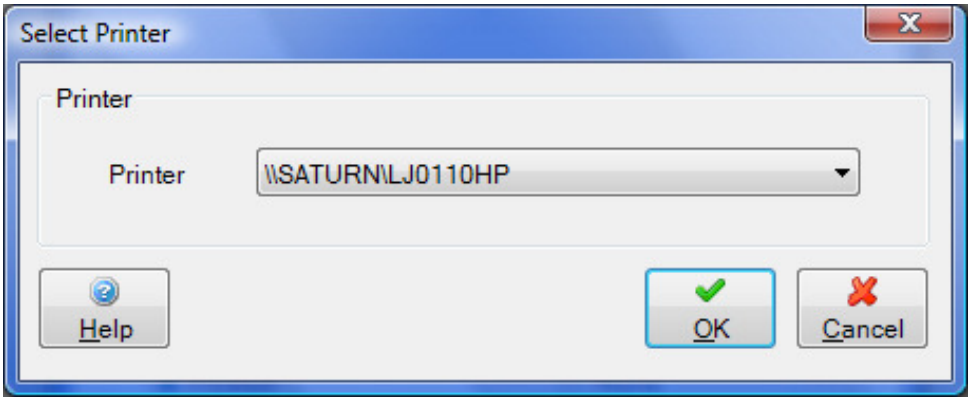
Select a page mode of either **Portrait** or **Landscape** using the buttons in the **Orientation** section.

Override the report's Scaling behaviour by choosing **None**, **Fit to width** or **Fit to width (preserve barcodes)** using the button in the **Scaling** section.

When you are satisfied with the details entered click **OK**, To exit the dialog without saving your changes click **Cancel**.

When settings are changed that affect the way the report is laid out then it will be redrawn in the viewer. For instance, changing paper the size will affect what can be fit on each page of the report and may change the number of pages required to render it.

The printer select dialog is shown below:



Use the **Printer** drop-down list to select a printer from one of those installed on your computer. When you are satisfied with the selection click **OK**, To exit the dialog without changing the printer click **Cancel**.

8 Useful examples

This section contains several useful examples which illustrate how common mapping requirements can be achieved with the Xe Mapper.

8.1 Mapping from EDF to HDF

This section has examples to illustrate mapping from an EDI file (any standard and syntax) to a flat file (including flat files, CSV files and SAP IDocs).

Some of these examples may also be applicable to mapping from an HDF to an EDF.

8.1.1 Map the Interchange Control Reference to every record in the target file.

Simply drag and drop the Interchange Control Reference element (UNB 0070) in the source definition file to each target record field in which it is required.

This creates a one-to-many link.

8.1.2 Control the sequential record number in the target file

Problem – you need to keep a sequential count of each record in the target file and insert the appropriate value in each target record. This issue is often associated with the creation of SAP IDocs.

Solution

Insert a unique count, updated before insertion, in the target node each time the record is created, as follows:

Create a global integer (e.g. int iSegNum) using the DEFN option of the Global code page tab.

Use the INITMAP option to initialise the global integer to zero (e.g. iSegNum = 0)

For each target node that is to contain the counter, highlight the target node, select the Map page tab and configure as follows:

Source type = Fromvar

Source value = iSegNum

Target type = Target node

Target value = <Select name of target node from dropdown list>

Save these settings and select PRE on the Code Snippets page tab.

Type the following code snippet in the space provided:

```
iSegnum++;
```

This will insert a unique count, updated before insertion, in the target node each time the record is created.

8.1.3 Insert the parent record number in the target file

Problem – using the record number generated in the example above, insert the record number of the parent record in each target record. This issue is often associated with the creation of SAP IDocs.

Solution

Store the sequential count of each record until it is required in a child segment, as follows:

Create a global integer (e.g. iSegNum) as described above.

Create a global integer for each segment number level required (e.g. iLastSegnumLevel1)

Use the INITMAP option to initialise each global integer to zero (e.g. iLastSegnumLevel1= 0)

For each target node that is to contain the counter, highlight the target node, select the Map page tab and configure as follows:

Source type = Fromvar

Source value = iLastSegNumLevel1

Target type = Target node

Target value = <Select name of target node from dropdown list>

Save these settings and select PRE on the Code Snippets page tab.

Type the following code snippet in the space provided:

```
iSegnum++; iLastSegnumLevel1 = iSegnum;
```

The effect of this is to store the current value of iSegNum until it is required in a child segment. This means that you can process multiple records at a lower level in the hierarchy and maintain the same value in the global integer until the value of iSegNum changes and is again used to set the value of iLastSegnumLevel1.

8.1.4 Only map data according to the value of another element

Problem – there may be several occurrences of an EDI segment which contain different details depending on the segment qualifier. For example, an EDIFACT NAD segment may contain address details of a Buyer (qualifier = BY) or a Supplier (qualifier = SU). You want to ensure that the appropriate address details are mapped to the correct record according to the qualifier.

Solution

Set up a simple one-to-one mapping using the drag-and-drop technique.

Highlight either the source or target node and select the Conditions page tab.

Against the “Map condition (IF)” caption, type in the code required to set the condition e.g.

```
((LVAL="./PARENT/PRECEDING-
SIBLING[#REF=NAD01010]/#VALUE",CND="EQ",RVAL="BY”))
```

The example code above is testing for a value of BY in the source node with the unique reference ‘NAD01010’ that is the preceding sibling of the parent of the context node.

In this particular case, NAD01010 is the reference of NAD element 3039 (the segment qualifier), so the test ensures that the value is only mapped for the correct qualifier.

For details of how to use XML path values and expressions correctly, please refer to the section entitled “Path values and expressions”.

8.1.5 Change target value depending on source value

Problem – the source file contains a coded value which must be checked to determine what value to map to the target file. For example, the code in the source file may be 1, 2, 3 or 4, where 1 indicates a Firm order and 2, 3 and 4 indicate Forecast orders, but the target file may contain only 1 to indicate a firm

order or 4 to indicate a forecast order. Therefore, if the code is 1 we should map a value of 1, otherwise we should map a value of 4.

Solution

Highlight the source or target node, select the Map page tab and configure as follows:

Source type = Source node

Source value = <Select name of source node from dropdown list>

Target type = Target node

Target value = <Select name of target node from dropdown list>

Map from local var = type in a name for the local variable e.g. strSCC00010Dst

Select the Code variables page tab and add a new string variable e.g. strSCC00010Src.

Save these settings and select PRE on the Code Snippets page tab.

Type the following code snippet in the space provided:

```
if  (strSCC00010Src == "1")
    {
    strSCC00010Dst = "1";
    }
else
    {
    strSCC00010Dst = "4";
    }
```

The effect is as follows:

The source node value is copied into the string variable declared on the Code variables page tab.

The code snippet is performed prior to the mapping, which sets the local variable strSCC00010Dst to be either 4 or 1.

The value of strSCC00010Dst is then mapped to the specified target node.

8.1.6 Format a date correctly for the target file

Problem – the date in the source file is not in the required format for the target file.

Solution

There are 3 parts involved in achieving the required outcome:

- specify the code variable in which to place the data from the source node prior to manipulation
- specify the local variable in which to place the data after manipulation (this is the value that will be mapped).
- provide the code snippet that will perform the actual manipulation.

Highlight the source node from which the data will be mapped, select the Map page tab and configure as follows:

Source type = Source node

Source value = <Select name of source node from dropdown list>

Target type = Target node

Target value = <Select name of target node from dropdown list>

Map from local var = type in the name of a local variable (e.g. strDate). This is the value that will be mapped to the target node after manipulation.

Now select the Code variables page tab.

Click the **New** button and provide the name of the variable (e.g. strDateIn), its type (e.g. string) and the path. In this particular example, we are using the actual source data, so the path will be ./#VALUE to indicate the value of the current node.

Select the Code Snippets page tab and use the dropdown arrow to select PRE (since in this example we are manipulating the data before it is mapped).

Example 1

Transform the source date 20051225 into 051225 for insertion into the target file.

```
strDate = strDateIn.Substring (2, 6);
```

The effect of this code snippet is to take 6 digits from the source date (starting at the 3rd digit – the Substring works with a zero base) and put them into strDate, which is then mapped to the target.

Example 2

Transform the source date 20051225 into 2005/12/25 for insertion into the target file.

```
strDate = strDateIn.Substring (0, 4) + "/" + strDateIn.Substring (4, 2) +
"/" + strDateIn.Substring (6, 2);
```

The effect of this code snippet is to take the three elements of the source date (YYYY, MM and DD) separated by the "/" character, and put them into strDate, which is then mapped to the target.

Example 3

Transform the date 20040923 into 23/09/04 for insertion into the target file.

```
strDate = strDateIn.Substring(6,2) + "/" + strDateIn.Substring(4,2) + "/" +
strDateIn.Substring(2,2) ;
```

The effect of this code snippet is to create a date made up from the three elements of the source date (DD, MM and YY).

8.1.7 Manipulate a date according to the value of a code from the source file

Problem – a date in the target file has to be calculated according to a code in the source file. The code is mapped and stored for later comparison in a PRE code snippet. The calculated date is based on a stored date.

Solution

Map the code and the date in the normal way when they occur.

Store the code and the date as global variables and use the stored code later to determine how to calculate the date, as follows:

Create a global string for the code (e.g. string strPRGRS) and a global string for the date (e.g. strEDATUV) using the DEFN option of the Global code page tab.

Use the INITMAP option to initialise the global string for the code (e.g. strPRGRS = "") and the global string for the date (e.g. strEDATUV = "").

Store the required value of the code (strPRGRS) when you process the node from which it comes. To do this, set up a local variable on the Code Variables page associated with the source node, and use a PRE code snippet to store the

node value in the local variable. N.B. If you are not mapping this node, you will have to set up an ad hoc mapping which does not link to the target file.

Store the required value of the date (strEDATUV) when you process the node from which it comes. To do this, set up a local variable on the Code Variables page associated with the source node, and use a PRE code snippet to store the node value in the local variable. N.B. If you are not mapping this node, you will have to set up an ad hoc mapping which does not link to the target file.

Highlight the relevant target node for the calculated date (not source node as we are not mapping this date from the source file), select the Map page tab and configure as follows:

Source type = Setval (not source node as we are not mapping this date from the source file)

Source value = Type in the name of a local variable e.g. strEDATUB

Target type = Target node

Target value = <Select name of target node from dropdown list>

Map from local var = type in the name of a local variable (e.g. strEDATUB). This is the value that will be mapped to the target node after manipulation.

When you come to process the target date field, check the value of the stored code string and write a PRE code snippet to manipulate the stored date string accordingly e.g.

```
if  (strPRGRS == "D" || strPRGRS == "Y")
    {
    strEDATUB = strEDATUV;
    }
else if (strPRGRS == "W")
    {
// convert CCYYMMDD to date, add 7 days, convert back to CCYYMMDD
    DateTime dtEDATUB = new
DateTime(Convert.ToInt32(strEDATUV.Substring(0,4)),
                         Convert.ToInt32(strEDATUV.Substring(4,2)),

Convert.ToInt32(strEDATUV.Substring(6,2))).AddDays(7);

    strEDATUB = dtEDATUB.ToString("yyyyMMdd");
    }
else if (strPRGRS == "M")
    {
// convert CCYYMMDD to date, add 30 days, convert back to CCYYMMDD
    DateTime dtEDATUB = new
DateTime(Convert.ToInt32(strEDATUV.Substring(0,4)),
                         Convert.ToInt32(strEDATUV.Substring(4,2)),

Convert.ToInt32(strEDATUV.Substring(6,2))).AddDays(30);

    strEDATUB = dtEDATUB.ToString("yyyyMMdd");
    }
```

The code snippet above can be interpreted as follows:

If the value in the global string strPRGRS is D or Y, copy the stored date string strEDATUV into the local variable strEDATUB and map strEDATUB as is without any calculation.

If the value in the global string strPRGRS is W, add 7 days on to the stored date string strEDATUV, copy strEDATUV into the local variable strEDATUB and map the value of strEDATUB.

If the value in the global string is strPRGRS M, add 30 days on to the stored date string strEDATUV, copy strEDATUV into the local variable strEDATUB and map the value of strEDATUB.

8.2 Mapping from HDF to EDF

This section has examples to illustrate mapping from a flat file (including flat files, CSV files and SAP IDocs) to an EDI file (any standard and syntax).

Some of these examples may also be applicable to mapping from an EDF to an HDF.

8.2.1 Insert a fixed value in the EDI file

This is expected to be a common requirement, especially when creating an EDIFACT EDI file, since EDIFACT messages make use of segment qualifiers. These are not normally found in in-house files so are ideal candidates for mapping as fixed values.

Highlight the target node that is to contain the fixed value and configure the Map page tab parameters as follows:

Source type = Value

Source value = type in the required value

Target type = Target node

Target value = <Select name of target node from dropdown list>

8.2.2 Map the Interchange Control Reference to the UNB segment.

There are 2 solutions, depending on whether you can map the Interchange Control Reference from the source file or not.

- Simply drag and drop the Interchange Control Reference element in the source definition file to the Interchange Control Reference (UNB 0070) in the target EDI file.

Or

- Set up a stream entity and a counter entity in the Xe index as shown in the example below.

```
<stream id="str.GM" from="co.Gm" to="co.Me" counter="cnt.IntCtlRef "/>
<counter id="cnt.IntCtlRef" increment="1" />
```

The stream entity must contain a reference to the counter id and a reference to the company for which the Interchange Control Reference is being set.

Once the index has been set up correctly, Xe will maintain the ICR for each trading partner and insert the appropriate value automatically when the EDI file is created.

For further details of how to set these entities up, please refer to the sections entitled “Stream” and “Counter”.

8.2.3 Format a date correctly for the EDI file

Problem – the date in the source file is not in the required format for the EDI file.

Solution

There are 3 parts involved in achieving the required outcome:

- specify the code variable in which to place the data from the source node prior to manipulation
- specify the local variable in which to place the data after manipulation (this is the value that will be mapped). Alternatively, you could use a global variable instead.

- provide the code snippet that will perform the actual manipulation.

Highlight the source node from which the data will be mapped, select the Map page tab and configure as follows:

Source type = Source node

Source value = <Select name of source node from dropdown list>

Target type = Target node

Target value = <Select name of target node from dropdown list>

Map from local var = type in the name of a local variable (e.g. strDate). This is the value that will be mapped to the target node after manipulation.

Now select the Code variables page tab.

Click the **New** button and provide the name of the variable (e.g. strDateIn), its type (e.g. string) and the path. In this particular example, we are using the actual source data, so the path will be ./#VALUE to indicate the value of the current node.

Select the Code Snippets page tab and use the dropdown arrow to select PRE (since in this example we are manipulating the data before it is mapped).

Example 1

Transform the source date 20051225 into 051225 for insertion into the target file.

```
strDate = strDateIn.Substring (2, 6);
```

The effect of this code snippet is to take 6 digits from the source date (starting at the 3rd digit – the Substring works with a zero base) and put them into strDate, which is then mapped to the target.

Example 2

Transform the source date 2005/12/25 into 20051225 for insertion into the target file.

```
strDate = strDateIn.Substring (0, 4) + strDateIn.Substring (5, 2) +
strDateIn.Substring (8, 2);
```

The effect of this code snippet is to take the three elements of the source date (YYYY, MM and DD), and put them into strDate, which is then mapped to the target.

Example 3

Transform the date 23/09/04 into 20040923 for insertion into the target file.

```
strDate = "20" + strDateIn.Substring(6,2) + strDateIn.Substring(3,2) +
strDateIn.Substring(0,2) ;
```

The effect of this code snippet is to create a date made up from a fixed value “20” and the three elements of the source date (YY, MM and DD).

8.2.4 Provide a value in the target if the source is blank

Problem – you are required to provide a value in a certain data element, even though the source field may be blank

Solution

Check the value of the source node to be mapped and provide a value if it is blank, as follows:

Set up one-to-one mapping as usual.

Select the source node from which the data is to be taken and click on the Code variables page tab.

Click the **New** button and provide the name of the variable (e.g. strValueIn), its type (e.g. string) and the path. In this particular example, we are using the actual source data, so the path will be ./#VALUE to indicate the value of the current node.

Select the Map page tab. In the “Map from local var” field, type in the name of a local variable (e.g. strValueOut). This is the value that will be mapped to the target node after manipulation.

Select the Code Snippets page tab and use the dropdown arrow to select PRE (since in this example we are manipulating the data before it is mapped).

Example

Test the value of strValueIn and set the value of strValueOut accordingly.

```
if  ( strValueIn == "" )
{
strValueOut = "NO PREVIOUS";
}
else
{
strValueOut = strValueIn;
}
```

The effect of this code snippet is to test the value of strValueIn. If it is blank, set strValueOut to “`NO PREVIOUS`”. If it is not blank, copy the value of strValueIn to strValueOut, which is then mapped to the target.

8.2.5 Concatenate two or more data items before mapping

Problem – you want to add a prefix and/or suffix to a data item before mapping it to the target

Solution

Set up one-to-one mapping as usual.

Select the source node from which the data is to be taken and click on the Code variables page tab.

Click the **New** button and provide the name of the variable (e.g. `strPIADR`), its type (e.g. string) and the path. In this particular example, we are using the actual source data, so the path will be ./#VALUE to indicate the value of the current node.

Select the Map page tab. In the “Map from local var” field, type in the name of a local variable (e.g. `strPIATEXT`). This is the value that will be mapped to the target node after manipulation.

Select the Code Snippets page tab and use the dropdown arrow to select PRE (since in this example we are manipulating the data before it is mapped).

Example 1

This example concatenates a fixed-text string and a code variable. Note the space that is inserted between the word “`ISSUE`” and the quotation marks, to separate the data from the text.

```
strPIATEXT = "DRAWING ISSUE " + strPIADR;
```

The effect of this code snippet is to prefix the value stored in the code variable `strPIADR` with the fixed-text value “`DRAWING ISSUE `” and store the resulting string in the local variable `strPIATEXT`, which is then mapped to the target.

Example 2

This example concatenates two fixed-text strings and a code variable. Note the space that is inserted between the quotation marks and the word "NEW", to separate the text from the data.

```
strPIATEXT = "DRAWING ISSUE " + strPIADR + " NEW";
```

The effect of this code snippet is to prefix the value stored in the code variable strPIADR with the fixed-text value "DRAWING ISSUE ", append the fixed-text value " NEW" to the value stored in the code variable strPIADR and store the resulting string in the local variable strPIATEXT, which is then mapped to the target.

Example 3

This example concatenates two code variables. Note that a space character must be inserted between the two items to separate them.

```
strPIATEXT = strPIADR + " " + strVariable2;
```

The effect of this code snippet is to concatenate the value stored in the code variable strPIADR with the value stored in the code variable strVariable2, and store the resulting string in the local variable strPIATEXT, which is then mapped to the target.

8.2.6 Only map data according to the value of another element

Problem – there are records in the in-house file which contain a code to indicate what type of information is held in the record. For example, a SAP IDoc record may contain address details of a Buyer (code = AG) or a Supplier (code = LF). You want to ensure that the appropriate address details are mapped to an EDI segment with the appropriate qualifier.

Solution

To map the EDI segment qualifier, configure the Map parameters as follows:

Source type = Value

Source value = "BY" (or whatever is appropriate)

Target type = Target node

Target value = <Select name of target node from dropdown list>

Next set up a simple one-to-one mapping for each address line using the drag-and-drop technique.

For each address line, highlight either the source or target node and select the Conditions page tab.

Against the "Map condition (IF)" caption, type in the code required to set the condition e.g.

```
LVAL="./PARENT/CHILD[#NAME=E1EDKA1-PARVW]/#VALUE",CND="EQ",RVAL="AG"
```

The example code above is testing for a value of AG in the source node with the unique reference 'E1EDKA1-PARVW' that is the child of the parent of the context node. The source node value will only be mapped if this condition is true i.e. if the value of the source node being tested is equal to "AG".

For details of how to use XML path values and expressions correctly, please refer to the section entitled "Path values and expressions".

8.2.7 Insert a sequential number in every occurrence of a segment

Problem – A segment requires a sequential number, unique per message, in every occurrence of the segment.

Solution

Set up a global counter and map it to the target node, as follows:

On the global Counters page tab, define a counter for each segment type that requires a sequential number. Give each one an initial value and an increment value e.g. CPSCOUNTER, initial value 1, increment value 1

Highlight the target node that is to contain the sequential number and open the Map page tab.

Source type = Counter.

Source value = CPSCOUNTER (using our example).

Target type = Target node.

Target value = <Select name of target node from dropdown list>

For more information about Counters, please refer to the section entitled “Counter”.

8.2.8 Store a value for later use

Problem – You want to store a value from the source file so that it can be used later.

Solution

This particular value may be used in one-to-one mapping, but we are not concerned with that here.

Set up a global variable for the value that is to be stored (strPrevCode in our example).

This can be as straightforward as simply defining a global variable and storing the value from the source file in it.

However, our example first compares pairs of dates and then stores the value based on this comparison.

On the Code variables page tab, set up a code variable for each value that is to be compared. Our example has 3 variables: strDateDATUV, strDateABFDE and strDateABMDE.

On the Map page tab, store the appropriate source file value in each code variable.

On the Code snippets page, use the PRE setting to type in the following code:

```
if  (string.Compare(strDateDATUV, strDateABFDE) < 0)
    {
    strPrevCode = "1";
    }
    else if (string.Compare(strDateDATUV,strDateABMDE) < 0)
    {
    strPrevCode = "3";
    }
    else if (string.Compare(strDateDATUV,strDateABMDE) > 0)
     {
      strPrevCode = "4";
     }
else
    {
    strPrevCode = "3";
    }
```

This code compares pairs of dates to see which is the greater and sets the global variable strPrevCode accordingly.

If DATUV is less than ABFDE, the global variable is set to “1”

Else if DATUV is less than ABMDE, the global variable is set to “3”

Else if DATUV is greater than ABMDE, the global variable is set to “4”

Else the global variable is set to “3”

This will result in a value being stored in the designated global variable. It can be referred to whenever it is needed, within the same map.

Please note that the value is subject to change each time the source file hierarchy results in this code being performed.

8.2.9 Map a node only if the value has changed

Problem – You want to map a value which depends on a previously occurring value, but only if that previous value has changed since the last time (in the same map).

Solution

You first need to store the value which is going to be tested to see if it has changed. An example of how to do this is given in the previous mapping example, entitled “Store a value for later use”. We will use the variable “strPrevTgt” for the stored value in this example.

In this example, we are going to store the value of a source node in a code variable (strFrequency in our example). After the source node value has been stored, we will test the value of the code variable and use it to set the value of another global variable.

Highlight the source node from which the value is to be taken and configure the Map page tab as follows:

Source type = source node

Source value = <Select name of source node from dropdown list>

Target type = Readvar

Target value = strFrequency (this is how we define a code variable for use with a node that is not being mapped).

Click the **Save** button, then on the Code snippets page, use the POST setting to type in the following code:

```
if  (strFrequency != "W" && strFrequency != "M")
  {
   strPrevTgt = "Y";
  }
  else
  {
  strPrevTgt = strFrequency;
  }
```

Highlight the target node to which you want to map if the stored value has changed, and configure the Map page tab as follows:

Source type = Setval

Source value = type in a code variable name e.g. strTarget

Target type = Target node

Target value = <Select name of target node from dropdown list>

Click the **Save** button, then on the Conditions page, click the **New change variable** button and select strPrevTgt from the dropdown list that appears alongside the Change variable condition.

On the Code snippets page, use the PRE setting to type in the following code:

```
strTarget = strPrevTgt;
```

9 Command Line Application

The Command Line version of Xe has been included with this release. This enables you to integrate the Mapper component of Xe with ODEX Enterprise or to run it as a stand-alone feature.

The Command Line version allows you to do two things:

- Generate a mapping script
- Generate a DLL from a map and run it to create a transformation

9.1 Map generation

Once tables have been defined (if necessary) and a map script has been created, a mapping DLL can be generated using the command line application `XeCmd`. The command for generating a map DLL is in this format:

```
XeCmd /gen <script_file> <dll_file>
```

Where:

- The `/gen` switch indicates that Xe is generating a map rather than executing one.
- The `<script_file>` parameter specifies the name of a map script file from which the DLL is to be generated.
- The `<dll_file>` parameter specifies the filename of the generated DLL.

9.2 Map execution

Once a map script has been created, a mapping DLL generated and an index put together, the map can be executed using the same Xe command line application that was used to generate the map. XeCmd is invoked to map using this command line syntax:

```
XeCmd /map <config_file> <log_to_screen>
```

Where:

- The `/map` switch indicates that Xe is executing a map rather than generating one.
- The `<config_file>` parameter specifies the name of a configuration file that contains parameters in respect of this invocation.
- The `<log_to_screen>` parameter takes the value `Y` or `N` and specifies whether Xe should log to the console.

9.3 Config file

When running Xe from the command line using XeCmd, a config file is required. The config file includes all the mandatory information Xe requires to be able to execute a map, as well as optional parameters that can be used to alter Xe's behaviour.

9.3.1 Example

The config file uses the same macro file format as the map script. A simple example follows:

```
EXEC LOG-FILE="C:\xe\log.txt",
 LOG=Y,
 LOG-TRACE=Y,
 LOG-TIMESTAMP=DT,
 INDEX-FILE="C:\xe\index.xml",
 DEFINITION-DIR="C:\xe\"

SOURCE SOURCE-FILE="C:\test\in.edi",
 DATA-TYPE="AUTO"

DEST DEST-FILE="C:\test\out.hse"

USER PARAM=(%ABC%,”12345”),
 PARAM=(%DEF%,”54321”)
```

There are four macros in the config file:

- `EXEC` – specifies parameters that configure Xe for the map; the mandatory parameters are:
  - `LOG-FILE`
  - `INDEX-FILE`
  - `DEFINITION-DIR`
- `SOURCE` – specifies parameters relating to the source data ; the mandatory parameters are:
  - `SOURCE-FILE`
- `DEST` – specifies parameters relating to the target data; the mandatory parameters are:
  - `DEST-FILE`
- `USER` – specifies the values of named parameters that may be accessed in the map at runtime.

9.3.2 Parameters

The complete list of parameters in respect of each config macro appears in the table:

Macro	Parameter	Value	Description
EXEC	LOG-FILE	[filename]	Log filename
	LOG-TIMESTAMP	DT (default)	Write date-time in log file
		D	Write date in log file
		T	Write time in log file
		X	Write no date or time in log file
		MS	Write milliseconds in log file
	LOG-TYPE	OVERWRITE	Overwrite existing log file
		APPEND	Append to existing log file
	LOG	Y (default)	Log general messages
		N	Don't log general messages
	LOG-TRACE	N (default)	Don't log trace messages
		Y	Log trace messages
	LOG-EX	Y (default)	Log exceptions
		N	Don't log exceptions
	LOG-ERR	Y (default)	Log recoverable errors

		N	Don't log recoverable errors
	LOG-DATA	N (default)	Don't log loading of source data
		Y	Log loading of source data
	LOG-MAP	N (default)	Don't log map instructions
		Y	Log map instructions
	LOG-EDF	N (default)	Don't loading of EDFs
		Y	Log loading of EDFs
	LOG-HDF	N (default)	Don't loading of HDFs
		Y	Log loading of HDFs
	LOG-IDX	N (default)	Don't loading of index
		Y	Log loading of index
	LOG-TBL	N (default)	Don't log loading of TBLs
		Y	Log loading of TBLs
	LOG-CFG	N (default)	Don't log starting configuration
		Y	Log starting configuration
	LOG-XSD	N (default)	Don't log loading of XML Schemas
		Y	Log loading of XML Schemas
	INDEX-FILE	[filename]	Filename of a valid Xe index
	DEFINITION-DIR	[directory]	Directory where definition files are located
	CONTENT-VALIDATION	Y (default)	Validate data values against EDF or HDF, or perform Schema validation in respect of XML files
		N	Don't perform data validation
	ICR-VALIDATION	N (default)	Don't validate interchange control references
		Y	Attempt to validate interchange control references with respect to a defined trading partner stream and a counter in the index.
	TRIM-EDI	N (default)	Don’t trim whitespace from EDI data elements
		Y	Removes and whitespace from the end of EDI data elements
	ON-ERROR	STOP (default)	Stop on recoverable error
		GO	Continue on recoverable error
	ON-WARNING	GO (default)	Continue on warning
		STOP	Stop on warning
	ERROR-CONFIG	[filename]	Error configuration filename
SOURCE	SOURCE-FILE	[filename]	The source file to be mapped
	HSE-ID	[id]	The ID of a <hse> entity in the index that identifies the source document type

	XML-ID	[id]	The ID of an <xml> entity in the index that identifies the source document type
	IDOC-ID	[id]	The ID of an <idoc> entity in the index that identifies the source document type
	DATA-TYPE	AUTO (default)	Auto-detect source data type
		XML	Source data is XML
		CSV	Source data is CSV
		FLAT	Source data is a flat file
		TRA	Source data is TRA format
		EDIFACT	Source data is EDIFACT
		EDIFACT4	Source data is EDIFACT version 4
		UNGTDI	Source data is UNGTDI (Tradacoms)
		VDA	Source data is VDA
		IDOC	Source data is an IDOC
		X12ISA	Source data is ANSI X.12 (ISA interchange segment)
		X12ICS	Source data is ANSI X.12 (ICS interchange segment)
		X12BG	Source data is ANSI X.12 (BG interchange segment)
	ENCODING	AUTO (default)	Auto-detect source encoding
		ASCII	Source encoding is ASCII/Latin-1
		EBCDIC	Source encoding is EBCDIC
		UNICODE	Source encoding is Unicode
		UNICODE-BE	Source encoding is Unicode big-endian
		UTF-8	Source encoding is Unicode UTF-8
	XML-PROLOG	OPTIONAL	An XML document without the prolog will be treated as valid
		REQUIRED	An XML document without the prolog will be treated as invalid
DEST	DEST-FILE	[filename]	Target filename or mask
	BAT-CONFIG	[id]	The ID of a <batConfig> entity in the index that specifies a batch configuration override to be used in respect of this map
USER	PARAM	[name, value]	The name and value of a parameter that will be available in the map

9.4 Error config file

The config file `EXEC,ERROR-CONFIG` entry allows an error config file to be specified. The error config file can be used to specify what action is taken by Xe when it encounters a given error or warning. Errors and warnings take the form:

```
"DXE0102E"      (error)
"DAT0304W"      (warning)
```

The config file contains a list of one or more such error codes along with the action to take:

```
DXE0102E,GO
DAT0304W,STOP
```

An entry in the error config file overrides the action specified in `ON-ERROR` and `ON-WARNING` in the config file in respect of the specific error code, but only if the error is recoverable (for example, a data content validation error).

9.5 Destination file mask

XeCmd supports the use of a destination file mask, in place of a single filename. This is because the batching capability will sometimes result in more than one target file from a single source. A mask can be specified as follows:

```
"lhs%cnt%rhs.ext"
"lhs%cnt:x%rhs.ext"
```

Where:

- `lhs` represents a fixed part of the mask consisting of an arbitrary string of letters and digits.
- `%cnt%` represents a counter whose value starts at 1 and is incremented by 1 in respect of each file written.
- `%cnt:x%` represents a counter with a width specifier; the numeric part of the generated filename will have a width of at least `x`.
- `rhs` represents a fixed part of the mask consisting of an arbitrary string of letters and digits.
- `ext` represents a fixed file extension.

For example:

```
"A%cnt:5%.xml"
```

Produces this sequence of filenames:

```
A00001.xml
A00002.xml
A00003.xml
...
```

If a mask is not specified and multiple files are produced, Xe will take the filename specified in `DEST,DEST-FILE` and add an incrementing number to it in respect of each new file created.

9.6 Licences

XeCmd requires a licence from Data Interchange Plc. This can be provided in three ways:

- If XeCmd was installed with the Xe GUI, then it will operate correctly so long as the Xe GUI remains licensed.
- If XeCmd was installed with ODEX Enterprise, then it will operate correctly so long as there is a valid licence configured in the ODEX Administrator, and the Xe component is unlocked.

- If XeCmd was installed as a standalone application, the serial number and licence code can be entered or updated from the command line, as follows:

```
XeCmd /code <serial_number> <licence_code>
```

In addition, the current licence information can be viewed using this command line syntax:

```
XeCmd /code
```

10 Appendix A – the Xe Index

10.1 Index Entities

The following sections set out in detail the properties of each index entity, how they are shown in the XML file, and what nesting of XML child elements is permitted. These sections are broken down as follows:

- XML fragment; a fragment of an index file with an example of the use of the element.
- Properties (attributes); properties of the entity that are defined in XML attributes of the XML element.
- Properties (child nodes); properties of the entity that are defined in XML child nodes of the XML element.
- Child entities; other index entities that appear as child nodes of the XML element.

10.1.1 Index

The index entity is the top-level entity in the Xe index, which owns all of the other entities.

10.1.1.1 XML fragment

```
<index>

  <cha ref="chatbl">
  <chn ref="chntbl">
  <chx ref="chxtbl">

  <tbl ref="chatbl" filename="c:\cha.tbl">
  <tbl ref="chntbl" filename="c:\chn.tbl">
  <tbl ref="chxtbl" filename="c:\chx.tbl">
  <tbl ref="CODES" filename="c:\codes.tbl">

  <prm ref="P0010" value="EDICODE999" validate="false">
  <prm ref="P0020" value="1234" validate="true">

  <!-- child entities -->

</index>
```

10.1.1.2 Properties (attributes)

None.

10.1.1.3 Properties (child nodes)

These child nodes are optional; the occurrence column refers to their attributes.

Node	Description	Attribute	Description	Occurrence
cha	CHA table name; corresponds to an alphabetic character validation table	ref	Reference to a TBL node that specifies the table filename	Mandatory
chn	CHN table name; corresponds to a numeric character validation table	ref	Reference to a TBL node that specifies the table filename	Mandatory
chx	CHX table name; corresponds to an alphanumeric character validation table	ref	Reference to a TBL node that specifies the table filename	Mandatory

tbl	TBL node; associates a lookup table or character validation table reference with a table filename	ref	Reference used to refer to the table in EDFs and HDFs	Mandatory
		filename	Table filename to be associated with the reference	Mandatory
prm	PRM node; defines a constant	ref	Reference used to refer to the constant in EDFs and HDFs	Mandatory
		value	Value of the constant	Mandatory
		validate	Validate data against the value of the constant on loading?	Optional (default="false")

10.1.1.4 Child entities

The index element in the XML file has these child entities:

Node	Description	Occurrence
company	A trading partner	Optional
stream	A trading partner stream	Optional
counter	A counter that can be associated with a stream	Optional
edi	An EDI message definition	Optional
hse	A HSE document definition	Optional
xml	An XML document definition	Optional
trans	A transformation	Optional

10.1.2 Company

The company entity represents a trading partner (or the user's company, or a division of it) which has one or more EDI codes.

10.1.2.1 XML fragment

```
<company id="co.Ford">
  <desc text="Descriptive text"/>

  <!-- child entities -->

</company>
```

10.1.2.2 Properties (attributes)

Attribute	Description	Occurrence
id	Unique ID	Mandatory

10.1.2.3 Properties (child nodes)

These child nodes are all optional; the occurrence column refers to their attributes.

Node	Description	Attribute	Description	Occurrence
desc	Attach descriptive text to the entity	text	The text	Mandatory

10.1.2.4 Child entities

The `<company>` element in the XML file has these child entities:

Node	Description	Occurrence
ediCode	Represents a single EDI code entity	One or more

10.1.3 EDI code

The EDI code entity represents information about a single EDI code.

10.1.3.1 XML fragment

```
<ediCode id="co.Ford.PlantAbc">
  <desc text="Descriptive text"/>
</ediCode>
```

10.1.3.2 Properties (attributes)

Attribute	Description	Occurrence
id	Unique ID	Mandatory
code	EDI code	Mandatory
qualifier	EDI code qualifier	Optional (no default)
internalId	Internal ID or (reverse) routing address	Optional (no default)
internalSubId	Internal sub-ID	Optional (no default)
userAppCode	User application code – partner ID for the user's back end system	Optional (no default)

10.1.3.3 Properties (child nodes)

These child nodes are optional; the occurrence column refers to their attributes.

Node	Description	Attribute	Description	Occurrence
desc	Attach descriptive text to the entity	text	The text	Mandatory

10.1.3.4 Child entities

None.

10.1.4 EDI

The EDI entity represents an EDI message definition.

10.1.4.1 XML fragment

```
<edi id="edi.D97Invoice" defFile="INVOICD97.EDF" syntax="Edifact"
  message="INVOIC">
  <desc text="Descriptive text"/>

  <ediDefField ref="UNB-SYNTAX" value="UNOA"/>
  <ediDefField ref="UNH-MESSAGE-VERSION" value="D"/>
  <ediDefField ref="UNH-MESSAGE-RELEASE" value="97A"/>
  <ediDefField ref="UNH-AGENCY" value="UN"/>

</edi>
```

10.1.4.2 Properties (attributes)

Attribute	Description	Occurrence
id	Unique ID	Mandatory
defFile	Definition file (EDF) associated with the message	Mandatory
syntax	Message syntax (EDI variant)	Mandatory
message	Message or transaction set type	Mandatory

10.1.4.3 Properties (child nodes)

These child nodes are optional; the occurrence column refers to their attributes.

Node	Description	Attribute	Description	Occurrence
desc	Attach descriptive text to the entity	text	The text	Mandatory
ediDefField	Zero or more instances of this node specify extra definition fields. If EDI data is to match this definition then the field must have the specified value.	ref	A reference to a field in a service segment	Mandatory
		value	The value that the field is expected to have	Mandatory

10.1.4.4 Child entities

None.

10.1.4.5 Field references

The field references recognised by Xe are listed below. An index that contains any other reference will be rejected on loading.

Service segment	Field ref
Edifact Interchange	UNB-SYNTAX
	UNB-SYNTAX-VERSION
	UNB-SENDER-ID
	UNB-SENDER-ID-QAL
	UNB-REVERSE-ROUTING
	UNB-RECIPIENT-ID
	UNB-RECIPIENT-ID-QAL
	UNB-ROUTING
	UNB-DATE
	UNB-TIME
	UNB-INT-CTRL-REF
	UNB-RECIPIENT-PWD
	UNB-RECIPIENT-PWD-QAL
	UNB-APP-REF
	UNB-PRIORITY
	UNB-ACK-REQ
	UNB-COMM-ID
	UNB-TEST
Edifact Group	UNG-GROUP-ID
	UNG-SENDER-ID-QAL
	UNG-SENDER-ID
	UNG-RECIPIENT-ID-QAL
	UNG-RECIPIENT-ID

	UNG-GROUP-REF-NUM
	UNG-AGENCY
	UNG-MESSAGE-VERSION
	UNG-MESSAGE-RELEASE
	UNG-ASSOCIATION
	UNG-PASSWORD
	UNG-DATE
	UNG-TIME
Edifact Message	UNH-MESSAGE-REF
	UNH-MESSAGE-TYPE
	UNH-MESSAGE-VERSION
	UNH-MESSAGE-RELEASE
	UNH-AGENCY
	UNH-ASSOCIATION
	UNH-COMMON-ACCESS-REF
	UNH-SEQUENCE
	UNH-FIRST-LAST
Edifact V4 Interchange	UNB4-SYNTAX
	UNB4-SYNTAX-VERSION
	UNB4-CODE-LIST-DIR-VER
	UNB4-ENCODING
	UNB4-SENDER-ID
	UNB4-SENDER-ID-QAL
	UNB4-SENDER-INTERNAL-ID
	UNB4-SENDER-INTERNAL-SUBID
	UNB4-RECIPIENT-ID
	UNB4-RECIPIENT-ID-QAL
	UNB4-RECIPIENT-INTERNAL-ID
	UNB4-RECIPIENT-INTERINAL-SUBID
	UNB4-DATE
	UNB4-TIME
	UNB4-INT-CTRL-REF
	UNB4-RECIPIENT-PWD
	UNB4-RECIPIENT-PWD-QAL
	UNB4-APP-REF
	UNB4-PRIORITY
	UNB4-ACK-REQ
	UNB4-AGREE-ID
	UNB4-TEST
Edifact V4 Message	UNH4-MESSAGE-REF
	UNH4-MESSAGE-TYPE

	UNH4-MESSAGE-VERSION
	UNH4-MESSAGE-RELEASE
	UNH4-AGENCY
	UNH4-ASSOCIATION
	UNH4-CODE-LIST-DIR-VER
	UNH4-MSG-TYPE-SUB-FUNC
	UNH4-COMMON-ACCESS-REF
	UNH4-SEQUENCE
	UNH4-FIRST-LAST
	UNH4-MSG-SUBSET-ID
	UNH4-MSG-SUBSET-VERSION
	UNH4-MSG-SUBSET-RELEASE
	UNH4-MSG-SUBSET-AGENCY
	UNH4-MSG-IMPLEMENTATION-ID
	UNH4-MSG-IMPLEMENTATION-VERSION
	UNH4-MSG-IMPLEMENTATION-RELEASE
	UNH4-MSG-IMPLEMENTATION-AGENCY
	UNH4-MSG-SCENARIO-ID
	UNH4-MSG-SCENARIO-VERSION
	UNH4-MSG-SCENARIO-RELEASE
	UNH4-MSG-SCENARIO-AGENCY
X12 ISA Interchange	ISA-AUTH-INFO-QAL
	ISA-AUTH-INFO
	ISA-SECURITY-INFO-QAL
	ISA-SECURITY-INFO
	ISA-SENDER-ID-AGENCY
	ISA-SENDER-ID
	ISA-RECIPIENT-ID-AGENCY
	ISA-RECIPIENT-ID
	ISA-DATE
	ISA-TIME
	ISA-INT-CTRL-VERSION
	ISA-INT-CTRL-NUMBER
	ISA-ACK-REQ
	ISA-USAGE
X12 ICS Interchange	ICS-INT-STANDARD
	ICS-INT-VERSION
	ICS-SENDER-ID-AGENCY
	ICS-SENDER-ID
	ICS-RECIPIENT-ID-AGENCY
	ICS-RECIPIENT-ID

	ICS-DATE
	ICS-TIME
	ICS-INT-CTRL-NUMBER
X12 BG Interchange	BG-COMM-ID
	BG-COMM-PWD
	BG-SENDER-ID
	BG-RECIPIENT-ID
	BG-DATE
	BG-TIME
	BG-TRANS-CTRL-NUMBER
X12 Group	GS-FUNC-ID-CODE
	GS-SENDER-ID
	GS-RECIPIENT-ID
	GS-DATE
	GS-TIME
	GS-GROUP-CTRL-NUMBER
	GS-AGENCY
	GS-VERSION-RELEASE
X12 Message	ST-MESSAGE-TYPE
	ST-TRANS-SET-CTRL-NUMBER
	ST-IMP-CONVENTION-REF
Tradacoms Interchange	STX-SYNTAX
	STX-SYNTAX-VERSION
	STX-SENDER-ID
	STX-SENDER-NAME
	STX-RECIPIENT-ID
	STX-RECIPIENT-NAME
	STX-DATE
	STX-TIME
	STX-SENDER-TRANS-REF
	STX-RECIPIENT-TRANS-REF
	STX-APP-REF
	STX-PRIORITY
Tradacoms Group	BAT-REFERENCE
Tradacoms Message	MHD-MESSAGE-REF
	MHD-MESSAGE-TYPE
	MHD-MESSAGE-VERSION
Vda	VDA-SENDER
	VDA-RECIPIENT

10.1.5 HSE

The HSE entity represents an HSE document definition.

10.1.5.1 XML fragment

```
<hse id="hse.Invoice" syntax="Flat" defFile="INVOIC.HDF">
  <desc text="Description"/>
</hse>
```

10.1.5.2 Properties (attributes)

Attribute	Description	Occurrence
id	Unique ID	Mandatory
defFile	Definition file (HDF) associated with the document	Mandatory
syntax	Document syntax (CSV, Flat, TRA)	Mandatory

10.1.5.3 Properties (child nodes)

These child nodes are optional; the occurrence column refers to their attributes.

Node	Description	Attribute	Description	Occurrence
desc	Attach descriptive text to the entity	text	The text	Mandatory

10.1.5.4 Child entities

None.

10.1.6 Idoc

The Idoc entity represents a SAP IDoc document definition.

10.1.6.1 XML fragment

```
<idoc id="idoc.delfor02" type="DELFOR02" defFile="IDOCDELFOR.HDF">
  <desc text="Description"/>
</idoc>
```

10.1.6.2 Properties (attributes)

Attribute	Description	Occurrence
id	Unique ID	Mandatory
defFile	Definition file (HDF) associated with the document	Mandatory
type	The type of IDoc, appearing in the IDOCTYP field of the EDI_DC40 header (control) record	Mandatory

10.1.6.3 Properties (child nodes)

These child nodes are optional; the occurrence column refers to their attributes.

Node	Description	Attribute	Description	Occurrence
desc	Attach descriptive text to the entity	text	The text	Mandatory
idocDefField	Zero or more instances of this node specify extra definition fields. If an IDoc is to match this definition then the field must have the specified value.	ref	A reference to a field in the IDoc control record EDI_DC40	Mandatory
		value	The value that the field is expected to have	Mandatory

10.1.6.4 Child entities

None.

10.1.6.5 Field references

The field references recognised by Xe are listed as follows. An index that contains any other reference will be rejected on loading:

```
IDOC-CIMTYP
IDOC-MESTYP
IDOC-MESCOD
IDOC-MESFCT
IDOC-STD
IDOC-STDVERS
IDOC-STDMES
IDOC-SNDPRN
IDOC-RCVPRN
```

10.1.7 XML

The XML entity represents an XML document definition.

10.1.7.1 XML fragment

```
<xml id="xml.PurchaseOrder" rootNode="purchaseOrder"
  namespace="http://www/dip.co.uk" defFile="po.xsd">
  <desc text='Description'/>
  <xmlDefField path="//purchaseOrder/shipTo/@code"
    namespace="http://www/dip.co.uk" value="ABC123" />
  <xmlDefField path="//purchaseOrder/billTo/@code"
    namespace="http://www/dip.co.uk" value="DEF456" />
</xml>
```

10.1.7.2 Properties (attributes)

Attribute	Description	Occurrence
id	Unique ID	Mandatory
defFile	Definition file (schema) associated with the document	Mandatory
rootNodeName	Name of XML root node	Mandatory
rootNodeNamespace	Namespace of XML root node	Mandatory

10.1.7.3 Properties (child nodes)

These child nodes are optional; the occurrence column refers to their attributes.

Node	Description	Attribute	Description	Occurrence
desc	Attach descriptive text to the entity	text	The text	Mandatory
xmlDefField	Zero or more instances of this node specify extra definition fields. If XML data is to match this definition then the field must have the specified value and namespace.	path	An absolute path identifying an element or attribute in the XML	Mandatory
		value	The value that the field is expected to have (if 'value' is empty it means that the element or attribute should be present. with any value)	Mandatory
		namespace	The namespace to which the field is expected to belong	Mandatory

10.1.7.4 Child entities

None.

10.1.8 Transformation

The transformation entity represents a single transformation, which may include multiple processes (but uses a single target DOM).

10.1.8.1 XML fragment

```
<trans id="trans.Invoice" docSource="hse.Invoice" from="co.Me"
  to="co.Ford" opConfig="op.Edifact">
  <desc text="Description"/>

  <!-- child entities -->

</trans>
```

10.1.8.2 Properties (attributes)

Attribute	Description	Occurrence
id	Unique ID	Mandatory
docSource	Reference to a source document definition: <edi>, <hse> or <xml>	Mandatory
from	Sender: reference to a company or EDI code entity	Optional (no default)
to	Recipient: reference to a company or EDI code entity	Optional (no default)
opConfig	Reference to an output configuration	Optional (no default)
compoundEdiSrc	Boolean: more than one source EDI document type defined in child processes	Optional (default="false")
doAlways	Boolean: specify that transformation will be performed even if we are in the middle of a compound one	Optional (default="false")
singleTgtDoc	Boolean: Create single target document (multiple processes all add data to a single target document)	Optional (default="false")
fid	Boolean: FID-style processing, composite source document	Optional (default="false")
fidTypeLen	Record type length	Mandatory only if fid="true" and fidFormat is F/A/L
fidTypeStart	Record type start position (or index if format is CSV)	Mandatory only if fid="true" and fidFormat is F/A/L
fidFormat	In-house format (one of F/A/V/X/L/C as on HSE FMT record)	Mandatory only if fid="true"
fidDelim	Record delimiter	Mandatory only if fid="true" and fidFormat is C
fidCsvSep	Separator character	Mandatory only if fid="true" and fidFormat is C
fidCsvQot	Quote character	Mandatory only if fid="true" and fidFormat is C
fidTraRecChar	TRA record delimiter character	Mandatory only if fid="true" and fidFormat is T
fidTraRecEnd	TRA record END flag	Mandatory only if fid="true" and fidFormat is T

10.1.8.3 Properties (child nodes)

None.

10.1.8.4 Child entities

The `<trans>` element in the XML file has these child entities:

Node	Description	Occurrence
proc	Represents a single process	One or more

10.1.9 Proc

The process entity represents a single map within a transformation.

10.1.9.1 XML fragment

```
<proc id="proc.Construct" mapDll="con.dll" mapScript="con.cfg"
  docSource="hse.Invoice" docTarget="edi.D97Invoice">
  <desc text="Description"/>

  <!-- child entities -->

</proc>
```

10.1.9.2 Properties (attributes)

Attribute	Description	Occurrence
id	Unique ID	Optional (no default)
docSource	Reference to a source document definition: <edi>, <hse> or <xml>	Mandatory
docTarget	Reference to a target document definition: <edi>, <hse> or <xml>	Mandatory
opConfig	Reference to an output configuration	Optional (no default)
mapDll	Filename of DLL that will perform the map	Mandatory
mapScript	Filename of mapping script	Optional (no default)
rootNodePath	Used where source is XML, optionally to specify that the map is to be performed multiple times – once for each instance of the given source node, which is to be treated as the root for the purposes of mapping	Optional (no default)
triggerRecord	Used where source is HSE, optionally to specify that the given record type triggers this process – mirrors MHD / FID triggers in Xlate	Optional (no default)
addNextTrigger	When using trigger records with HSE files, specifies that when processing this proc, the trigger record in respect of the next proc is to be added to the DOM before the map on this proc is carried out, AND added to the DOM in respect of the newly triggered proc.	Optional (default="false")

chgTrigger	When using trigger records with HSE files, specifies that the trigger will only operate if the record type has changed. For instance, if the trigger is “REC99” and this flag is set, then the sequence of records “REC88, REC99, REC99, REC00…” will result in a new map being triggered only on the first occurrence of “REC99”	Optional (default="false")

10.1.9.3 Properties (child nodes)

None.

10.1.9.4 Child entities

The `<proc>` element in the XML file has these child entities:

Node	Description	Occurrence
override	Represents a trading partner override on the process (specifying an alternative target document and map)	Zero or more

10.1.10 Override

The override entity represents a trading partner override on a process.

10.1.10.1 XML fragment

```
<override id="ovr.GmInvoice" mapDll="gminv.dll"
  mapScript="gminv.cfg" docTarget="edi.GmD97Invoice" from="Co.Gm">
  <desc text="Description"/>
</override>
```

10.1.10.2 Properties (attributes)

Attribute	Description	Occurrence
id	Unique ID	Optional (no default)
docTarget	Reference to a target document definition: <edi>, <hse> or <xml>	Mandatory
opConfig	Reference to an output configuration	Optional (no default)
mapDll	Filename of DLL that will perform the map	Mandatory
mapScript	Filename of mapping script	Optional (no default)
from	Sender: reference to a company or EDI code entity	Optional (no default)
to	Recipient: reference to a company or EDI code entity	Optional (no default)

10.1.10.3 Properties (child nodes)

None.

10.1.10.4 Child entities

None.

10.1.11 Stream

The stream entity represents a trading partner stream.

10.1.11.1 XML fragment

```
<stream id="str.GM" from="co.Gm" to="co.Me" counter="cnt.Gm"/>
```

10.1.11.2 Properties (attributes)

Attribute	Description	Occurrence
id	Unique ID	Mandatory
from	Sender: reference to a company or EDI code entity	Optional (no default)
To	Recipient: reference to a company or EDI code entity	Optional (no default)
Counter	Reference to a counter entity	Mandatory

10.1.11.3 Properties (child nodes)

These child nodes are optional; the occurrence column refers to their attributes.

Node	Description	Attribute	Description	Occurrence
Desc	Attach descriptive text to the entity	text	The text	Mandatory
Doc	A document definition used to constrain the stream further	ref	Reference to def: <edi>, <hse> or <xml>	Mandatory

10.1.11.4 Child entities

None.

10.1.12 Counter

The counter entity represents a counter used for interchange control reference validation and creation.

10.1.12.1 XML fragment

```
<counter id="cnt.Gm" increment="10" mask="ABC%cnt%" forwardWnd="3"
  reverseWnd="2"/>
```

10.1.12.2 Properties (attributes)

Attribute	Description	Occurrence
Id	Unique ID	Mandatory
Increment	Size of numeric increment	Optional (default="1")
Mask	Specify the format of the counter; include fixed sequences of characters before and/or after the counter value	Optional (default="%cnt%")
forwardWnd	Forward validation window (number of increments over the current counter value that are allowed before a validation error occurs)	Optional (default="0")
reverseWnd	Reverse validation window (number of increments below the current counter	Optional (default="0")

	value that are allowed before a validation error occurs)	

10.1.12.3 Properties (child nodes)

None.

10.1.12.4 Child entities

None.

10.1.13 Output configuration

The output configuration entity represents all of the formatting information that controls how a target document is to be written to file. Note that the characters '<' and '&' are not valid in an XML attribute, so you should use 'LT' and 'AMP' respectively.

10.1.13.1 XML fragment

```
<opConfig id="op.Ungtdi" recSize="72" format="A2" syntax="Ungtdi"
  encoding="Ascii" subElmSep=":" elmSep="+" segCode="=" segEnd="'">
  <desc text="Description"/>
</opConfig>
```

10.1.13.2 Properties (attributes)

Attribute	Description	Occurrence
id	Unique ID	Mandatory
recSize	Record length	Optional (default="72")
format	Format: one of U/F/A1/A2/A3	Optional (default="U")
syntax	Target syntax: one of Edifact, EdifactV4, Ungtdi, X12Isa, X12Ics, X12Bg, Vda, Flat, Csv, Xml	Mandatory
encoding	Encoding of target file: one of Ascii, Ebcdic, Unicode, UnicodeBe, UTF-8	Optional (default="Ascii")
csvSep	CSV separator character	Optional (default=",")
csvQot	CSV quote character	Optional (default="")
csvUseQot	Flag indicating whether to use quotes in CSV output	Optional (default="true")
fillChar	Fill character (when format implies fixed record length)	Optional (default=" ")
elmSep	Element separator (EDI)	Optional (no default)
subElmSep	Sub-element separator (EDI)	Optional (no default)
segCode	Segment code separator (EDI)	Optional (no default)
segEnd	Segment terminator (EDI)	Optional (no default)
rptSep	Repeat separator (EDI)	Optional (no default)
decSep	Decimal separator (EDI)	Optional (no default)
escChar	Escape character (EDI)	Optional (no default)
crlf	Carriage-return / line-feed config: one of CR, LF, CRLF, NONE	Optional (default="CRLF")

trimRecords	Trim trailing spaces from end of flat file records if format is A	Optional (default="true")
gsDate6	Write 6 character date field in the GS segment (instead of the standard 8)	Optional (default="false")
forceUnaSch	Causes the separator segment (UNA or SCH) to be written out in all cases. The default is not to write out those segments unless at least one non-standard separator is in use	Optional (default="false")
traRecChar	Specify the TRA format record character to use (standard form of TRA format records is "/REC-TYPE")	Optional (default="/")
traRecEnd	Specify whether the end of a TRA format records should be indicated by an END field (standard form of TRA format records uses "/END")	Optional (default="true")

10.1.13.3 Properties (child nodes)

Node	Description	Attribute	Description	Occurrence
desc	Attach descriptive text to the entity	text	The text	Mandatory

10.1.13.4 Child entities

None.

10.1.14 Batch configuration

The batch configuration entity represents the batching rules that will be applied in respect of a transformation. Batching rules determine how documents and messages are arranged into files, interchanges and functional groups.

10.1.14.1 XML fragment

```
<batConfig id="bat.Split" fileRule="interchange"
  interchangeRule="message">
  <desc text="Description"/>
</batConfig>
```

10.1.14.2 Properties (attributes)

Attribute	Description	Occurrence
id	Unique ID	Mandatory
fileRule	Specifies the rule used for arranging messages and documents into files	Optional (default = "none")
interchangeRule	Specifies the rule used for arranging messages into interchanges	Optional (default = "none")
groupRule	Specifies the rule used for arranging messages into functional groups	Optional (default = "none")

10.1.14.3 Properties (child nodes)

Node	Description	Attribute	Description	Occurrence
desc	Attach descriptive text to the entity	text	The text	Mandatory

10.1.14.4 Child entities

None.

10.1.14.5 File rules

The permitted file-level batching rules are as follows:

Rule	Description	Applies to
interchange	One file per interchange	Any EDI variant except VDA
senderCo	One file per originator trading partner	Any EDI variant
senderCode	One file per originator EDI code	Any EDI variant
recipientCo	One file per destination trading partner	Any EDI variant
recipientCode	One file per destination EDI code	Any EDI variant
stream	One file per trading partner stream	Any EDI variant
messageType	One file per message type	Any EDI variant
message	One file per message	Any EDI variant
documentType	One file per documentType	Flat or CSV
document	One file per document	Flat or CSV

10.1.14.6 Interchange rules

The permitted interchange-level batching rules are as follows:

Rule	Description	Applies to
message	One interchange per message	Any EDI variant except VDA
messageType	One interchange per message type	Any EDI variant except VDA
none	No interchange-level rule	Any EDI variant except VDA

10.1.14.7 Group rules

The permitted group-level batching rules are as follows:

Rule	Description	Applies to
message	One group per message	Any EDI variant except VDA
messageType	One group per message type	Any EDI variant except VDA
none	No group-level rule	Any EDI variant except VDA

10.2 Examples

The following sections show how the index would be used to set up various scenarios.

10.2.1 EDI variants to single flat file

A user receives different EDI messages from various trading partners, but they are all of the same 'type' (e.g. schedules/orders), and are mapped to the same in house file type. There are two ways of achieving this. The first is to specify a separate document definition for each trading partner:

```
<edi id="Delfor.Gm" syntax="EDIFACT" message="DELFOR" defFile="GMDEL.EDF">
  <ediDefField ref="UNH-MESSAGE-VERSION" value="D"/>
  <ediDefField ref="UNH-MESSAGE-RELEASE" value="97A"/>
</edi>
<edi id="Delfor.Nissan" syntax="EDIFACT" message="DELFOR"
  defFile="NISDEL.EDF">
  <ediDefField ref="UNH-MESSAGE-VERSION" value="D"/>
  <ediDefField ref="UNH-MESSAGE-RELEASE" value="97A"/>
</edi>
<edi id="Delfor.Saab" syntax="EDIFACT" message="DELFOR"
  defFile="DELINV.EDF">
  <ediDefField ref="UNH-MESSAGE-VERSION" value="D"/>
  <ediDefField ref="UNH-MESSAGE-RELEASE" value="96A"/>
</edi>
<hse id="SCHEDULES" syntax="FLAT" defFile="DEL.HDF"/>
<company id="Gm">
  <ediCode code="GM01234"/>
</company>
<company id="Ns"
  <ediCode code=="NS01234"/>
</company>
<company id="Saab">
  <ediCode code="SB01234"/>
</company>
<trans id="Translate.Delfor.Gm" docSource="Delfor.Gm" from="Gm">
  <proc id="Translate.Delfor.Gm" mapDll="del_gm.dll"
    docSource="Delfor.Gm" docTarget="SCHEDULES">
  </proc>
</trans>
<trans id="Translate.Delfor.Nissan" docSource="Delfor.Nissan" from="Ns">
  <proc id="Translate.Delfor.Nissan" mapDll="del_ns.dll"
    docSource="Delfor.Nissan" docTarget="SCHEDULES">
  </proc>
</trans>
<trans id="Translate.Delfor.Saab" docSource="Delfor.Saab" from="Saab">
  <proc id="Translate.Delfor.Saab" mapDll="del_sb.dll"
    docSource="Delfor.Saab" docTarget="SCHEDULES">
  </proc>
</trans>
```

We include three EDI definitions (one for each trading partner), a single in-house definition for our flat file, three trading partners, and three transformations. Each transformation contains one process – the appropriate translation – and is triggered by the sender ID and document type.

The alternative, which is to be preferred (where possible) is to set up the document definitions with no reference to trading partner (in most cases, the TP will use a standard EDI message, that can be represented by a standard EDF – the variation is in how the message is used, and therefore in the map):

```
<edi id="Delfor97" syntax="EDIFACT" message="DELFOR" defFile="DEL97.EDF">
  <ediDefField ref="UNH-MESSAGE-VERSION" value="D"/>
  <ediDefField ref="UNH-MESSAGE-RELEASE" value="97A"/>
</edi>
<edi id="Delfor96" syntax="EDIFACT" message="DELFOR" defFile="DEL96.EDF">
  <ediDefField ref="UNH-MESSAGE-VERSION" value="D"/>
  <ediDefField ref="UNH-MESSAGE-RELEASE" value="96A"/>
</edi>
<hse id="SCHEDULES" syntax="FLAT" defFile="DEL.HDF"/>
<company id="Gm">
  <ediCode code="GM01234"/>
</company>
<company id="Ns"
```

```
  <ediCode code=="NS01234"/>
</company>
<company id="Saab">
  <ediCode code="SB01234"/>
</company>
<trans id="Translate.Delfor.Gm" docSource="Delfor97" from="Gm">
  <proc id="Translate.Delfor.Gm" mapDll="del_gm.dll"
    docSource="Delfor97" docTarget="SCHEDULES">
  </proc>
</trans>
<trans id="Translate.Delfor.Nissan" docSource="Delfor97" from="Ns">
  <proc id="Translate.Delfor.Nissan" mapDll="del_ns.dll"
    docSource="Delfor97" docTarget="SCHEDULES">
  </proc>
</trans>
<trans id="Translate.Delfor.Saab" docSource="Delfor96" from="Saab">
  <proc id="Translate.Delfor.Saab" mapDll="del_sb.dll"
    docSource="Delfor96" docTarget="SCHEDULES">
  </proc>
</trans>
```

10.2.2 Single flat file to EDI variants

The reverse of the above scenario. The user wants to send different invoices to be generated from the same source file:

```
<edi id="Invoic97" syntax="EDIFACT" message="INVOIC" defFile="INV97.EDF">
  <ediDefField ref="UNH-MESSAGE-VERSION" value="D"/>
  <ediDefField ref="UNH-MESSAGE-RELEASE" value="97A"/>
</edi>
<edi id="Invoic96" syntax="EDIFACT" message="INVOIC" defFile="INV96.EDF">
  <ediDefField ref="UNH-MESSAGE-VERSION" value="D"/>
  <ediDefField ref="UNH-MESSAGE-RELEASE" value="96A"/>
</edi>
<hse id="INVOICES" syntax="FLAT" defFile="INV.HDF"/>
<company id="Gm">
  <ediCode code="GM01234"/>
</company>
<company id="Ns"
  <ediCode code=="NS01234"/>
</company>
<company id="Saab">
  <ediCode code="SB01234"/>
</company>
<trans id="Construct.Invoices" docSource="INVOICES">
  <proc id="Construct.Invoices" mapDll="inv.dll" docSource="INVOICES"
    docTarget="Invoic97">
    <override id="Construct.Invoices.Gm" mapDll="gminv.dll" to="Gm"
      docTarget="Invoic97/>
    <override id="Construct.Invoices.Nissan" mapDll="nsinv.dll" to="Ns"
      docTarget="Invoic97/>
    <override id="Construct.Invoices.Saab" mapDll="sbinv.dll" to="Sb"
      docTarget="Invoic96/>
  </proc>
</trans>
```

10.2.3 Different formats from a single source

The user has different in-house systems with different file formats. An EDI file needs to be processed to produce two different target documents for the two in-house systems:

```
<edi id="Delfor97" syntax="EDIFACT" message="DELFOR" defFile="DEL97.EDF">
  <ediDefField ref="UNH-MESSAGE-VERSION" value="D"/>
  <ediDefField ref="UNH-MESSAGE-RELEASE" value="97A"/>
</edi>
<hse id="DELFOR" syntax="Csv" defFile="DEL.HDF"/>
<xml id="DELFOR-XML" rootNodeName="delfor"
  rootNodeNamespace="http://www/dip.co.uk" defFile="del.xsd"/>
<trans id="Translate.Delfor.Csv" docSource="Delfor97">
  <proc id="Translate.Delfor.Csv" mapDll="del_csv.dll"
    docSource="Delfor97" docTarget="DELFOR">
  </proc>
</trans>
<trans id="Translate.Delfor.Xml" docSource="Delfor97">
  <proc id="Translate.Delfor.Xml" mapDll="del_xml.dll"
    docSource="Delfor97" docTarget="DELFOR-XML">
  </proc>
</trans>
```

In this example we define the source EDI document type and the two target types (one CSV and one XML). We also define two transformations; these will be both be performed on the source document; they will be performed in parallel. There are no trading partner overrides.

11 Appendix B

11.1 Code variable types

Code variables can be used in the following areas:

- Global variables page tab
- Code snippets
- Setvar, to specify a type

In any of these places, you may use the following system variable types:

- System.String (String, string) – this is the default for setvar
- System.Boolean (Boolean, bool)
- System.Byte (Byte, byte)
- System.Char (Char, char)
- System.Double (Double, double)
- System.Int16 (Int16, short)
- System.Int32 (Int32, int)
- System.Int64 (Int64, long)
- System.Single (Single, single, float)
- Any of the date formats specified for use with the Sys type system variables

12 Frequently Asked Questions

12.1 How do I provide the source file and target file definitions for the Mapper?

There are several ways to provide the source and target file definitions:

- You can use existing EDF and HDF files if you have any
- You can generate EDF and HDF files from standard EDI message definitions in the Xe Standard Reference Library
- You can edit the standard EDF and HDF files generated from the Xe Standard Reference Library
- You can create EDF and HDF files from scratch (not recommended)

Once you have created definitions for the source file and target file, you can create a new map project which specifies that they are to be used as the source and target definitions.

12.2 How can I maintain a unique Interchange Control Reference for each of my trading partners?

The Xe index can be used to create an Interchange Control Reference (ICR) for outgoing EDI messages and to validate the ICR for incoming EDI messages, each of which is unique per trading partner.

In the Index, for each trading partner you need to create a counter entity with a unique id, an increment value, a forward validation window and a reverse validation window. You can provide a mask value if necessary. An example is shown below.

```
<counter id="cnt.Gm" increment="1" mask="ABC%cnt%" forwardWnd="0"
  reverseWnd="0"/>
```

You also need to create a stream entity in the Xe index and give it a unique id and a reference to the appropriate counter entity. For outgoing messages, you must provide a “to” attribute which contains a cross-reference to either the Company or its EDI code as specified in the index. The “from” attribute is only necessary if it is necessary to distinguish between more than one of your own companies. (To validate the ICR in incoming messages, specify the sending Company or its EDI code in the “from” attribute and omit the “to” attribute unless applicable).

```
<stream id="str.GM" to="co.Gm" counter="cnt.Gm"/>
```

Xe will then maintain each counter and ensure that the next value is inserted when an EDI message is created.

For more information about the format and content of the relevant index entries, please refer to the sections entitled “Counter” and “Stream”.

13 Xlate Evolution Glossary

13.1 Glossary

13.1.1.1.1 Construction

This is an Xlate term. Construction is the term used to describe the transformation from an in-house flat file into an EDI file. Its meaning has been extended in the Xe Mapper to mean the transformation from any type of in-house file (flat, CSV, IDoc etc) into an EDI file.

13.1.1.1.2 Composite elements

This is an EDI term. A data element consisting of 2 or more component data elements is called a composite data element. Each component data element has the same characteristics as a simple data element. Please also see glossary entry for simple elements.

13.1.1.1.3 Data elements

This is an EDI term. Please see definitions for composite elements and simple elements.

13.1.1.1.4 DOM

This stands for Document Object Model. The Xe DOM is used to store in-memory representations of documents and messages, both when loading source data in preparation for mapping, and when mapping to produce a target document. It supports XML documents, EDI messages and in-house data (including SAP IDocs). The Xe DOM is based on the W3C DOM specification for XML, but with some slight differences.

13.1.1.1.5 Escape characters

May also be referred to as release characters. The escape character is provided for use where separator characters have to be transmitted as part of the data in an EDI file. The escape character is used to indicate that the subsequent character is not to be processed as a separator character. The usual escape character is a question mark '?', but may be another character if defined in the UNA segment. Thus the segment FTX+REPL BY C/L 'T'.:SAMPLES DUE KW29.' would have to contain two escape characters, one before each single quote around the T, to read FTX+REPL BY C/L ?'T?'.:SAMPLES DUE KW29.'

13.1.1.1.6 Segments

This is an EDI term. Segments are designed containing logically related data elements and component data elements in a defined sequence. An individual data element (or elements) can only be transmitted within a segment. A segment is the basic unit of data element sets.

13.1.1.1.7 Simple elements

This is an EDI term. A simple data element is a unit of data for which the identification, description and representation have been specified. It contains a single value. It is a basic unit of identifiable and definable information. Please also see glossary entry for composite elements.

13.1.1.1.8 Translation

This is an Xlate term. Translation is the term used to describe the transformation from an EDI file into an in-house flat file. Its meaning has been extended in the Xe Mapper to mean the transformation from any type of file into any other type of file.