



D21.3 Preservation Strategy/RepInfo Toolkit Installation, Deployment and User Manual

Work package WP21 Services/Toolkits Development and Adaptation

Task T21.3, T21.4 Preservation Strategy, RepInfo Toolkits
Development

Author (s) Brian Ritchie, Shirley Crompton STFC

Author (s)

Author (s)

Author (s)

Author (s)

Author (s)

Author (s)

Authorized by

Reviewer Name Surname Company

Doc Id

Dissemination Level CONFIDENTIAL/PUBLIC

Issue 1.0.5

Date 17/06/14


Abstract:

This document represents the Installation, Deployment and User Manual for the Preservation Strategy/RepInfo Toolkit developed by the SCIDIP-ES project. This document contains all relevant information on how to install, configure and use the Toolkit.

Document Log

Date	Author	Changes	Version	Status
04/03/2014	Brian Ritchie	First draft for internal review	0.1	Draft
06/03/2014	Shirley Crompton	Overall edit and re-worked Sections 1 & 2	0.2	Draft
11/03/2014	Brian Ritchie, Shirley Crompton	Updated with editorial changes	1.0	Released
18/03/2014	Brian Ritchie	Updated installation instructions	1.0.2	Draft
17/04/2014	Brian Ritchie	Updated for 1.0.3 release	1.0.3	Draft
09/06/2014	Brian Ritchie	Updated for 1.0.4 release	1.0.4	Draft
17/06/2014	Brian Ritchie	Updated for 1.0.5 release	1.0.5	Released

TABLE OF CONTENTS



SCIDIP-ES

SCIENCE DATA INFRASTRUCTURE FOR PRESERVATION - EARTH SCIENCE

1	INTRODUCTION	1
1.1	PURPOSE AND SCOPE	7
1.2	WHO SHOULD READ THIS DOCUMENT	7
1.3	SYSTEM CONTEXT	7
1.4	LICENSE AND CONDITIONS OF USE	8
2	DESIGN OVERVIEW	9
2.1	DATA MODEL	10
2.2	SERIALIZERS	10
3	INSTALLATION GUIDE	12
3.1	OVERVIEW	12
3.2	PREREQUISITES	12
3.2.1	SOFTWARE PREREQUISITES	12
3.2.2	HARDWARE PREREQUISITES	12
3.3	OSS/COTS INSTALLATION	12
3.4	DOWNLOAD INFORMATION	12
3.5	TOOLKIT INSTALLATION	12
3.6	UNINSTALLATION	13
4	USING THE SCIDIP-ES PRESERVATION STRATEGY TOOLKIT	14
4.1	RUNNING PRESERVATION STRATEGY TOOLKIT	14
4.2	WORKSPACE	14
4.3	THE APPLICATION GUI	14
4.4	FIRST PROJECT	15
4.5	SCHEMA VIEW	17
4.6	FIRST PNM MODEL	19
4.7	EDITING PNM OBJECTS	21
4.8	EDITING RELATIONS	21
4.9	PNM OBJECTIVES	22
4.10	DECISIONS	25
4.11	RISKS	26
4.12	SOURCE AND EXPORTING MODEL	27
4.13	MISCELLANEOUS PNM MODEL EDITOR FEATURES	28
4.14	TRANSFORMATION OBJECTS	29
4.15	WORKING WITH SOLUTIONS	30
4.15.1	INSPECTING THE COST OF A SOLUTION	33

5	USING THE SCIDIP-ES REPINFO TOOLKIT	36
5.1	INTRODUCTION	36
5.2	CREATING A NEW RIN MODEL	40
5.3	SEARCHING FOR REPINFO OBJECTS AND GRAPH LABELS	41
5.4	EXPANDING A CHILD OF A GRAPH LABEL	43
5.5	CREATING NEW REPINFO	44
5.6	EDITING RINS AND RIDOS BY HAND – GUIDELINES	47
5.7	EXAMPLE OF EDITING A RIN	48
5.8	NEW, CHANGED AND UNCHANGED RIDOS	50
5.9	UPDATING AND CREATING IN A REGISTRY - OVERVIEW	53
5.10	UPDATING TO A REGISTRY	54
5.11	CREATING IN A REGISTRY	57
5.12	CLONING A RIDO	59
6	REFERENCE MANUAL	61
6.1	KEYBOARD SHORTCUTS	61
6.2	COMMAND-LINE COMMANDS	61
6.3	PUBLIC APIS	61
7	TROUBLESHOOTING COMMON ISSUES	62
7.1	TOOLBAR MENU ITEMS ARE NOT ENABLED PROPERLY	62
7.2	RIN MODEL: RIDO PROPERTY PANEL MAY NOT BE UPDATED	62
ANNEX A.	REFERENCES	63
ANNEX B.	FIGURES AND TABLES	64
B.1.	LIST OF FIGURES	64
B.2.	LIST OF TABLES	66
ANNEX C.	TERMINOLOGY	67
ANNEX D.	SAMPLE PNM SCHEMA FILE	68
ANNEX E.	SAMPLE RIN SCHEMA FILE	73
ANNEX F.	SAMPLE REGISTRYSERIALIZER.INI FILE	80

1 Introduction

1.1 Purpose and Scope

This document provides an overview of the M30 release of the Preservation Strategy/RepInfo Toolkit focusing in particular to its installation and usage.

1.2 Who should read this document

Users who wish to understand and use the Preservation Strategy/RepInfo Toolkit.

1.3 System Context

The Preservation Strategy Toolkit (PST) provides OAIS¹-conformant functionalities to support preservation planning activities, particularly the crucial tasks of formulating, managing and documenting the strategies adopted in an archive for the long-term preservation and use of its data holding. The toolkit uses the Preservation Network Model (PNM), which was developed by the CASPAR² project to represent the output of a preservation analysis conducted for a digital object to be preserved in a preservation archive or repository [CON1].

In an OAIS-compliant archive, there are a number approaches to preserve digital data. For example, besides using Representation Information³ (RepInfo; which includes emulation software), one could migrate (Transform in OAIS terminology) the data into different formats. The key objective of a preservation analysis is to systemically identify the additional information objects that must also be preserved to ensure the long term usability of the digital data by one or more target user communities and to quantify the associated preservation risks taking into account the preservation aims, related risk tolerance level, preservation policies and other archive specific requirements. PNM captures the semantics of the analysis as a network dependency graph. The data object being preserved is described as having a dependent relationship on other information objects and their relationships are qualified by the preservation functions (e.g. render, transform, etc.) as well as the decisions made by the archivist on how to manage the relationships. A basic PNM example for BADC⁴ Mesosphere-Stratosphere-Troposphere⁵ (MST) data in netCDF⁶ Cartesian format is given Figure 1. The PNM illustrates that in order to preserve and maintain the usability of the MST data for the long term, it is also necessary to preserve additional information objects such as *Directory Structure*, which explains the directory structure of the archived MST data.

¹ OAIS - <http://public.ccsds.org/publications/archive/650x0m2.pdf>

² Caspar project - <http://www.casparpreserves.eu/>

³ Representation Information is defined in OAIS as the additional information that maps a data object into more meaningful concepts that facilitate the interpretability of the data object.

⁴ British Atmospheric Data Centre - <http://badc.nerc.ac.uk/home/index.html>

⁵ MST data - http://badc.nerc.ac.uk/view/badc.nerc.ac.uk__ATOM__dataent_MST

⁶ Network Common Data Format - http://mst.nerc.ac.uk/file_format_netcdf.html

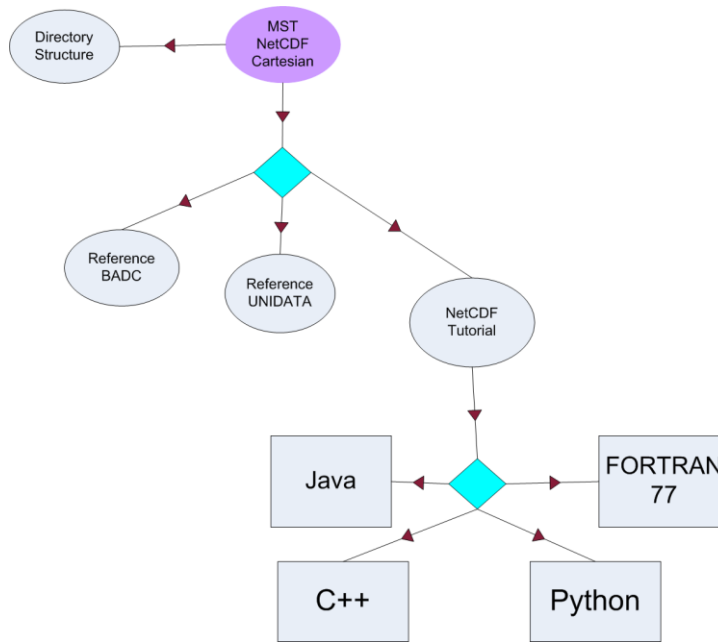


Figure 1 Example PNM for BADC MST data.

The Representation Information Toolkit (RIT), in contrast, supports preservation activities related to the implementation and operational stages of the archive after a suitable preservation strategy is achieved. PNM provides a sharable, stable and organized structure for capturing digital objects and their requirements. The PNM entities, i.e. PNM objects and PNM relationships, can be packaged and stored within an OAIS-compliant Archival Information Package (AIP) for easy discovery, access and use within an OAIS-aware preservation archive. RIT provides functionalities (see Section 5) for the creation and maintenance of these AIPs organised as a virtual RepInfo Network (RIN) stored within one or more RepInfo Registries within the SCIDIP-ES e-infrastructure, as well as the creation of new RepInfo objects using relevant SCIDIP-ES software components, e.g. Data Virtualization Toolkit, and other external tools.

1.4 License and Conditions of Use

The SCIDIP-ES Preservation Strategy/RepInfo Toolkit is licensed under the Apache License, Version 2.0 (the "License"). You may not use this software except in compliance with the License. A copy of the License could be obtained at: <http://www.apache.org/licenses/LICENSE-2.0>. Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

2 Design Overview

The Preservation Strategy/RepInfo Toolkit is a user friendly GUI developed on the Eclipse Rich Client Platform⁷ (RCP). Users can use the tool to visually design PNM/RINs by dragging and dropping graphical objects onto the drawing canvas and document their textual properties via simple form-based editors. Eclipse has an open architecture and its plugin framework provides a large number of extensible plugin modules that can be used to customise the basic GUI workspace provided by the Eclipse IDE to deliver feature-rich GUI applications that have a common look and feel across multiple platforms. The Eclipse plugin approach minimises development efforts as developers can exploit common functionalities provided by existing plugins and need only implement the missing application-specific logic either by extending existing plugins or implementing new ones.

In line with the plugin philosophy of sharing functionalities, PS and RIT are implemented as different perspectives of a single multi-module application. (Perspectives are a way to group Eclipse views and commands for a particular task.) The two perspectives offer specific functionalities for working with PNMs and RINs, both of which are essentially a network dependency graph, albeit with different properties and behaviour (see Section 1.3). By merging the two toolkits, we are able to optimise the development by sharing common capabilities, including GUI support for building network graphs and capturing textural properties, the mechanism of using pluggable Serializers to interact with persistent resources as well as the declarative approach of using external schemas to drive application behaviour (see Figure 2 and Section 2.1).

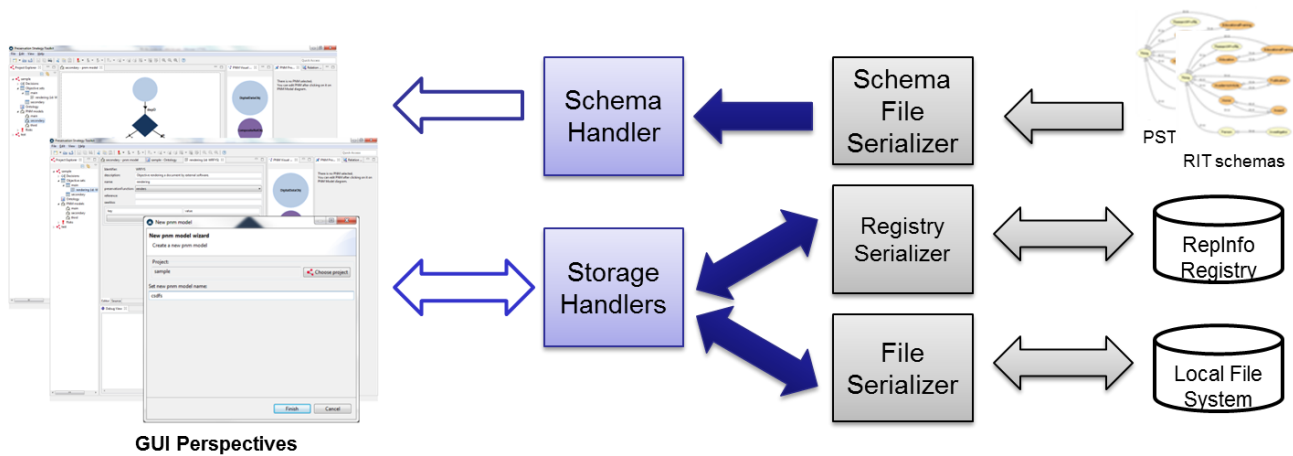


Figure 2 Overview of PS/RIT design.

PS/RIT application is built as a Maven⁸ project comprising the following modules (each of which is a separate Eclipse plugin):

- eu.scidipes.impl.pst - main part of the application containing all editors, handlers and core business logic;

⁷ Rich Client Platform - http://wiki.eclipse.org/Rich_Client_Platform

⁸ Apache Maven - <http://maven.apache.org/>

- eu.scidipes.impl.pst.serializers – this contains a collection of serializer interfaces used to abstract interactions with external resources, such as the file system and SCIDIP-ES RepInfo Registry via the SCIDIP-ES Framework component library;
- eu.scidipes.impl.pst.logger – provides a logging system and Debug View to the main application;
- eu.scidipes.impl.pst.serializers.simple – default serializers implementation provided by PS/RIT. Developers can replace this with their own custom implementation for interacting with specific resources;
- eu.scidipes.impl.pst.parent – contains the parent Maven POM used by the Tycho⁹ Maven plugin for building PS/RIT;
- eu.scidipes.impl.pst.repo - a placeholder for the PS/RIT products targeting different OS built by Tycho.

2.1 Data Model

The Preservation Strategy Toolkit uses Eclipse's Workspace and Project features. Each project is mapped to an instance of *ProjectModel* (see Figure 3), which is a utility class for managing projects.

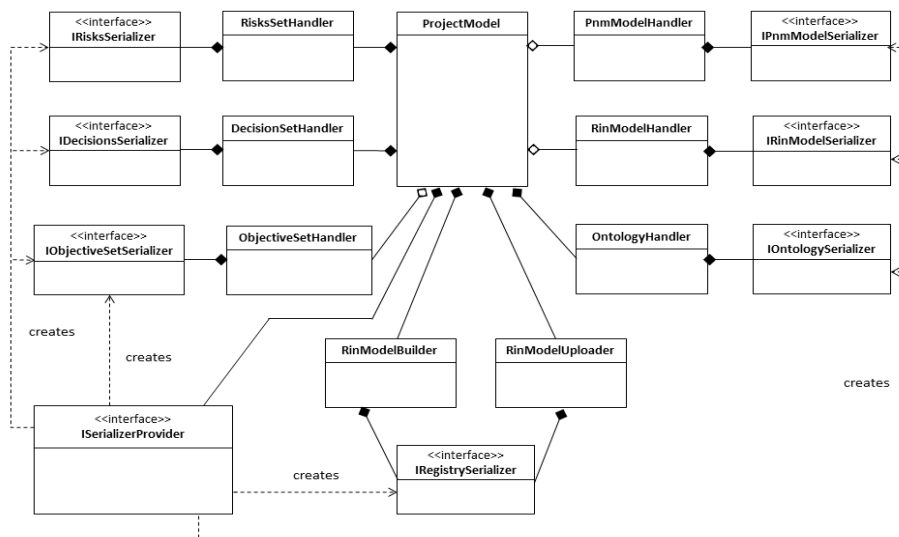


Figure 3 PS/RIT project model.

As described above, much of the behavior of the application is driven by external schemas, generic POJOs and standard Java objects such as HashMap are used to encapsulate application information. For examples, HashMaps are used in the GUI editors to store information at runtime and users can enter arbitrary information, in addition to the schema defined attributes, by adding them as key-value pairs (see Section 4.7).

2.2 Serializers

⁹ Tycho - <https://eclipse.org/tycho/>

The *Serializer* interfaces provide a layer of abstraction to decouple interactions between the application and backend resources. This design enables a declarative approach of using externally defined schemas to drive the runtime behaviour of the application. The schema defines the PNM/RIN object types, their attributes and relationship dependencies. When a new preservation project is created (see Section 4.4), the external schema files are specified and loaded; the schema attributes are mapped to POJO objects while the operations are provided by concrete serializer implementations (see Figure 2). The design allows individual repositories to customize the application by providing their own PNM or RIN information model that maps to an archive-specific business model and/or RepInfo Registry implementation.

PS/RIT uses Java Service Provider Interface¹⁰ and Eclipse Extension Points¹¹ mechanisms to enable the creation and runtime plugin of concrete serializers implementations. Handlers (see Figure 3) are used to further decouple the serializer implementations from the main application and to add standard utility methods. When a project is created, the user specifies the *ISerializerProvider* implementation to use (see Section 4.4). *ISerializerProvider* is a factory class which is used by the application to instantiate all the defined serializers. PS/RIT currently provides a single implementation, *SimpleSerializerProvider* and a set of simple, default serializer implementations for demonstration purposes. These are intended to be replaced by custom implementations by the archives to deliver archive-specific behaviours.

¹⁰ Java Service Provider Interface (SPI) - <http://docs.oracle.com/javase/tutorial/sound/SPI-intro.html>

¹¹ Eclipse Extension Point - <http://www.vogella.com/tutorials/EclipseExtensionPoint/article.html>

3 Installation Guide

3.1 Overview

The Preservation Strategy/ReplInfo Toolkit is developed as an Eclipse plug-in (see Section 2) and the product is packaged as a self-contained executable application.

3.2 Prerequisites

3.2.1 Software prerequisites

Software prerequisites respect SCIDIP-ES guidelines and include:

- Java Runtime Environment 1.7 or higher

The Eclipse RCP framework is platform-independent, but depends on platform-specific libraries of its lower-level modules, (including the Standard Widget Toolkit (SWT)). The PST/RIT is available for Windows 7, Windows XP and Mac OSX. However, a Linux version is not provided due to graphics issues.

3.2.2 Hardware prerequisites

None.

3.3 OSS/COTS Installation

None.

3.4 Download information

The most recent stable source code can be accessed from SVN at *Sourceforge*. The URL to the svn trunk is:

```
<svn://svn.code.sf.net/p/digitalpreserve/code/SCIDIP-  
ES/software/toolkits/PreservationStrategyToolkit/trunk>
```

The current release of the PST/RIT can be downloaded from the SCIDIP-ES Interactive Platform at <http://int-platform.scidip-es.eu/joomla/>.

The most recent release of the software may also be downloaded from the SCIDIP-ES maven nexus repository at:

```
<http://nexus.scidip-  
es.eu/content/repositories/releases/eu/scidipes/toolkits/perservationstrategytoolkit/>.
```

The platform-specific product downloads can be found in the subfolder <pst-distribution/1.0.5>, e.g. pst-distribution-1.0.5-win64.zip.

3.5 Toolkit Installation

Download and unzip the archive file `pst-distribution-<your platform>.zip` to create a folder hierarchy. The top level includes another zip file, with a name of the form `pst.product-<platform>.zip`; unzip this into the same directory. This unpacks an executable file ('Preservation Strategy Toolkit.exe' in Windows) that launches the combined Toolkit.

The distribution includes a sample workspace with a demonstration preservation project and a schemas folder that includes the PNM and RI schema files required for the creation of new projects.

The distribution also contains several configuration files; please see `README.txt` in the distribution for more details.

The RepInfo Toolkit requires a network connection to communicate with external RepInfo Registries. If your network uses a HTTP proxy, you need to:

- create a small configuration file, with the default name *registryserializer.ini*, to define the proxy settings. A sample *registryserializer.ini* file is included in Annex F and in the distribution zip;
- If the file is not in the same folder as the PST executable, then set an environment variable named *PST_HOME* to point to the folder containing the file.

3.6 Uninstallation

The PST can be uninstalled by deleting the folder hierarchy obtained from the archive file as described in Section 3.5. Users may optionally delete the project files that they have created. These are stored in a 'workspace' folder specified by the user on application start up.

4 Using the SCIDIP-ES Preservation Strategy Toolkit

This section gives a brief tutorial on how to use the Preservation Strategy Toolkit (PST).

4.1 Running Preservation Strategy Toolkit

After unzipping the archive, simply run the 'Preservation Strategy Toolkit' executable file located in the unzip folder.

4.2 Workspace

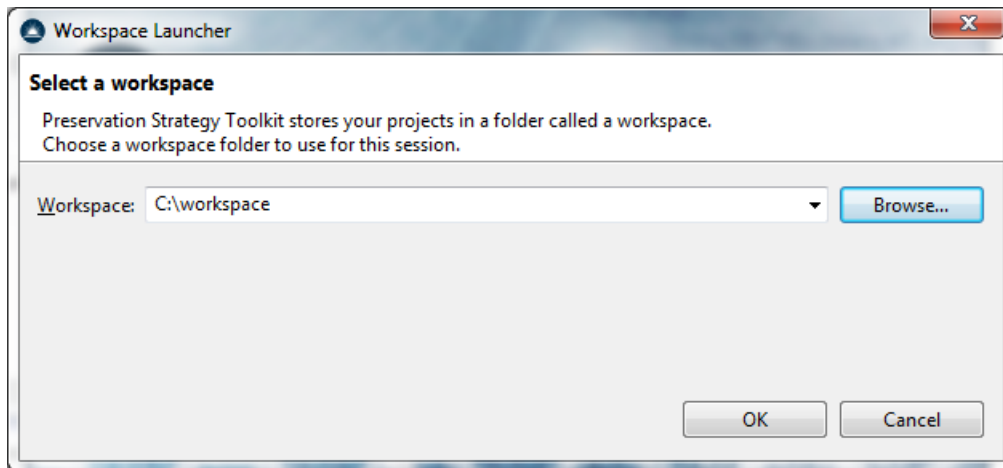


Figure 4 Defining a PST workspace.

After launching the application, a pop-up dialog prompts user for a workspace location (Figure 4). The workspace is a folder where projects created by users within an application session are stored. This dialog appears every time on launching the application and it is possible to use multiple workspaces. A sample workspace containing a simple project is contained in the distribution.

Users are advised not to edit files inside the workspace directory manually and only use Preservation Strategy Toolkit to change data in it.

4.3 The Application GUI

Figure 5 shows the main components of the application GUI.

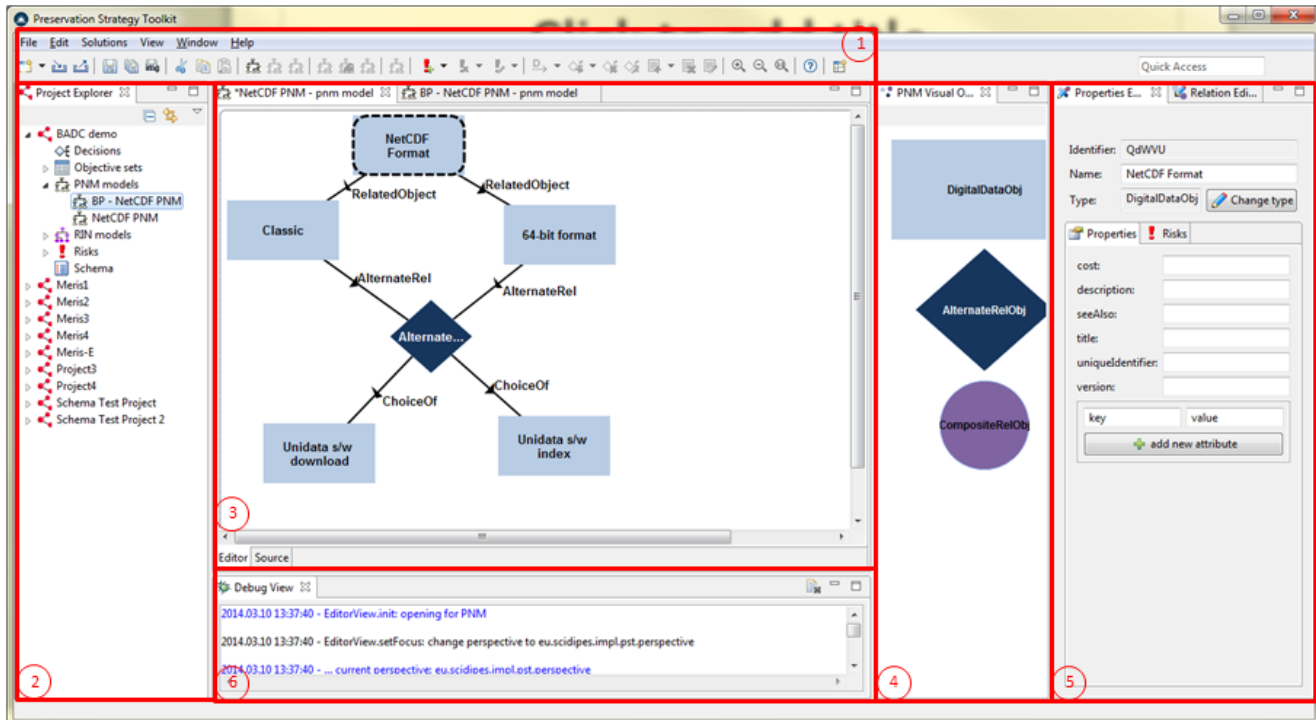


Figure 5 Main GUI components – PST perspective.

1. Toolbar menus and buttons
2. Project explorer: a hierarchical view of each Preservation Project in the workspace
3. Main editor pane. The tabs along the top show currently-open project components
4. Visual objects toolbox: list of graphical objects that can be dragged onto graphical editors (for PNM and RINs)
5. Property editors. Properties for the currently-selected object or relation (if any) are shown and can be edited here.
6. Debug View. Shows logging messages.

4.4 First Project

A preservation project may be created in several ways:

- in main menu by clicking on file -> new -> Preservation Project (Figure 6)

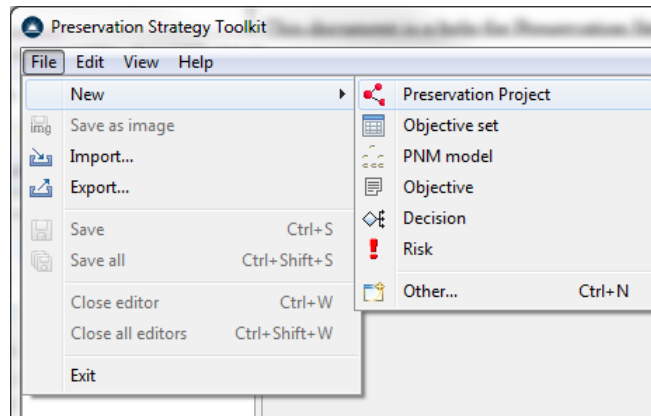


Figure 6 Create a new preservation project using the main menu.

- by clicking on the first toolbar's icon just above file menu and selecting Preservation Project (Figure 7)

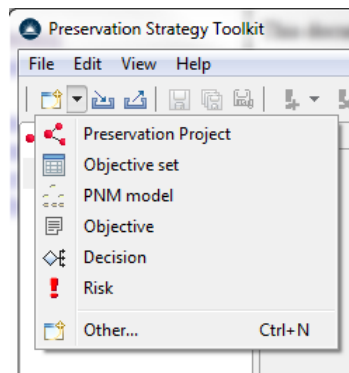


Figure 7 Create a preservation project using the toolbar icon.

- using the right mouse click context menu on the Project Explorer view which is on the left side of the application (Figure 8)

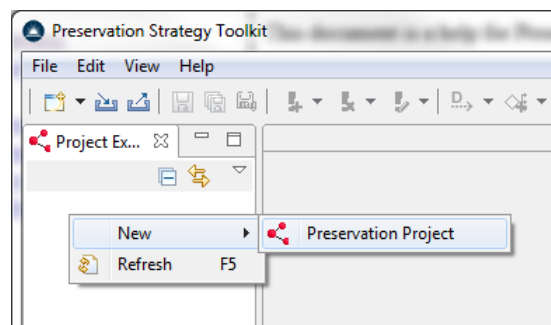


Figure 8 Create a new preservation project using the context menu.

The New Project dialog (Figure 9) appears. Set the project name and specify the Serializer Provider and schemas to use in the new project. Users can provide their own custom schema file, or use the provided default schema file.

The RepInfo Toolkit uses a separate schema file. An example is provided in Annex E and a copy is available in the schemas folder in the distribution zip.

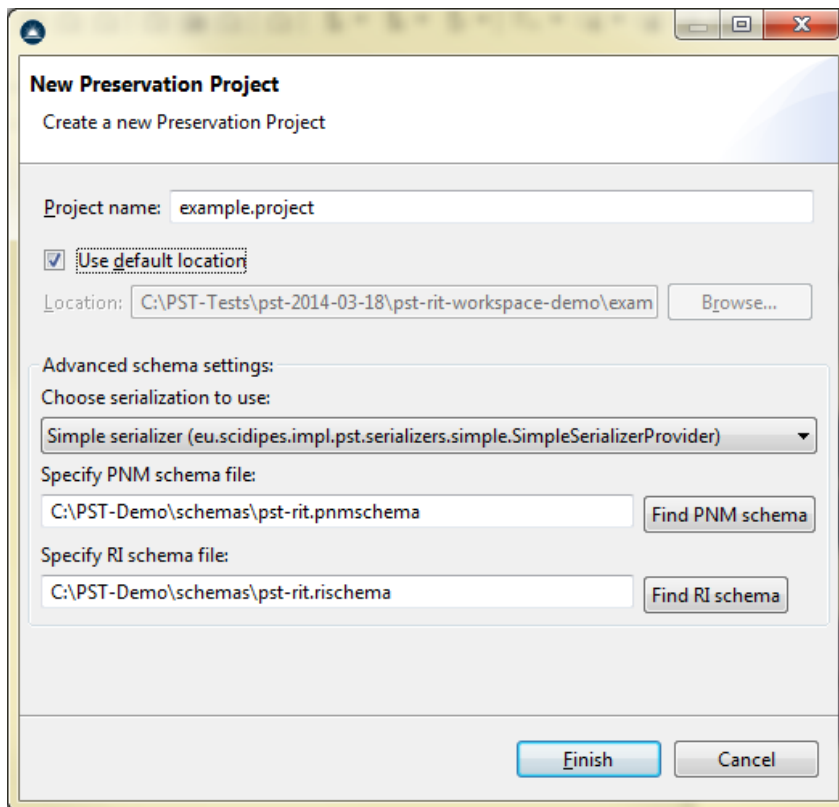


Figure 9 New project dialogue.

After clicking on finish button you will notice that your project was added to the Project Explorer view on the left side of the application.

4.5 Schema View

Double click on the Schema in the Project Explorer (Figure 10) to open the Schema Viewer.

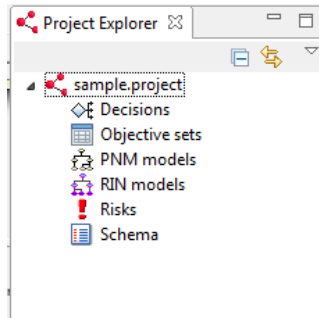


Figure 10 Project explorer.

The schema cannot be modified, but the editor provides a useful view to inspect it.

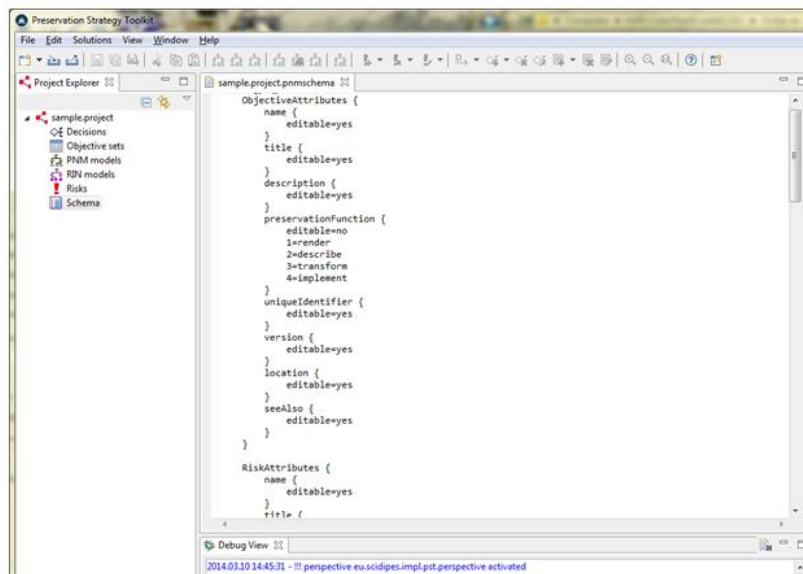


Figure 11 Schema viewer.

The view shows the source of the schema used for PNM models. The main sections are:

- Objective attributes – shows default attributes which are automatically set for every new objective created in the project. There are eight attributes in the default schema and most of them only have the keys predefined. The 'preservationFunction' drop down list shows a list of preset preservation function.
- Risk attributes - shows attributes which are automatically set for every new risk created in the project.
- Decision attributes – shows default attributes which are automatically set for every new decision created in the project.
- PNM object properties - shows a list of default attributes which are automatically added to every new PNM object.
- Dependencies or Relations - shows a list of dependencies which may be used while creating relations between PNM objects. The default Schema restricts the relationships permitted between different types of PNM objects.

4.6 First PNM Model

Create a new PNM model using one of three ways as described in Section 4.4 for creating a new project.

In the new PNM Model dialog (Figure 12), provide the model name. The host preservation project should be automatically selected; if not use the 'Choose project' button and select it.

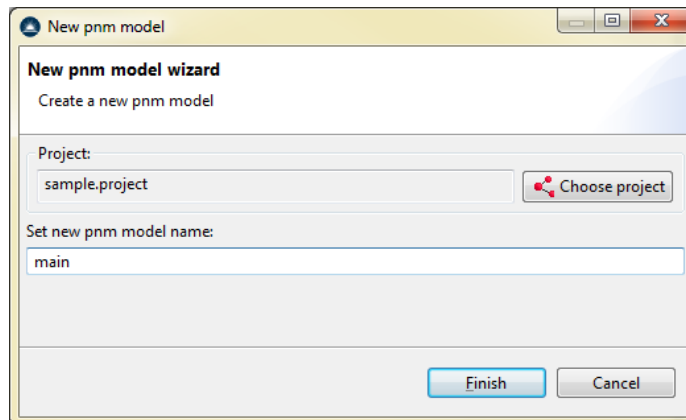


Figure 12 New PNM Model dialog.

After creating a new model, the PNM model editor associated with the PST perspective (**Error! eference source not found.**) should open in the centre of the Preservation Strategy Toolkit. If not, you can open it by double-clicking on the model listed in the Project Explorer.

The centre panel of the application should look like this:

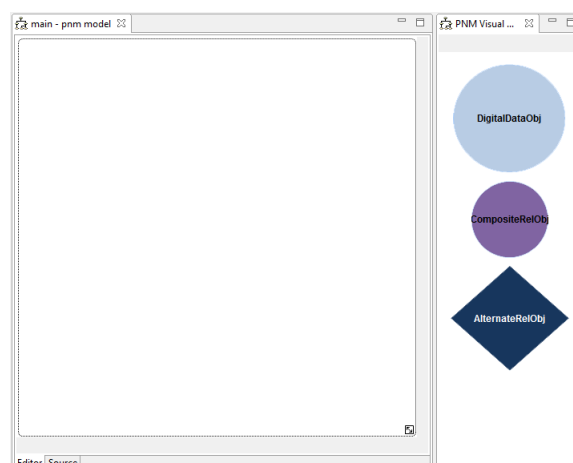


Figure 13 The PNM drawing canvas.

The PNM model editor is a graphical editor which is a drawing canvas (Figure 13) in which you design your preservation network model graph. You can add new PNM objects by dragging it from the PNM Visual view to the right of the canvas. Figure 14 shows the canvas with two PNM objects.

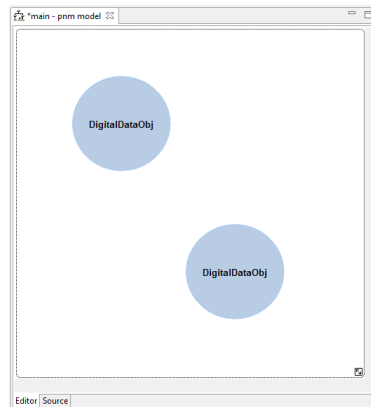


Figure 14 PNM drawing canvas with two PNM objects.

You can move objects by dragging them on the canvas. If you drag an object outside the canvas, the canvas will automatically expand. You can also use the small arrow in the bottom right hand corner to change the canvas size. To remove a PNM object just use right mouse click to bring up the context menu and select 'Delete object' or use Main Menu -> Edit -> Delete to delete the selected PNM.

To create a relation between the two objects, move the mouse over one object and click on the arrow icon which is on the top left corner of the object. Next move the mouse over the second object (this will show a dotted line rooted on the first object) and click inside the second object. A 'Choose Dependency' dialog will appear and ask you to choose a dependency type to describe the relationship.

Choose one of the dependencies and click the OK button.

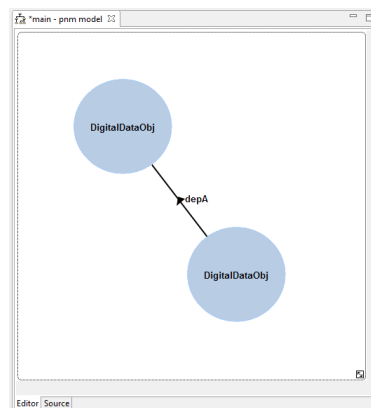


Figure 15 Two PNM objects with a 'depA' relationship.

Removing relations is similar to removing PNM objects described above; you can use the delete menu option from the Main Menu -> Edit or the context menu by right mouse click on the relation object.

If there are unsaved changes in the object, a * is displayed next to the editor name on the dialog box title. You can save changes by pressing CTRL + S hot key, or by clicking the Save button in the toolbar (📁), or using main menu -> File -> Save menu item.

4.7 Editing PNM objects

Click on the bottom PNM object in the canvas. It should be marked as selected with a dotted border around it; and the PNM Properties Editor (Figure 16) on the right side of the application should be filled with information about the selected object.

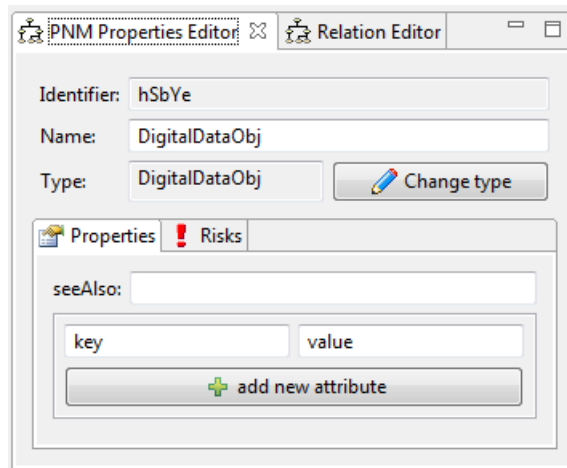


Figure 16 PNM properties editor.

You can use the PNM Properties Editor to edit PNM data. Try changing the name of the object to 'Source'; the name used in the canvas should be updated automatically.

Clicking on the 'Change type' button will pop up the type chooser dialogue with all DigitalDataObj subtypes specified in the schema file.

There are two tabs in this Properties Editor:

- Properties - shows all properties which are specified for this object. The default Schema specifies one property 'seeAlso'. You can add new property as a simple key and value pair and clicking on 'add new attribute' button at the bottom of the dialogue. It is permissible to delete user-defined properties, so that there is a 'delete' button to the right of every property which is not pre-defined in the default schema.
- Risks - shows the list of all risk objects associated with this PNM object (see Section 4.11).

4.8 Editing relations

Click on the arrow between two PNM objects in the canvas. Note that the Relation Editor (Figure 17) tab is now displayed in the Properties Editor instead of the PNM Properties Editor tab.

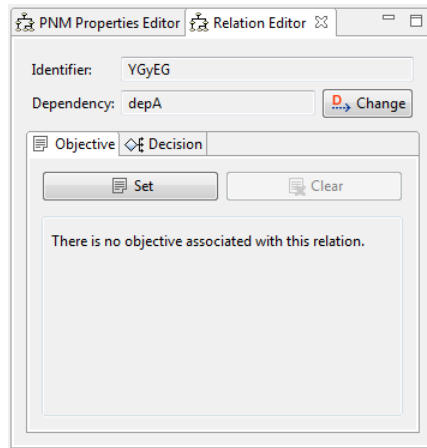


Figure 17 Relation editor.

You can change previously-set dependency of the relation by clicking 'Change' button near dependency name.

The Relation Editor also has two tabs:

- Objective - shows information about the set of objectives associated with this relation,
- Decision - shows information about the set of decisions associated with this relation.

These two properties are described in Sections 4.9 and 4.10 below.

4.9 PNM Objectives

Objectives are stored in multiple containers called Objective Sets and are created using similar methods as described for creating a new project (see Section 4.4).

Figure 18 shows the Objective Set editor:

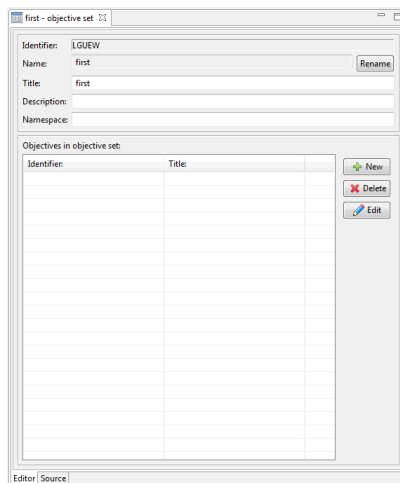


Figure 18 Objective set editor.

Default attributes defined by the PNM schema are shown at the top part of the editor. The default attributes include name, title, description and namespace. The specified namespace will be used for every objective in this set.

The lower part of the editor shows a list of all objectives within the set. The three buttons on the right are used to manage objectives:

- New - creates a new objective in this set and opens the Objective Editor,
- Delete - deletes the selected objectives,
- Edit - opens the Objective Editor for the selected objective.

New objectives can be created using similar approaches for creating new PNM object by using the menu option in the main menu (File -> New -> Objective) or the right click context menu.

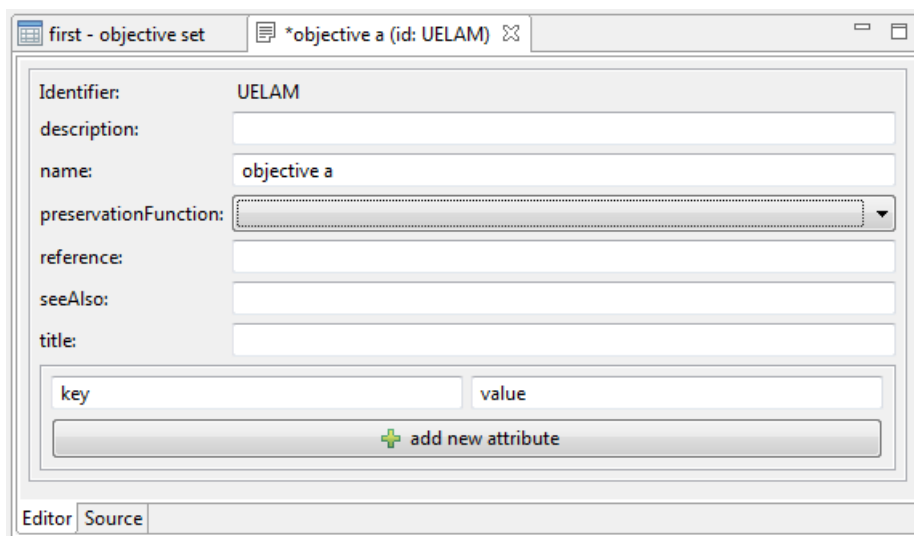


Figure 19 Objective editor.

Similar to the Objective Set Editor (see Section 4.9), attributes defined in the schema is displayed towards the top part of the menu and users can input arbitrary information as key-value pairs using the lower part of the dialogue. Objectives are associated with the relation between two PNM objects. You can set objectives for a relation by going to the 'Objective' tab and click the 'Set' button (see Figure 20). This brings up the Choose Objective Set dialogue (Figure 21).

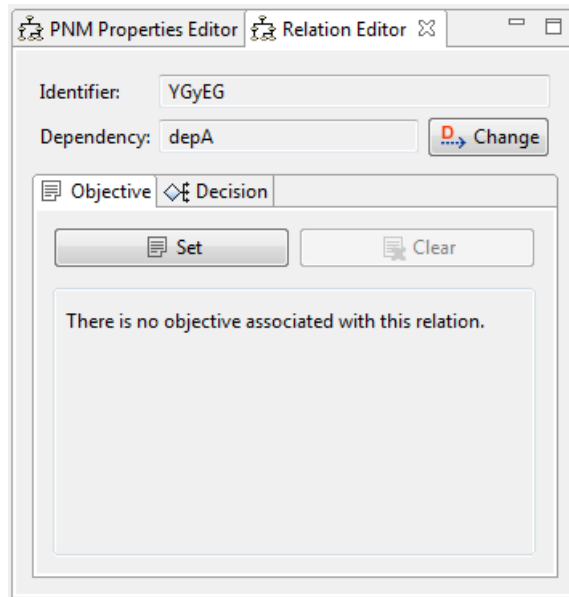


Figure 20 Working with objective sets via the relation editor.

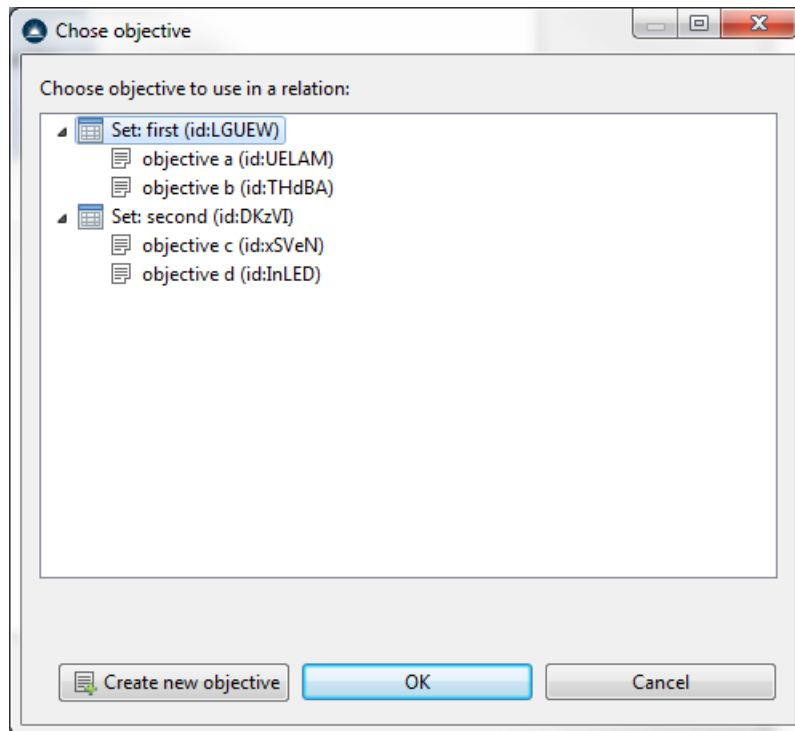


Figure 21 Choose objective set dialogue.

The 'Choose objective' dialog lists all objectives grouped in sets. Choose one of them and click the OK button. You can create new objectives using the 'Create new objective' button.


The tab in the Relation Editor (Figure 17) should now be filled with the chosen objective data. The editor provides three operations for working with objectives:

- Set - sets a different objective to use in this relation,

- Clear - removes the objective from this relation (it does not remove the objective from the Objective Set),
- Edit objective - opens the Objective Editor.

The same options are available in the:

- Main Menu in Edit menu (Set Objective, Delete Objective, Edit Objective, creating new objective is included in Set Objective operation),
- Context menu; by right clicking on the relation,

- Toolbar 

4.10 Decisions

Decisions are defined in a similar manner to objectives (see last Section). The main difference is that decisions do not have multiple decision sets; there is only one set of decisions for the entire preservation project. The Create New Decision wizard will only ask for the project in which the decision should be created.

New decisions are created using similar approaches to creating a new decision (see Section 4.9). Double click on the 'Decisions' node in project explorer to display the list of decision objects associated with the project (Figure 22):

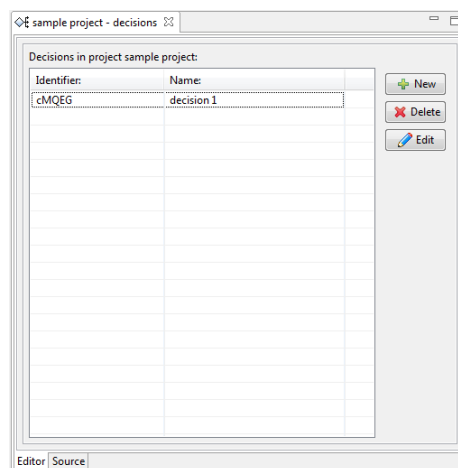


Figure 22 Decision object/s associated with the project.

To add a new decision, clicks on the 'New' button, a new unnamed decision will be added to the list. Double click on it or use the 'Edit' button to open the Decision Editor to edit its attributes:

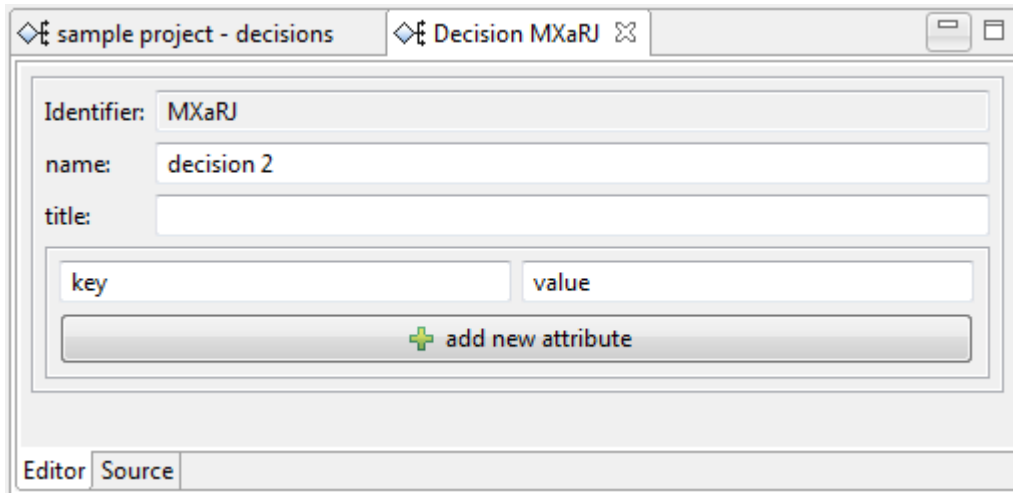



Figure 23 Decision editor.

The Decision Editor (Figure 23) is similar to the Objective editor; the default attributes are predefined by the schema. Users can input custom attributes using the key value input fields and add them using the 'add new attribute' button.

A decision is associated with a PNM relation object. To add a decision to a relation, open the 'main' PNM model editor, click on a relation to display the relation editor and choose the 'Decision' tab.

As with objective (see previous section), you can set a decision for a relation using one of the following options:

- Clicking on the 'Set' button in the tab and choose a decision from the pop up dialog. You can create a new decision by pressing 'create new objective' in the dialog;
- Using the Main Menu -> Edit -> Set decision menu item and choose an existing decision or create a new one;
- Using the right click context menu option on the relation;
- Clicking on the toolbar icon: 

Afterwards, the 'Decision' tab should be filled with the decision information: its name and list of attributes. The tab provides two buttons, which offers similar operations as in the Objective tab:

- Clear – removes the decision from the relation (it does not remove decision from the project),
- Edit decision - opens the editor for the decision.

All set, clear, edit options are also available in the toolbar, main menu and right click context menu.

4.11 Risks

Risks are grouped in one set for the entire project. Risk Set Editor and Risk Editor look and behave in the same way as for decisions.

A risk object is associated with a PNM object. You can manage risks for a PNM object by choosing the Risks tab on an PNM object editor (Figure 24).

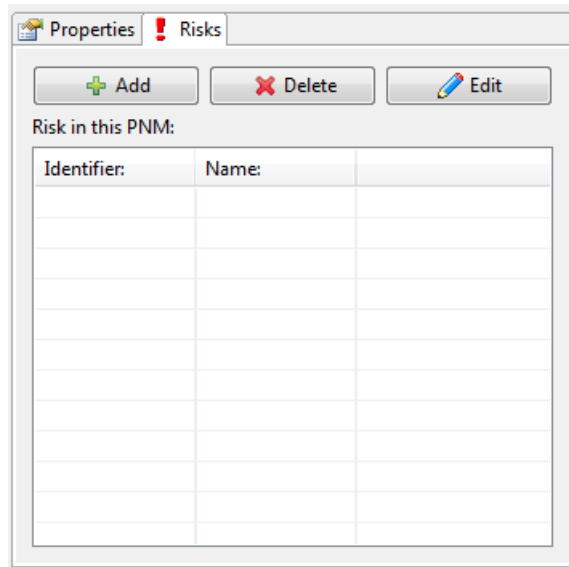



Figure 24 Risks associated with a PNM object.

The editor shows a list of all risks set for this PNM object. You can add new risks in several ways:

- Clicking on the 'Add' button and choosing a risk from the list shown in the popup dialog. There is a 'Create new risk' button which allows you to create a new risk and automatically add it to the selected PNM object;
- Clicking on the Main Menu -> Edit -> Add risk and selecting a risk. There is also an option to create new risk and automatically add it to the PNM object;
- Using the right click context menu after selecting a PNM object in the canvas;
- Clicking on the toolbar icon .

Afterwards, the 'Risks' tab should be refreshed and filled with all selected risks. There are two more buttons in the tab:

- Delete - deletes selected risks from the PNM object only un-maps the association between the Risk and PNM objects (it does not delete the actual risk objects from the project),
- Edit - opens editors for every selected risk.

All add, delete, edit options are also available in toolbar, main menu and right mouse click context menu.

4.12 Source and exporting model

You can view the source of each editor by selecting source tab on the bottom of the editor.

It is possible to export a PNM model and all associated information to an external file. If there are any objectives, risks or decisions used in a model they will be in the exported file. To perform an export use Main Menu -> File -> Export and select the PNM Model to export.

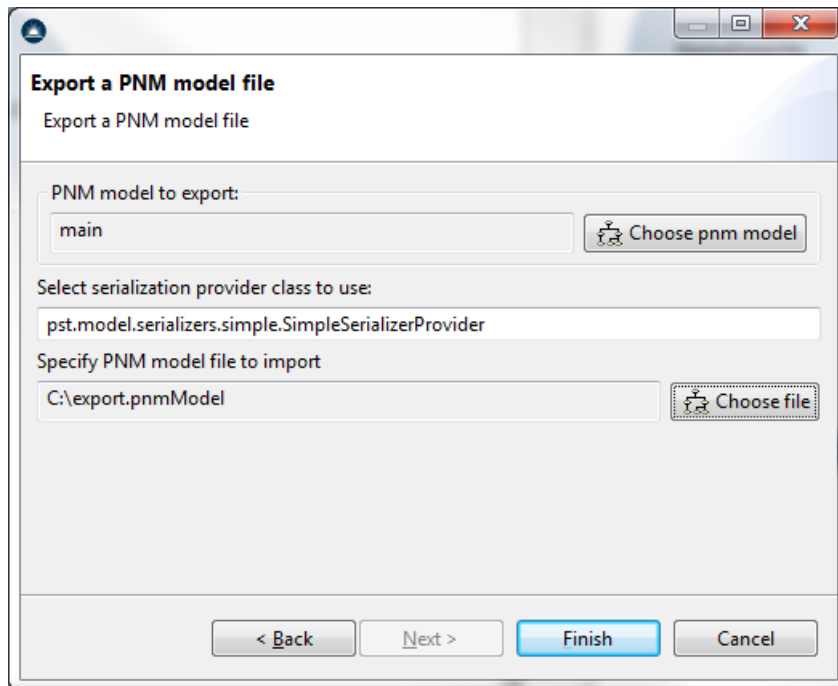




Figure 25 Export a PNM model.

Click on the 'Choose pnm model' button to select the PNM model that you want to export and use 'Choose file' to specify the location of the destination file (see Figure 25). You may change the serializer provider for handling the export instead of using the default one, which is the one chosen on creating the project. Press the 'Finish' button to perform the export.

4.13 Miscellaneous PNM Model editor features

The PNM Model editor enables users to save a snapshot of the model as an image file. To perform this use Main Menu -> File -> Save as image option or user toolbar icon . Three image formats are supported:

- jpg
- png
- bmp

PNM Models may grow very large and difficult to edit. The zoom feature can be used to focus on part of a large model. You can change the zoom resolution by using Main Menu -> View -> Zoom options, right mouse click context menu on the blank space in the editor or use the toolbar icons .

You can also maximize the editor view by pressing CTRL + M; restoring the zoom level is performed with the same key action.

4.14 Transformation objects

Transformation is a special preservation strategy that describes how a particular object type A could be preserved by transforming it to a different object type B; for example, if A becomes obsolete, one option would be to convert all A-objects into B-objects, and then follow the preservation strategy for B.

In the PST there is no special treatment of Transformation objects, except for the calculation of the cost of a Solution (see Section 4.15.1). Transformation objects are modelled using specific object properties in the PNM schema (which is chosen by the user on creation of each Preservation Project). The PNM schema supplied with the PST distribution contains these properties:

- Transformation object: a yes/no flag used to indicate whether the object is a Transformation object;
- Transformation cost: a numerical value for the (one-off) cost of performing the transformation;
- Cost: a numerical value for the (ongoing) preservation cost of this object;
- Cost model ref: this is intended to be a reference to a document defining the cost model (this is not used by the PST, but is intended as information for model builders/users)

Figure 26 shows an example of a PNM containing a Transformation object. This PNM describes two alternative preservation strategies for MST: one is to preserve the MST manual and directory structure; the other is to transform MST to AMES. The “AMES Converter” object captures the Transformation step (its “transformation object” property is set to “yes”) and assigns a transformation cost.

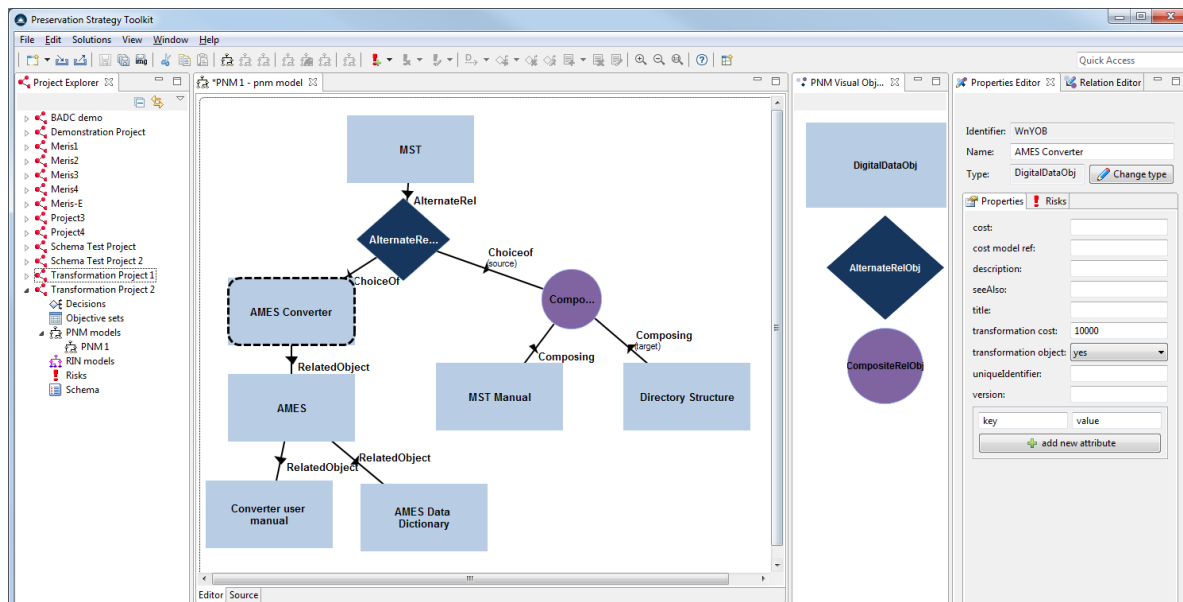


Figure 26 Example of a Transformation object

4.15 Working with Solutions

A PNM Model may contain Alternative nodes (see Figure 27); the interpretation is that the parent or source PNM object may be preserved by preserving one or more of the children (and any network on which they depend). The Solutions function allows users to “mark” parts of a model as Included, Excluded or Undecided. Once created, Solutions can be saved, and reloaded later.

Suppose we have the following PNM:

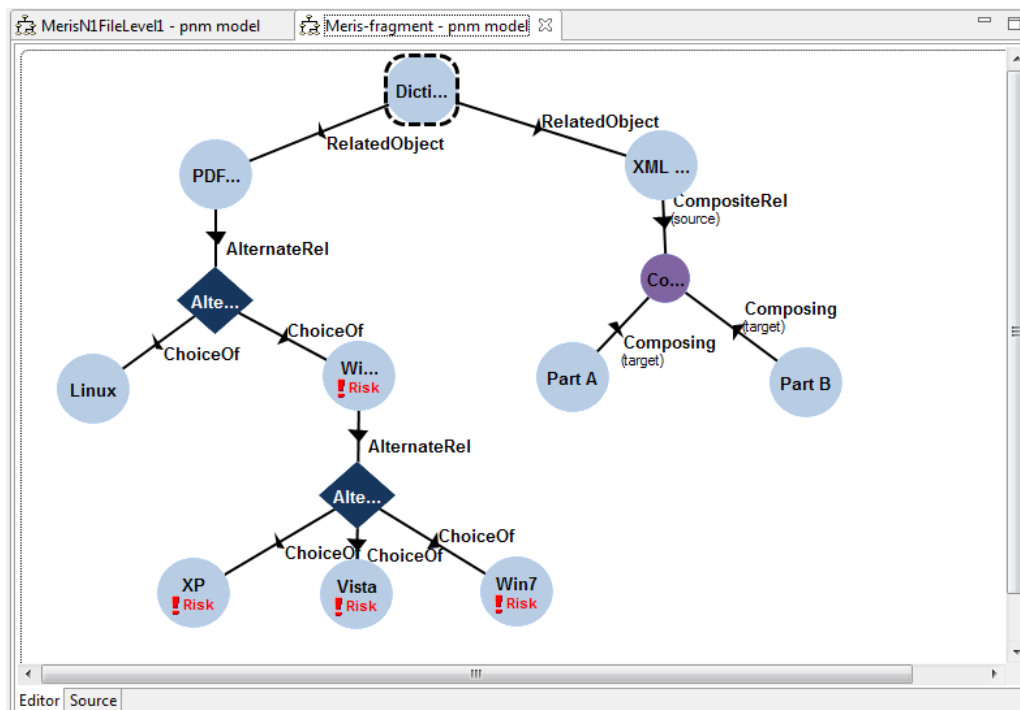



Figure 27 Example PNM with alternative preservation solutions.

Normally, we start by Including the topmost or primary object in the PNM. To do this, we select the object, and then choose “Include subtree in current Solution...” option from the main Solutions menu, the right-click context menu or the  toolbar icon.

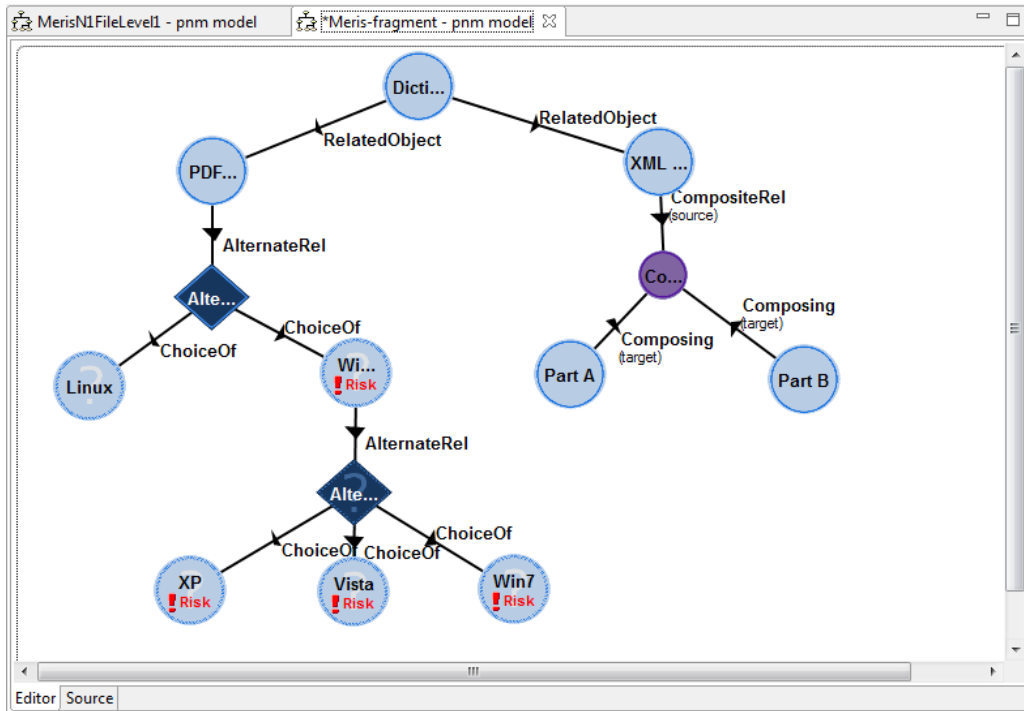


Figure 28 An incomplete PNM solution showing undecided nodes.

PST marks as Included (by drawing a ring around the edge) all objects on which the chosen object *definitely* depends. However, where there is an Alternative node, all objects under it are now marked as Undecided (with a '?' in the background, see Figure 28). We could save the Solution like this, or we could decide to include one of the alternatives. To do this, select the top object (e.g. Windows) under the Alternative node, and choose “Include subtree in current Solution...” as before.

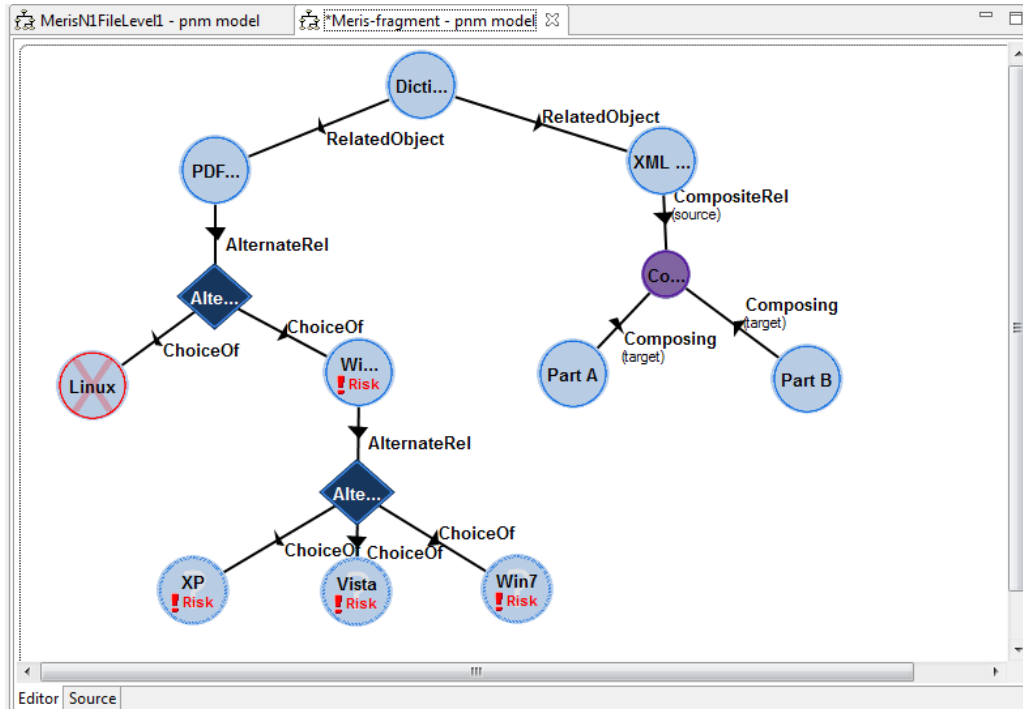


Figure 29 PNM solution showing the user chosen nodes.

Again, all objects on which this second object definitely depends are marked as Included (see Figure 29). Note that the other alternative is now marked as Excluded (with a red cross). The PST does this because, as we have chosen one alternative, the others are not mandatory. We can still decide to include some or all of the alternatives as well: simply select the topmost object and ask to include it.

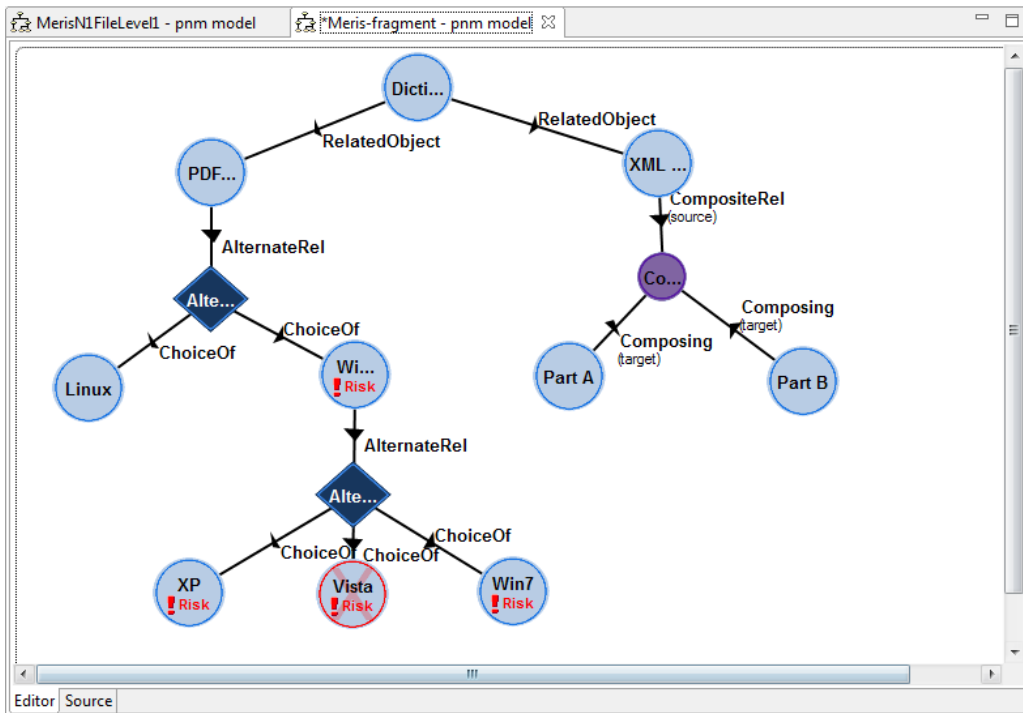



Figure 30 PNM solution showing more than one alternative nodes are chosen.

Here, we have asked to include Linux, and refined the alternatives under Windows to include XP and Win7, but not Vista (see Figure 30).

We can also explicitly exclude part of the PNM model from our solution by selecting its topmost object and choosing the “exclude subtree from current Solution...” option from the main Solutions menu, the right-click context menu or the  toolbar icon.

4.15.1 Inspecting the Cost of a Solution

The PST currently provides a basic cost mechanism that aggregates the total cost value in the chosen solution path. This assumes that each PNM object has a numerical ‘cost’ property, and that Transformation objects have a numerical “transformation cost” property. The PNM schema supplied with the PST distribution defines these. However, if “cost” or “transformation cost” are missing from an object, the calculation ignores them (so the cost is effectively zero). Transformation costs are considered separately, and can be included or excluded from the calculation.

The ‘Show cost of current Solution’ action on the toolbar Solutions menu asks whether to include or exclude any transformation costs; then it displays a dialogue showing the total cost of (all the nodes in) the current Solution.

Figure 31 shows a PNM with a Solution that includes a Transformation object. When “Show cost of current Solution” is chosen, a dialog asks whether the Transformation costs should be included (Figure 32); choosing Yes produces the result in Figure 33.

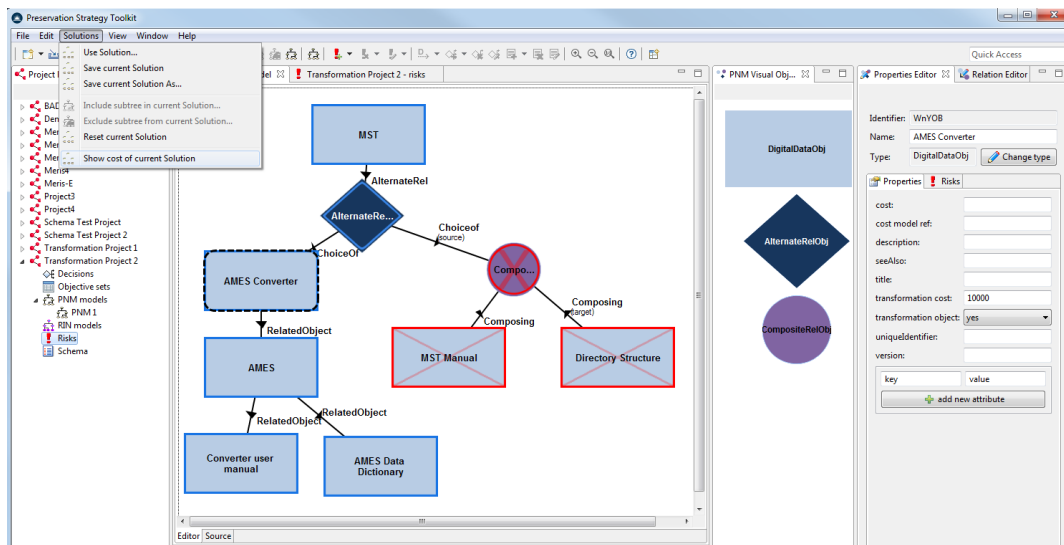


Figure 31 About to show cost of current solution

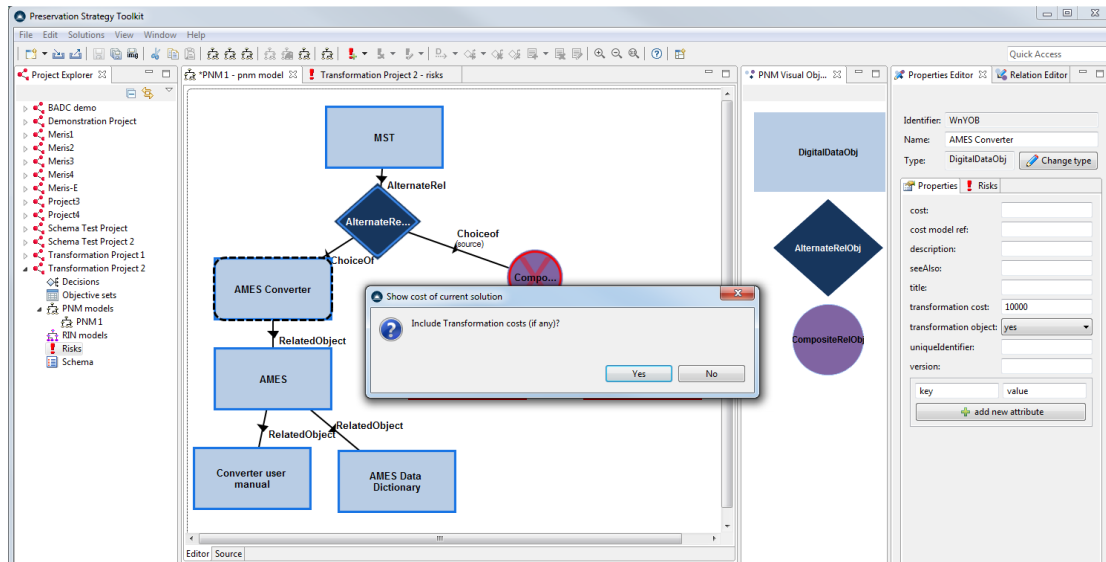


Figure 32 Include or exclude Transformation costs?

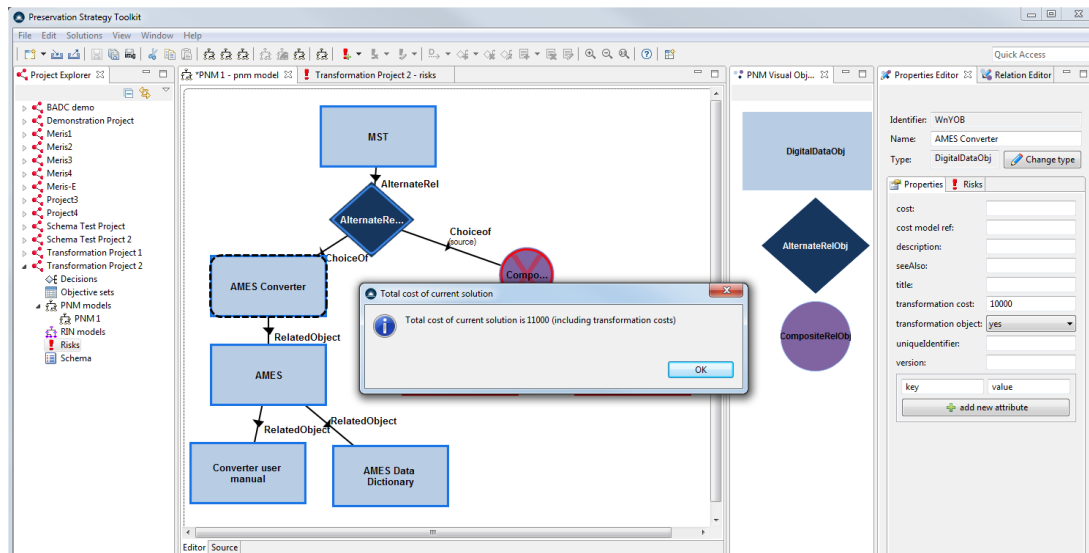


Figure 33 Cost (including transformations)

To compare the costs of two Solutions, you could construct the first Solution display and make a note of its total cost; then change it (or clear the whole Solution and create a new one), and display its cost.

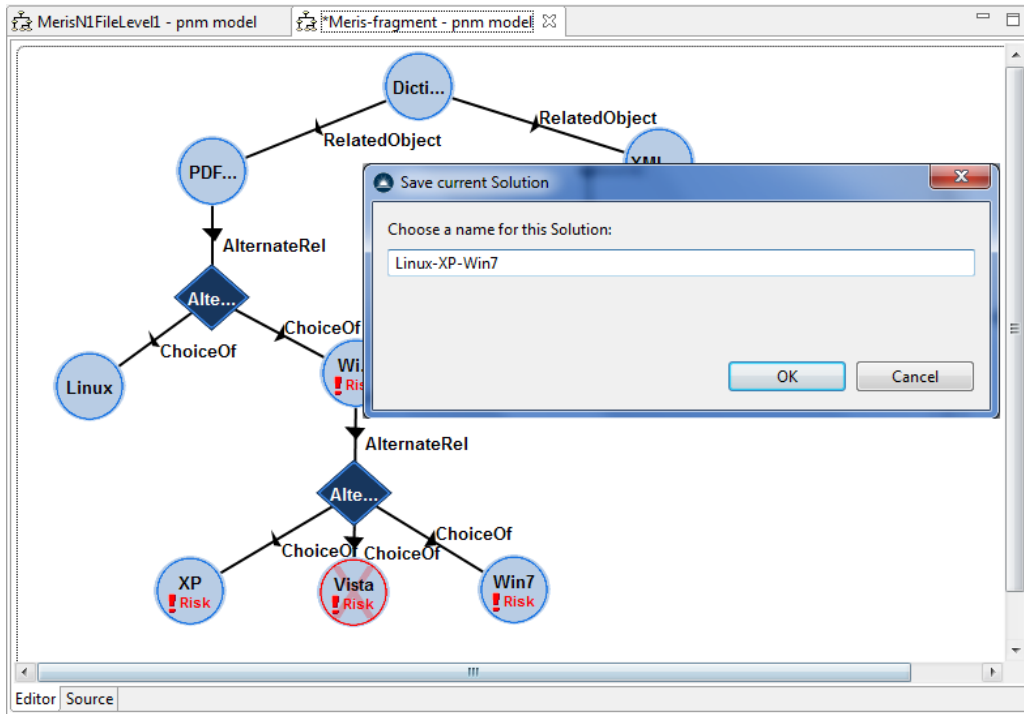


Figure 34 Saving a solution.

To illustrate, we save the current solution as Linux-XP-Win7 (Figure 34); then show its cost (Figure 35):

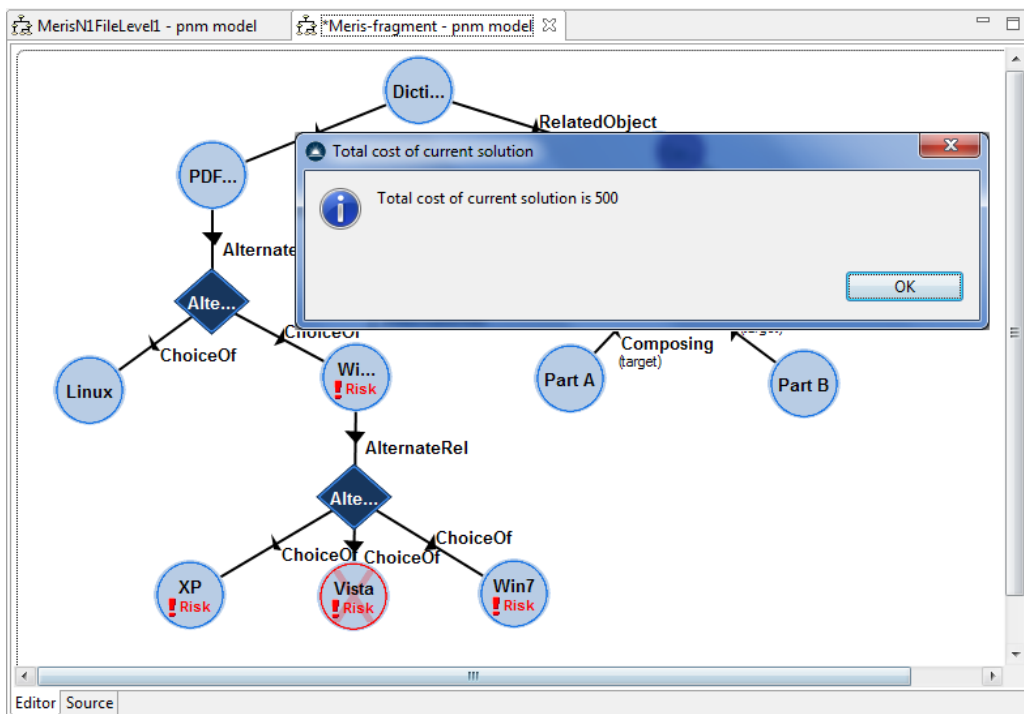


Figure 35 Displaying the cost of a chosen solution.

Next, we add Vista to the current solution, and calculate the cost again (Figure 36):

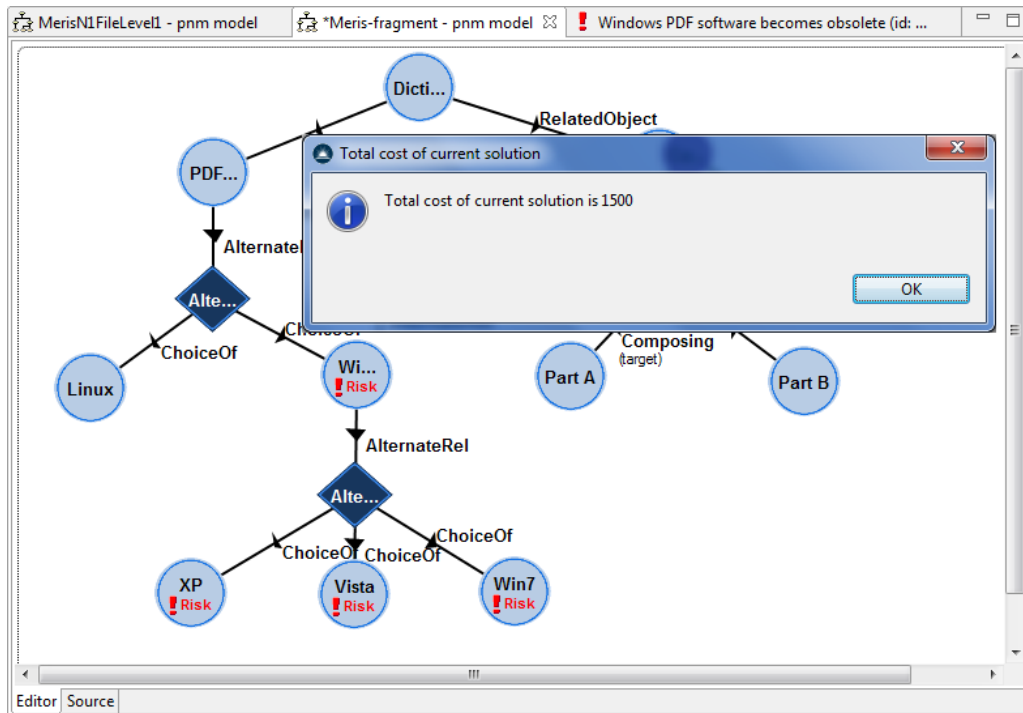


Figure 36 Displaying cost for the solution with Vista included.

If we decide this solution costs too much, we can discard this and reload the Vista-free solution that we saved earlier.

5 Using the SCIDIP-ES ReplInfo Toolkit

5.1 Introduction

As described in Section 2, the process for designing PNM and ReplInfo Network (RIN) graphs are similar. Rather than developing a separate ReplInfo Toolkit from the ground up, we have therefore decided to leverage the PST code base by providing a dedicated Eclipse perspective for working with ReplInfo objects. This approach enables us to use the same core graphical and properties editors for PNMs and RINs, thereby lowering the development efforts and learning curve for users.

A key objective of the RIT is to facilitate collaborative preservation activities by enabling data archives to share ReplInfo objects stored in one or more ReplInfo Registries. Users can use RIT to create new ReplInfo objects, retrieve, manage and store them from/in Registries (see the following sub-sections). For a discussion of the concepts of ReplInfo objects and their application within SCIDIP-ES, please see Sections 2 and 3 of the D21.3 Master Document.

As RIT and PST shares the core code base, the treatment of RINs is very similar to PNMs. Figure 37 shows the RIT perspective which has a similar look and feel to the PNM one (Figure 5). RIN models are grouped under the 'RIN models' node in the Project Explorer as PNM models are stored under the 'PNM models' node, and the process of creating new RINs are similar to creating PNMs (see Section 4.6).

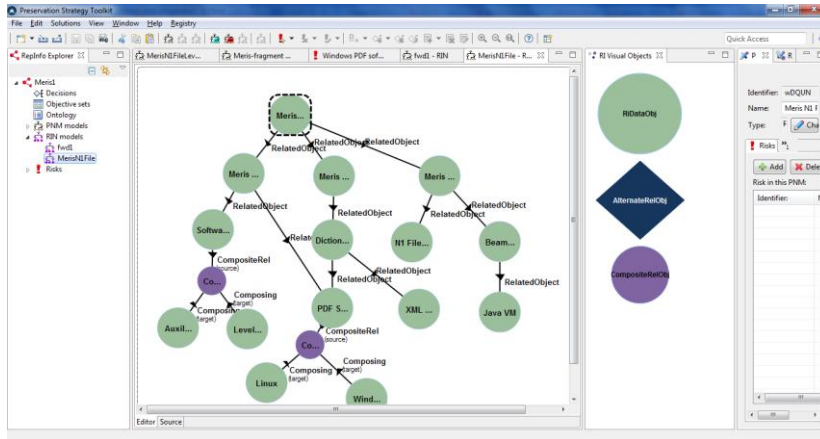


Figure 37 The RIT perspective.

The key difference between PNM and RIN models is that they are based on different information models (though some aspects are the same; for example, both contain Composite and Alternate Relation objects). Users need to provide this schema on application start up as described in Section 4.4 above. An example of the RIT schema file is provided in the schemas folder of the distribution zip and appended in Annex E.

In a RIN, the basic building blocks are ReplInfo Data Objects (RiDataObj or RIDO's). A RIDO can contain details of a concrete ReplInfo (RI) object and/or information about the graph structure, or ReplInfo Label, that describes the ReplInfo object in terms of other ReplInfo objects. See Section 3 of D21.3 Master Document for a discussion of ReplInfo and ReplInfo Label. (The RIT uses the term “Graph Label” as a synonym for “ReplInfo Label”.)

The details of the RI and information about the graph label are held in the RIDO's properties fields; these are similar to PNM object properties, and most can be seen (and some edited) in the Properties Editor panel in the RIT.

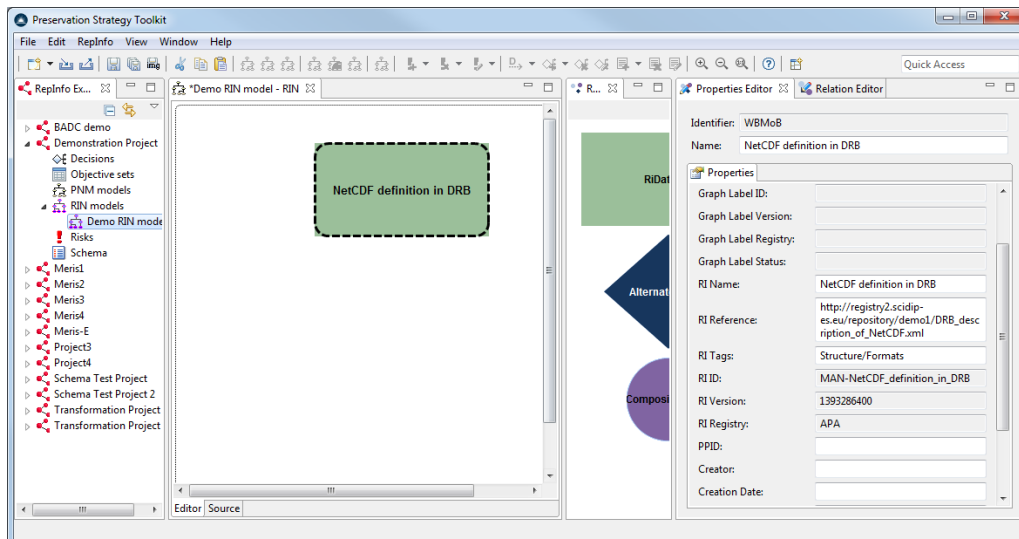


Figure 38 A simple RIDO.

Figure 38 shows a RIDO that contains details of an RI object. The Properties panel shows several properties whose name begins with "RI", including the RI Name, RI Reference (a locator for the object), and RI Tags (a list of OAIS Types to which the RI is applicable, e.g. Semantic/Document). The RI ID, RI Version and RI Registry properties are all set when the RIDO (or more precisely, its RI) has been loaded from, or has been uploaded to, a Registry; these values cannot be (directly) modified by the user. (We will cover loading from and uploading to a Registry later.)

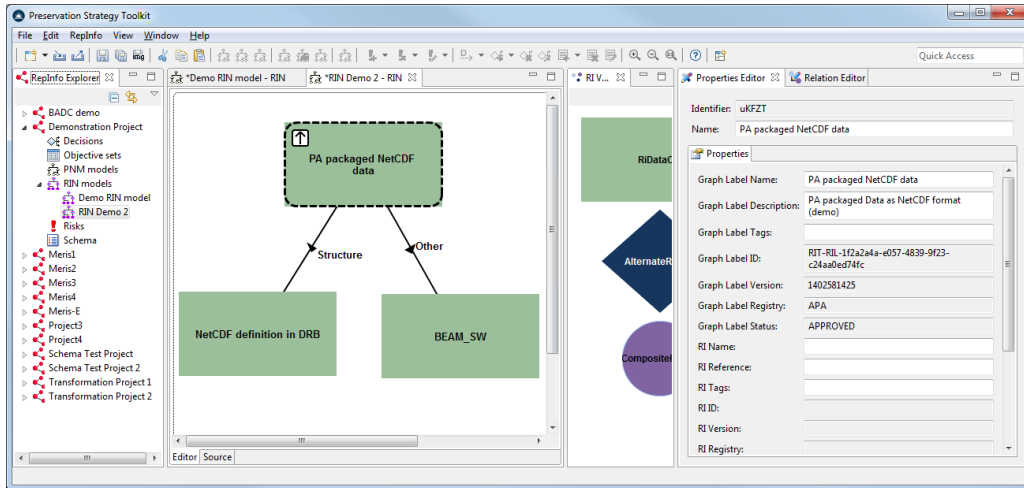


Figure 39 A small RIN.

Figure 39 shows a small RIN that consists of a single RepInfo Label, or Graph Label. The topmost RIDO contains details of the Graph Label: the Properties panel shows content in fields beginning with "Graph Label". Graph Label Name and Graph Label Description are the (display) name and description of the Graph Label; Graph Label Tags is a list of OAIS Types for this Graph Label. The remaining fields (Graph Label ID, Version, Registry and Status) are all set and maintained by Registry load/upload operations (similarly to RI ID etc.)

The top RIDO has no RI details. In general, a RIDO can contain both.

The structure of the Graph Label is shown in the rest of the RIN: here there are two children (with arcs labelled Structure and Semantic), each of which is another RIDO.

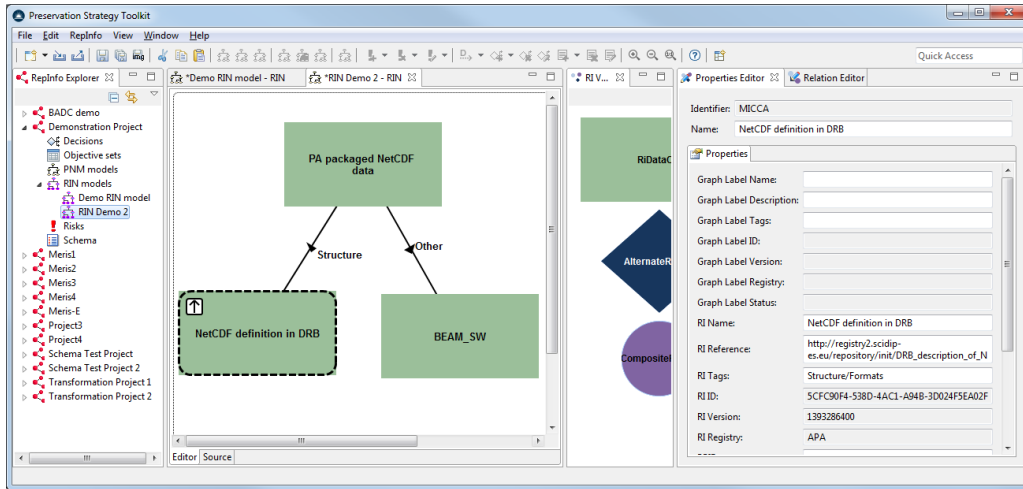


Figure 40 Simple RIN with child selected.

Clicking on one of the child RIDOS (Figure 40) shows its properties. In this case, it contains RI details but no Graph Label details. (In fact, there is a Graph Label associated with the RI, but this is not visible; we will return to this later.)

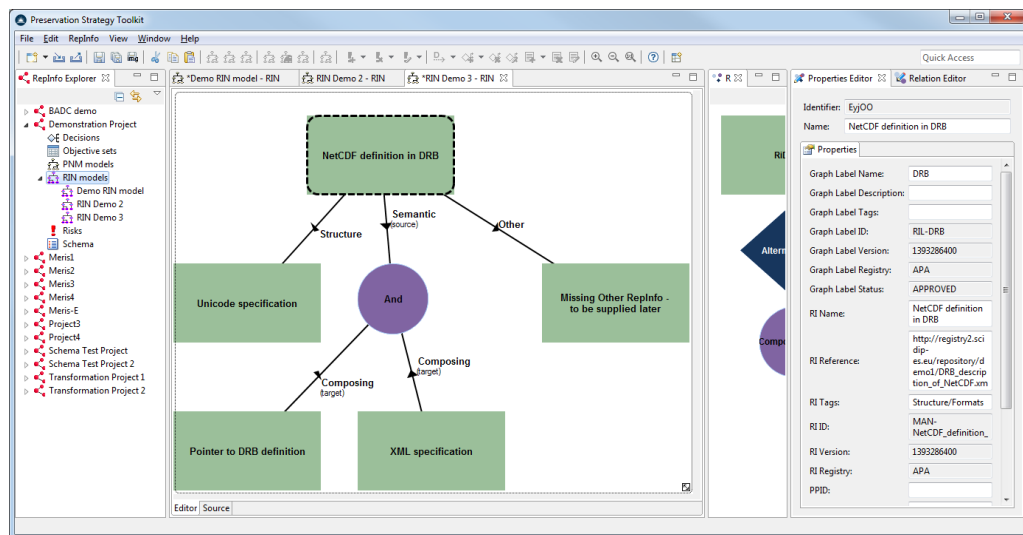


Figure 41 Graph Label with more complex structure

Figure 41 shows another RIN; again, this contains a single Graph Label, but one with a more complex structure. There are three relations from the top RIDO (there can be three at most), but the middle (Semantic) "child" is a more complex tree comprised of an And node and two RIDO 'leaves'. For a valid Graph Label, each And (or Or) node must have at least two subtrees, and each leaf must be a RIDO. (The leaf must also contain valid RI details, which we will cover later.)

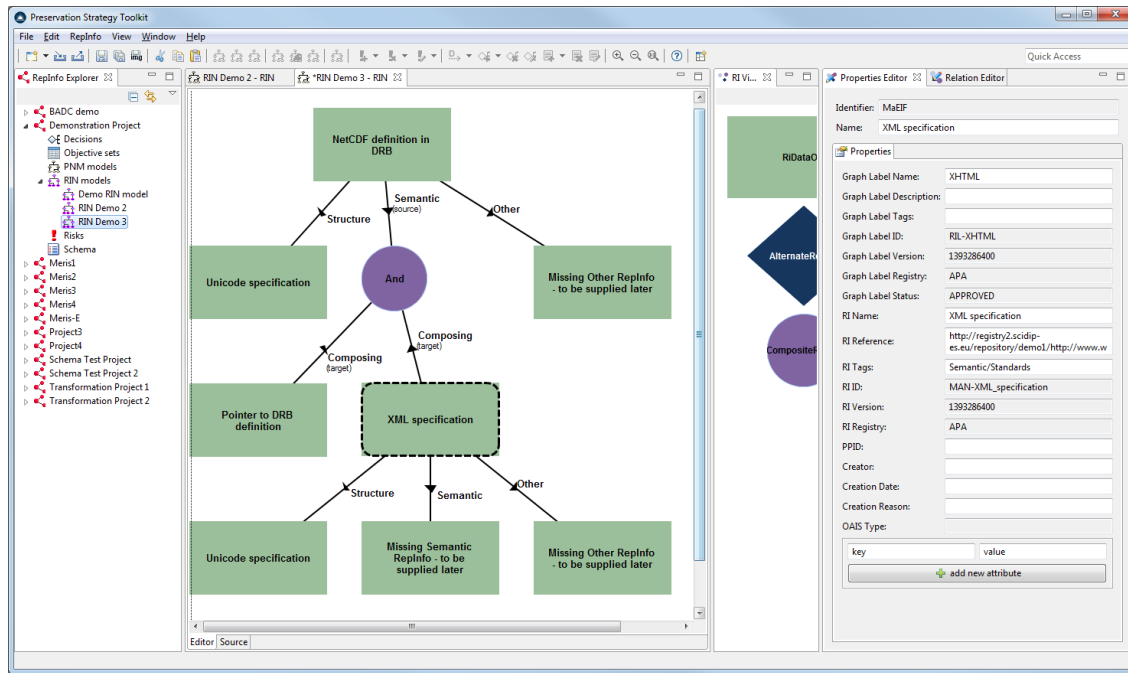


Figure 42 Complex RIN with multiple Graph Labels

Figure 42 shows a more complex RepInfo Network (RIN), containing multiple Graphs. The RIDO “XML Specification” has been selected, and the Properties Panel shows that it contains both RI and Graph Label details. It is one of the leaves of the topmost RIDO “NetCDF definition in DRB”, but the Graph for its RI has been added to the RIN. There are several ways to do this; we will introduce them later.

As in PST, the user provided a schema to specify the default properties or attributes being used in the RIDO properties editors. The example RIN schema is tightly coupled to the SCIDIP-ES information model and the SCIDIP-ES Common Framework library which abstracts the interactions between RIT and SCIDIP-ES Registries. If users wish to provide their own custom schema, they may also need to implement their own Serializers (see Section 2.2) to support the custom behaviour encapsulated in their schema.

It should be noted that similar to PNM data objects (see Section 4.7), users can add their own custom properties as key-value pairs to individual RIDOS. These will be saved in the RIT’s local copy of the RIN but are **not** preserved when RIDOS are created or updated in a SCIDIP-ES Registry as the latter adheres strictly to the information model defined in the RIN schema.

RIN graphs can be constructed in the same manner illustrated in the previous Section for creating PNM graphs (though there are additional validation criteria that must be met before a RIN can be uploaded to a Registry; these will be covered later). In the remainder of this section, we focus on describing RIT-specific functionalities within the context of the information model defined by the example RIN schema. For example, it is possible to use RIT to search RepInfo Registries (Section 5.3), to extract and display fragments of RINs (Section 5.4); to modify them (Section 5.6); to create new RepInfo objects (Section 5.5); and to upload modifications back to the Registry (Section 5.10) for sharing with other users.

5.2 Creating a new RIN model

To create a new RIN model, choose File > New > RIN model (Figure 43). As with new PNM models, you may have to select a Preservation Project in which it will be created (see Section 4.6).

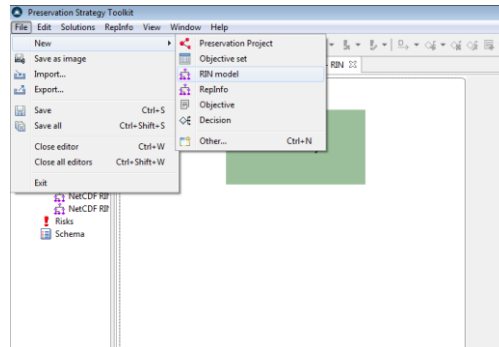


Figure 43 Create a new RIN model.

5.3 Searching for ReplInfo objects and Graph Labels

The RIT can be used to search SCIDIP-ES ReplInfo Registries for ReplInfo (RI) objects and Graph Labels (ReplInfoLabels or RILs, the building blocks of RINs), via the SCIDIP-ES Common Framework. First, you should open an existing RIN model, or create a new one (see previous Section). This launches the RIT perspective in which extra, RIT-specific, functions become available.

Start by dragging a new RiDataObj (or RIDO) onto the canvas, and selecting it. Search will then use this RIDO as its starting point. If no RIDO is selected, Search will add a new one to the canvas, at the top left hand corner of the canvas.

A search can be started in two ways:

- By selecting the ReplInfo main menu item and choose Search Registry for Graph Labels/ReplInfo;
- By using the right click context menu; select a RIDO, right-clicking on it and choose Search Registry for Graph Labels/ReplInfo from the pop-up context menu.

This brings up the Search dialog:

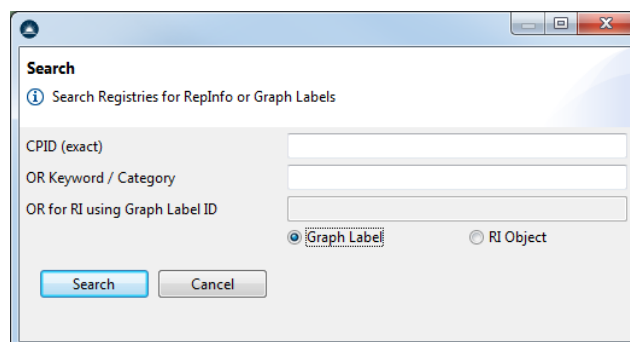


Figure 44 Search ReplInfo Registry dialogue.

The CPID field can be used if you know the exact CPID of the target RI or Graph Label (for example, it could be copied from the RI or Graph Label ID fields of another RIDO). The Keyword/Category field is

more general-purpose: the value entered here will be used as a search term by the target Registry (or Registries). Graph Labels are searched by keyword; RI objects are searched by the Category (as a substring) published by the Registries. See Section 3 of the D21.3 Master Document for a description of Category. As an example, searching for RI Objects with “software” will find all RI objects that have a category that contains the word “software”. The “RI using Graph Label ID” field can be used to search for RI that uses a particular Graph Label (see later). The inputs are mutually exclusive (you can search by ID *or* keyword/category, but not both); entering a value in one input disables the others. (To re-enable them, clear the active input.)

The “RI using Graph Label ID” input only applies to RI, so it is disabled whenever the Graph Label radio button is selected. If you select the RI Object button and enter a value here, the Graph Label button is disabled.

Click Search; the results will be displayed as a list of CPIDs of RI objects or RILs, showing their names, IDs and source Registry.

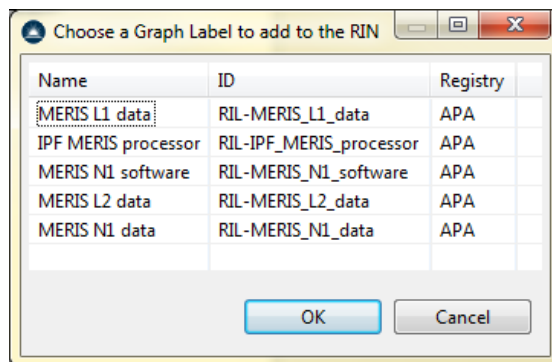


Figure 45 Registry Search result.

To select a particular result, click on its Name and click OK. If it is a RI object, the RIT will add its details to the selected RIDO (or a new RIDO, if none was selected). If the result is a Graph Label, the RIT will obtain its details from the Registry, populate the selected RIDO (or create a new RIDO) and update the RIN graph to include the structure described by the Graph Label.

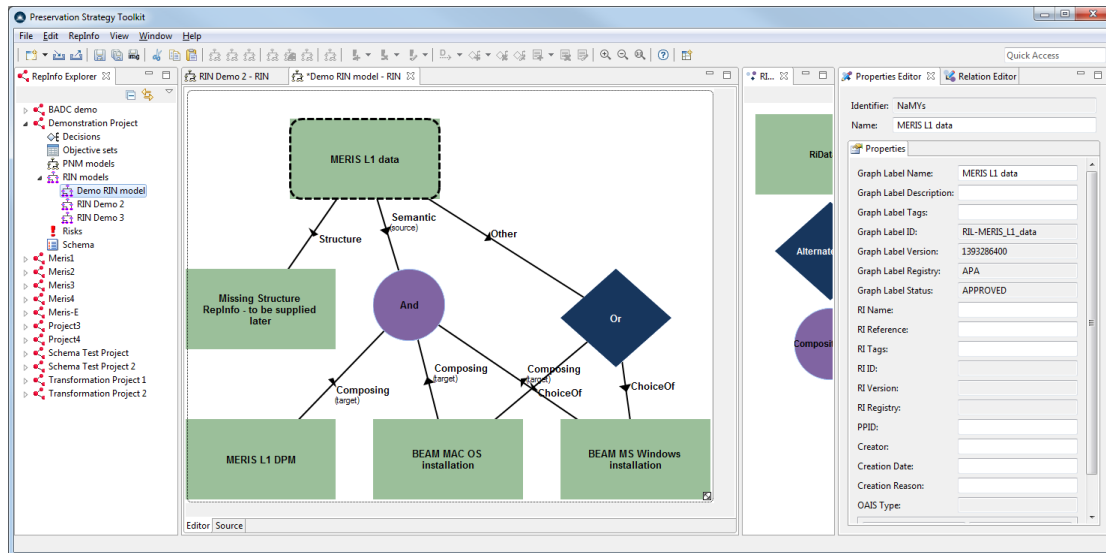


Figure 46 RIN representing the retrieved MERIS L1 data Graph Label.

If you create a new RIDO and search for a Graph Label, the RIDO will contain details of the Graph Label, but no RI details. However, for RIDOS like this you can use Search to look for RI that uses this Graph Label: selecting the RIDO and choosing Search shows:

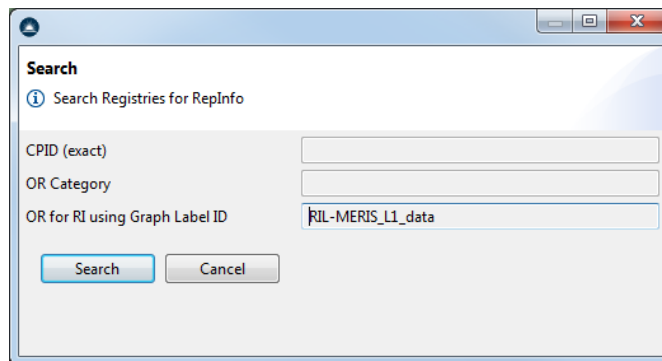


Figure 47 Search on a RIDO that has a Graph Label (but no RI)

The “RI using Graph Label ID” box has been pre-filled with the ID of the RIDO’s Graph Label. None of the fields are editable, so this is the only search that can be performed in this case. In other words, if a RIDO has Graph Label details but no RI details, you can only search for RI that uses that Graph Label. It is not possible to search for other RI, nor is it possible to search for another Graph Label.

If the Graph Label is “new” and has no ID (see later), it is not possible to search for RI objects that use it. You must create a new RI, or search for RI in a blank RIDO before creating the Graph Label. (See later.)

A Graph Label may be described as a fragment of a virtual RIN graph and typically contains one to three child objects representing the three OAIS Representation Information types (Semantic, Structure, Other). Each child object may be a simple leaf or a more complex And/Or tree. For complex Graph Labels, the resultant RIN may be cluttered and some of the RIDOS may overlap on the canvas, in which case users may need to manually reposition the objects to optimise the display.

5.4 Expanding a child of a Graph Label

If the child RIDO is a simple leaf, it is normally a placeholder for another Graph Label lower down the virtual RIN graph. When a Graph Label is extracted from the Registry, such a leaf RIDO contains only details of its RepInfo object: its name, a URL pointing to the actual Representation Information object and the ID pointing to the Graph Label used to help interpret it. If you want more information about the RI, you can “expand” the leaf and reveal more of the virtual RIN (see Section 2 of the D21.3 Master Document). The operation retrieves the Graph Label for the leaf from the Registry and expands the graph at that point.

To expand a leaf: click on the RIDO to select it; then right-click, and choose “Expand from Registry” from the pop-up context menu. The retrieved child-trees of the RIL should be appended to the RIDO.

Once a leaf has been expanded, it cannot be expanded again.

*There is no corresponding “collapse” operation. Note that though it is possible to delete the children of a RIDO, or the arrows to the children, this is interpreted as **editing** the RIDO’s current Graph Label, not as hiding it.*

The screenshot below shows the result of expanding the “GIS3D Model Example” child of the “GIS3D Model GVS” RIDO.

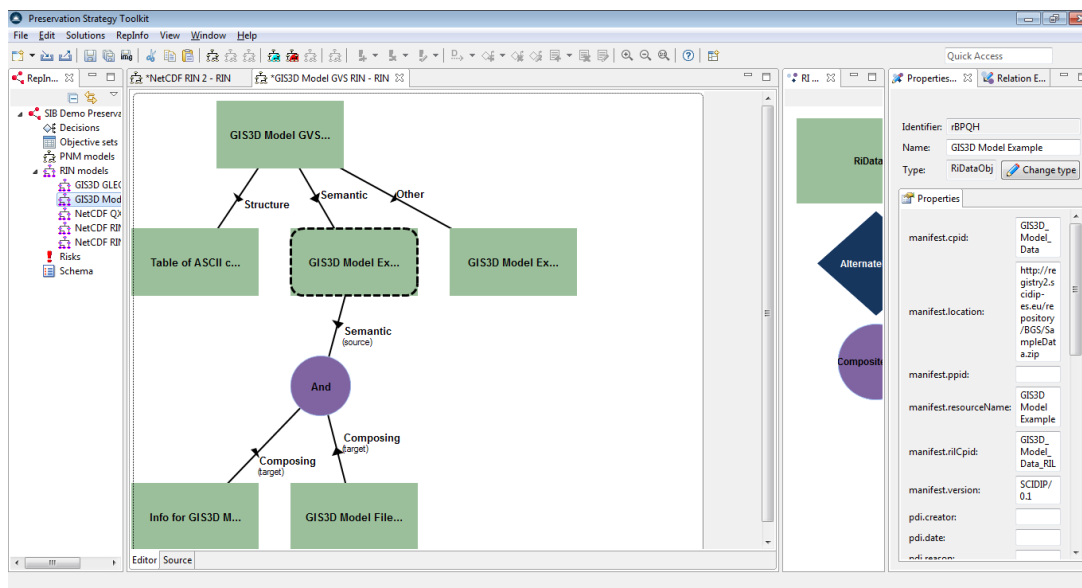


Figure 48 RIN after expanding the GIS3D Model Example leaf RIDO.

Thus it is possible to use the RIT to explore selectively the virtual RIN that contains a particular RepInfo object, by searching for its Label in the Registry and displaying it in the RIT, then expanding the leaves to display as much of the virtual RIN as desired.

5.5 Creating New ReplInfo

The RIT provides a set of dialogues that guide users through the process of creating a new Representation Information objects. Users can add and configure tools for creating particular categories of Representation Information or access external help resources via links embedded in the dialogues.

As with searching for RILs, the simplest way to create new RepInfo objects is first to drag a new RIDO onto a RIN model canvas, then click on it to select it. (If no RIDO is selected, then a new one will be added to the top-left of the canvas on completion). Next choose “Create new RepInfo...” from either the RepInfo main menu item, or from the right click context menu on the selected RIDO.

*It is possible to choose Create New RepInfo on a RIDO that already has RI details. In this case, you will be asked to confirm **replacement** of the RI with a new RI object. To **edit** the current RI instead, (cancel and) use the Properties panel.*

This launches the New Repinfo Wizard, a sequence of dialogues to guide you through the process of creating a new Representation Information object. The first dialog (Figure 49) asks you to select the category for tagging the new object.

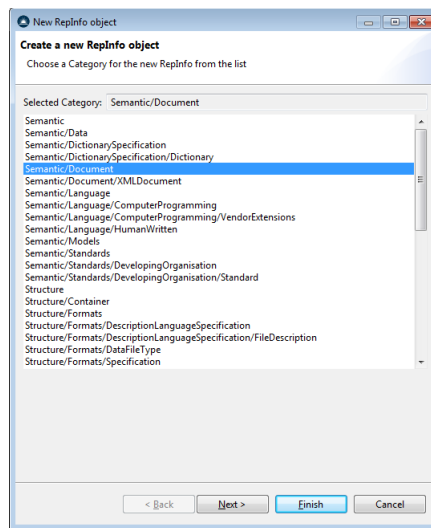


Figure 49 Specify a category for the new Representation Information object.

The next dialog presents a list of tools suitable for creating Representation Information of the chosen category (Figure 50).

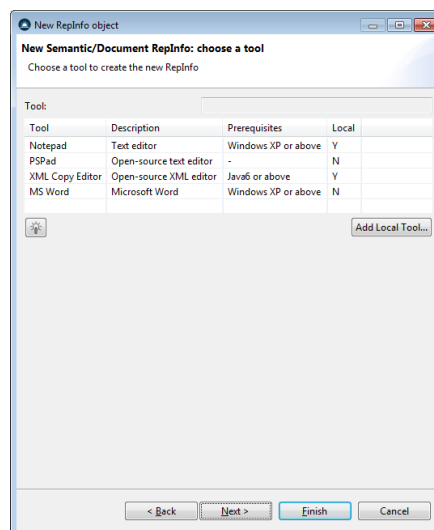


Figure 50 Dialogue for choosing or adding a tool for creating Representation Information.

Each tool has a name, a description, a note about any prerequisites, and an indication of whether the tool is locally installed. If the tool is locally installed, you can use it to create RepInfo immediately, and if chosen, the tool will be launched when user select the ‘Finish’ button. If the tool is not installed, select it and click Next to open a browser and navigate to the external resource which gives more information on the tool, how to use and install it. APA¹² currently provides the external help resources. If you would like more general information on your chosen category of RepInfo, clicking the light bulb icon will open a browser on an appropriate help page provided by the APA.

The “Add Local Tool...” button can be used if you would like to use a locally installed tool not listed in the dialogue.

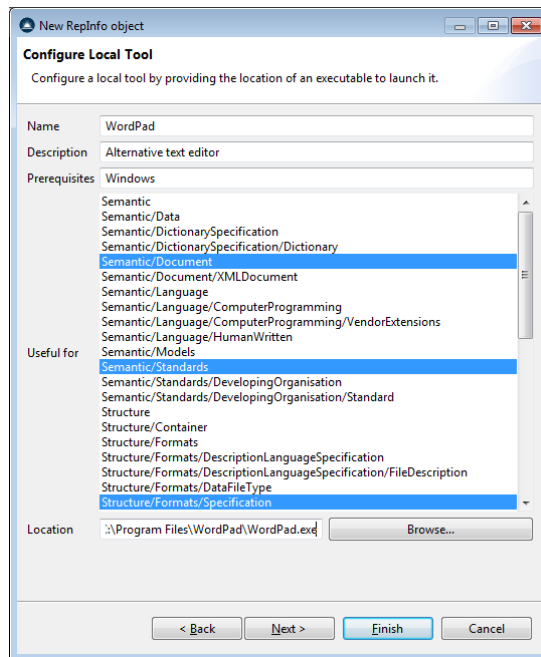


Figure 51 Configure local tool dialogue.

This brings up the Configure Local Tool dialogue (Figure 51) for you to provide information about the tool: name, description, prerequisites, categories of Representation Information that the tool may be used for, and the location of an executable file for launching the tool.

Once an installed tool is chosen or a new local tool is added, clicking Next will bring up a dialog for you to input extra information about the new RepInfo:

¹² Alliance for Permanent Access - <http://www.alliancepermanentaccess.org/>

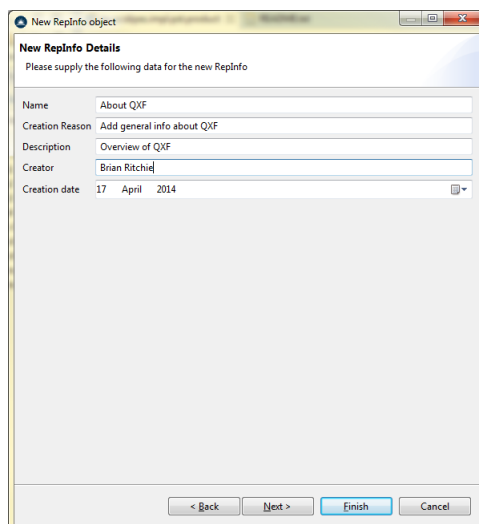


Figure 52 New Representation Information dialogue.

Name, Creation Reason, Description, Creator and Creation date map to default properties in the RIDO.

The final dialog in the wizard tells you that your chosen tool will be launched, and reminds you to back fill the location of the new RepInfo to the RIDO as described below once it has been created. The SCIDIP-ES convention is to use an HTTP addressable URL to reference RepInfo objects.

To complete the process, once you have created your RepInfo and stored it in the correct location, you should edit the RIDO associated with this new object and back fill the “RI Reference” property (which is the location of the new object; the RIT does not know this as it depends on the tool used to create it). The RI Name and RI Reference are mandatory and saving a new RepInfo RIDO to a Registry will fail if either is left blank.

Furthermore, every RI must have an associated Graph Label before it can be created in a registry. If you use Create New RepInfo on a blank RIDO, you can add a Graph Label by hand (to create a new Graph Label at the same time) or by searching for an existing Graph Label. If Create New RepInfo is used on a RIDO that has a Graph Label, the new RI will be associated with that Graph Label.

It is important to note that the process creates a RIDO local to the RIT only. Saving the RIN model at this point will save it to the RIT’s local workspace. Uploading (updating or creating) RepInfo objects to a Registry is a separate process which is described in Section 5.8.

5.6 Editing RINs and RIDOs by hand – guidelines

The RIT’s editor shares many features with the PNM editor. Building a RIN by hand is similar to building a PNM: new objects can be dragged onto the canvas and new relations can be added between them, while familiar form-based editors are used to capture properties. RINs downloaded from the Registry can be edited in the same ways. However, RINs that are to be uploaded (back) to a SCIDIP-ES Registry must satisfy a number of constraints, including:

- A RIDO should have at most one child from each of the OAIS types Semantic, Structure and Other. Children of the same type must be grouped under a Composite AND/OR relation object;

- A RIDO that has children is also a Graph Label as this describes a fragment of a virtual RIN graph. For a Graph Label, the “Graph Label Name” is mandatory (cannot be blank);
- A RIDO that has no children must at least have the properties “RI Name” and “RI Reference” set. The internal property “RI Graph ID” must also be set to the CPID of an existing RIL. As described in Section 5.5, RI Reference and RI Graph ID are **not** set by the New RepInfo Wizard, you should complete them manually after creating the new Representation Information object. The RI Graph ID can be set by searching for an existing Graph Label for the RIDO, or you can create a new Graph Label by hand.
- A child RIDO must be associated with a valid RepInfo object (see next section).

RINs constructed by searching and expanding from a Registry should satisfy these constraints already; but you need to bear these in mind when modifying RINs, especially when adding new RIDOs by hand. These constraints are validated by the operations to Update and Create RIDOs in a Registry. If any constraints are not met, the operation will report them, and the upload will not be possible until they have been addressed.

5.7 Example of Editing a RIN

The following example should demonstrate some of the principles of using the RIT to construct a new RIN. To illustrate, we have an incomplete RIN for the NetCDF QXF data, with a top-level RIDO and a single Structure child:

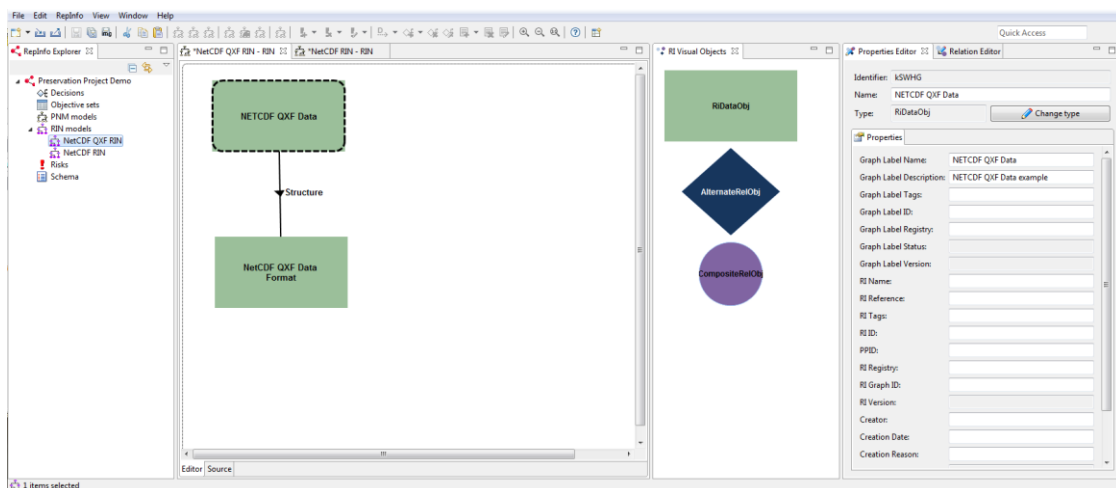


Figure 53 Editing a RIN using the visual editor.

In keeping with the above guidelines (Section 5.6), the top RIDO has “Graph Label Name” set (“Graph Label Description” is set too), and its child has “RI Name” and “RI Reference” set.

QXF is a variant of NetCDF data format and it can be used to help user to understand QXF. Therefore, we want to add NetCDF as a child and according to OAIS, it is categories as ‘Other’ RepInfo. NetCDF is a common data format and the requisite RepInfo objects already exist in the Registry and are available for re-use. So we can drag a new RIDO onto our canvas, select it, and then search the Registry for Graph Labels using the keyword “netcdf” (Figure 54).

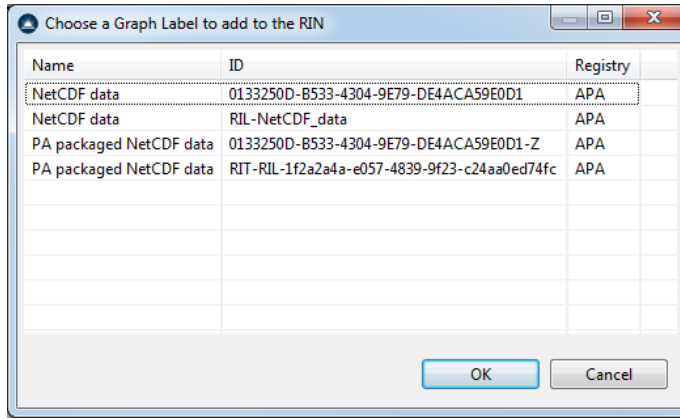


Figure 54 Search result for 'netcdf'.

Choosing the first NetCDF data adds the Graph Label for NetCDF to our canvas (Figure 55).

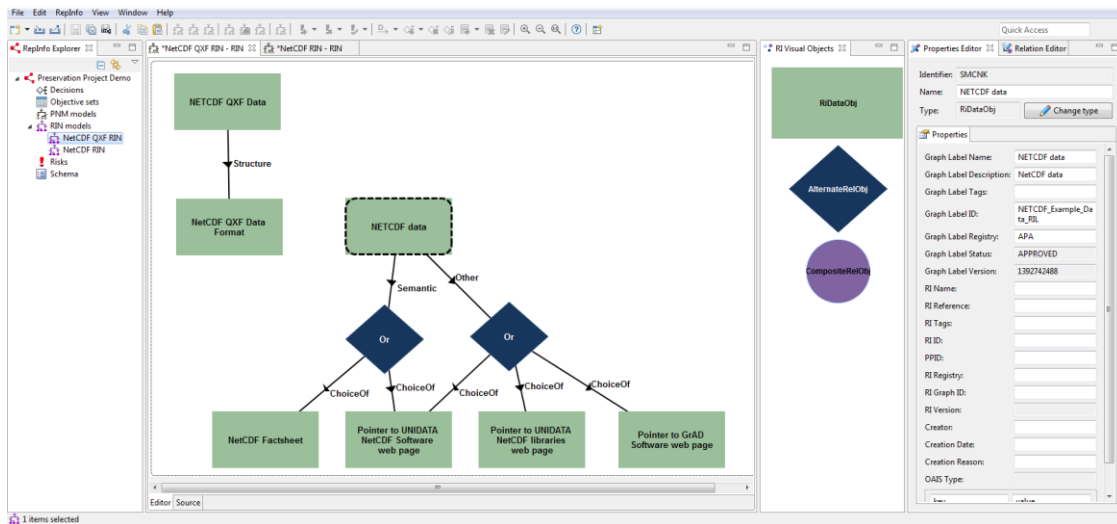


Figure 55 NetCDF RIL added to RIN.

To build our QXF RIN, we add an 'Other' relation from the QXF RIDO to the top node of the NetCDF graph fragment (Figure 56).

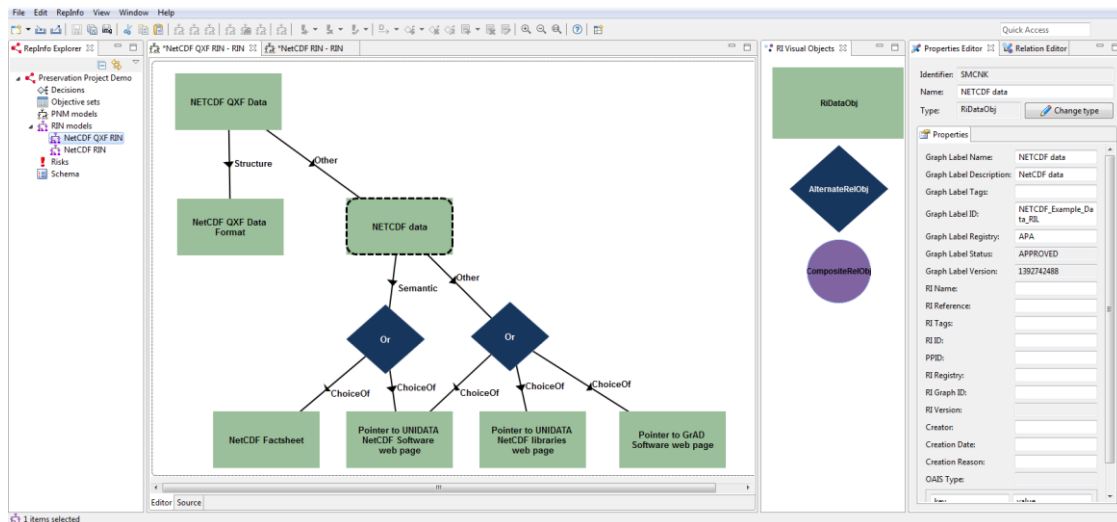


Figure 56 The completed NetCDF QXF data RIN.

The resulting RIN contains a combination of new and existing objects.

The RIN is not quite complete and cannot be uploaded to a Registry yet. There is a subtle point to note here: when we extract the NETCDF Data graph label from the registry, only the graph label which describes the structure of the RIN is retrieved and added to the RIDO. When we make this RIDO a child of another RIDO, we also need to choose (or create) a specific RepInfo object for it as well. The RepInfo is the actual content that is used to facilitate interpretation of the preserved data. There are two main ways to do this: select the “NETCDF data” RIDO and:

- Search Registry for RepInfo (which will search for RI that uses the NETCDF data Graph Label); or
- Use Create New RepInfo to define a new RepInfo object (see Section 5.5).

Alternatively, we could Search for a NetCDF RI object initially, by category or by ID (if we know it). In practice, it is often easier to search for a Graph Label by keyword and then search again to find RI objects that use it.

5.8 New, Changed and Unchanged RIDOS

When considering the relationships between a RIN model in the RIT and the RI and Graph Labels in a Registry, it is important to distinguish between (objects in) RIDOS that are new, changed or unchanged.

When an RI object or a Graph Label is retrieved from a Registry by Search, it is flagged as unchanged.

When a RIDO’s RI or Graph Label are created from scratch, or are retrieved from a registry and then modified, the RIDO is flagged as changed; this is shown by a question-mark in the RIDO (Figure 57).

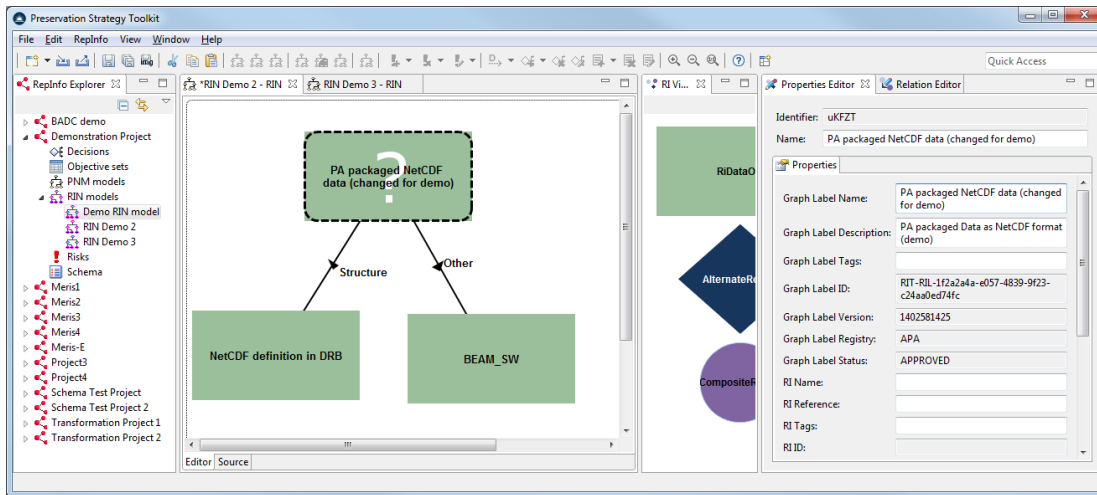


Figure 57 A Changed RIDO

A RIDO is considered to be changed when *either* its RI or Graph Label have been changed.

A RIDO's RI or Graph Label is considered *new* when there is no corresponding RI ID or Graph Label ID (so the details are not (yet) associated with an object in some Registry, but only exist in this model in the RIT.) This is the case when the details are created by hand, by Create New RepInfo, or if the RIDO has been Cloned (see Section 5.12). A RIDO is entirely new when its RI and Graph Label are both new (or one of these is absent).

We will see shortly that when a RIDO is uploaded to a Registry, its change flag is cleared (that is, it is flagged as unchanged).

It is worth considering in some detail what it means to “change” a RIDO. It is important to distinguish between *changing* properties of the existing RI or Graph Label, and *replacing* the RI or Graph Label with a different object. Changing properties of the existing objects affects only the current RIDO, but replacing objects can affect any parent RIDO as well.

Replacement of either the RI or Graph Label counts as a change to the RIDO. Recall that the RI can be replaced using Create New RepInfo; a Graph Label can be replaced by Cloning (see later).

For changing properties of the current RI, the situation is straightforward: changing the RI Name, RI Reference or RI Tags counts as a change to the RI (and to the RIDO).

For changing the properties of the Graph Label, the situation is more complicated. As with RI, changing the corresponding Graph Label properties in the RIDO's Properties panel (Graph Label Name, Description or Tags) counts as a change. However, changes to the Graph Label's *structure* count as changes to the Graph Label (and RIDO) as well; this includes:

- Adding or removing children or relations anywhere in the Graph Label's tree (that is, through any And/Or nodes as far as any child RIDOs); the Graph Label's “shape” has been changed, so this counts as a change to the Graph Label;
- Replacing the RI or Graph Label of any BEAM child RIDO. The leaves of a Graph Label's graph refer to both the RI *and to its associated Graph Label* by their IDs; so replacing either of these

constitutes a change to the Graph Label itself. Note that modifying the existing RI or Graph Label of a child (without changing the IDs) does *not* count as a change to the parent’s Graph Label.

If a RIDO has existing (non-new) RI but no Graph Label, the RI itself contains a reference to “its” Graph Label; but the Graph Label and its details are not added to the RIDO unless we Expand it. If instead of Expanding the RIDO, we add a new child by hand, the RIT interprets this as a request to **replace** the RI’s current Graph Label with a new one. To be sure, it asks us to confirm the replacement (see Figure 58).

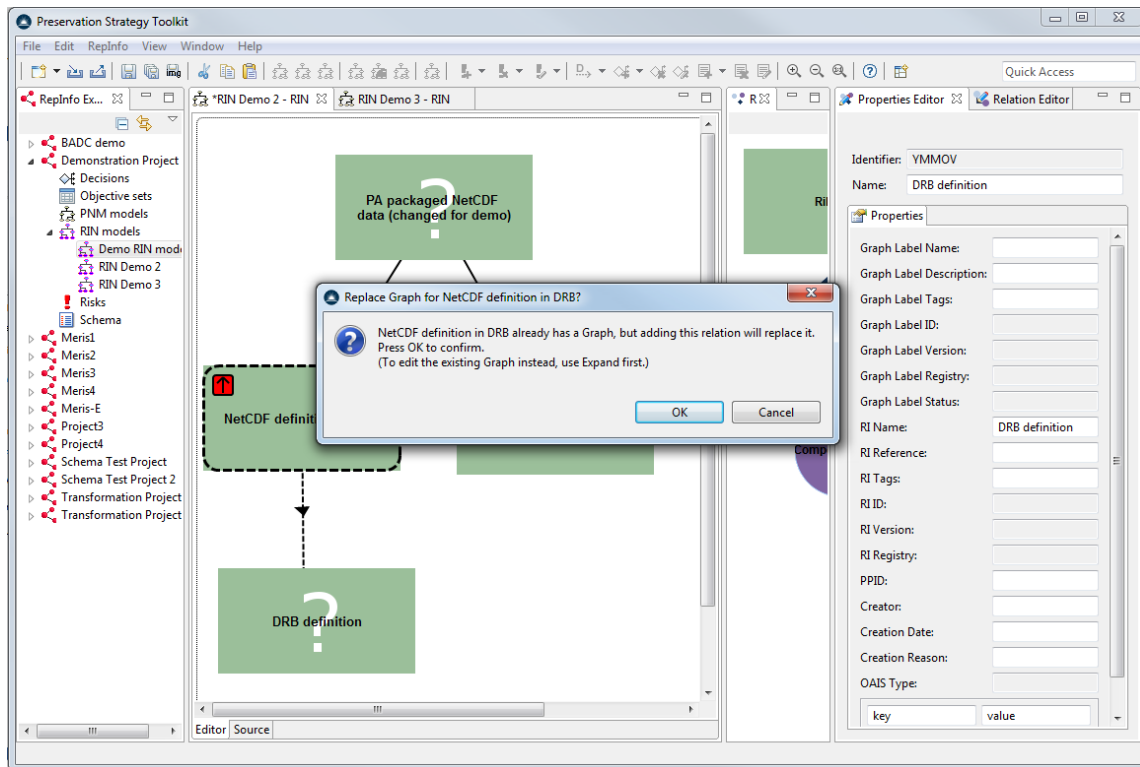


Figure 58 Confirmation dialog for replacement of an RI’s Graph Label

If we confirm at this point, the RI’s Graph Label ID will be cleared. The RI (and the RIDO) will be flagged as changed (as will any parent, by the second rule above).

Replacing an RI’s Graph Label is a significant commitment: we are saying that this RI should now have a different Graph Label (and its reference to the original Graph Label will be lost). If the RI is in common use elsewhere (e.g. if it is referred to from other Graph Labels), this change may affect many other people. It may be worth considering Cloning the RIDO anyway (see later), so that these changes are (subsequently) uploaded to the Registry as new objects.

If we intend instead to **edit** the RI’s associated Graph Label, we must Expand the RIDO first, so that the graph is added to our RIN. Any subsequent edits to the graph will be treated as changes to the existing Graph Label.

Editing existing Graph Labels is also a significant commitment: once the changes are updated back to the original registry, the Graph Label will be changed for everyone. Existing users may need to update

their data to pick up the changes. This includes the RIT, if the Graph Label has been used in other models, projects or workspaces (or even elsewhere in the same model).

5.9 Updating and Creating in a Registry - Overview

Once the user is satisfied with a RIN they have developed or modified, they can upload it to a Registry or Registries. (As a RIN is a network of objects, those objects may come from different Registries. Indeed, a single RIDO may contain an RI and a Graph Label from different registries.)

When uploading a RIN to a Registry (or Registries), we need to consider as separate cases whether existing objects (RIs and Graph Labels) are being *updated* or whether new objects are to be *created*. Broadly speaking, if a RIDO or RIN has been populated or created by searching or expanding, and subsequently modified, it should be Updated back to its source Registry (or Registries); if a RIDO is created using the New RepInfo wizard as described in Section 5.5, has been Cloned (see Section 5.12) or if it has been created by hand, then it must be Created in a Registry.

The RIT provides separate Update and Create operations. Their interfaces are described in subsequent sections; we give a brief overview here.

Create can only be applied to a RIDO that is entirely new; that is, any RI or Graph Label details that it contains must be new (i.e. must not have IDs). It operates on a single RIDO. If the RIDO has a subgraph, then the operation only considers the graph down to the “next level” of RIDOS, i.e. the immediate RIDO children (which form the leaves of its Graph Label). Create asks the user to select a target registry, and then creates a new RI object and/or Graph Label in that registry. The new IDs (and other details) are added to the RIDO, and it is no longer new (and indeed no longer flagged as changed).

One condition of Create is that the children must all exist (not be new) and must not be modified. When a RIN contains multiple new RIDOS, the children must be created before their parents.

Update operates on the entire subgraph, ensuring that children are updated before their parent. Update will also create any new objects that it finds. Update determines which objects (RI and Graph Labels) in the subgraph have been changed, and also any new objects that need to be created. The list is shown to the user, who must supply write credentials to all registries involved, and select a target registry for each new object. Changed objects are updated in their original registry, and new objects are created in the registry selected by the user.

Though Update applies to an entire subgraph, there is a condition that the initial RIDO must be changed but not entirely new; so Update must always start from a changed RIDO.

Update’s effects are wider-ranging than Create’s, as it operates over the entire subgraph, and both updates modified objects and creates new ones. It should always be born in mind that changes committed by Update to an RI or a Graph Label will affect all other users of that RI or Graph Label. If your changes are specific to your design, then you should consider Cloning first, so that the original object is not changed.

The Update and Create operations can only be performed on RINs that satisfy a number of validation constraints. Simple constraints are used to disable the menu options: for Create, the RIDO must be entirely new; for Update the RIDO must not be entirely new, but must be flagged as changed. Even

when these conditions are met and the menu item is available, further validity tests are performed; if any fail, they are reported back to the user; otherwise, the Update or Create proceeds to the next step.

For Update, each RIDO in the subgraph must satisfy the following constraints:

- If it has RI details (RI Name, RI Reference), then either there must be Graph Details or the RI must be associated with an existing Graph Label (this is true for any RI retrieved from a registry).
- If it has Graph Label details (Graph Label Name), then the graph below the RIDO must satisfy certain structural constraints (see below).
- If a RIDO's RI or Graph Label is changed and not new, then its Version must match the current value in the Registry; if the Version does not match then the object (and RIDO) are based on an earlier version, and cannot be updated (the RIDO must be Cloned or removed).

For Create, the RIDO must satisfy the following constraints:

- If the RIDO has RI Name set, then RI Reference must also be set, but RI ID must NOT be set (it will be assigned as part of the creation process);
- If the RIDO has RI Name set, then either the RIDO should have a new Graph Label too, or the internal property RI Graph ID must also be set. (The second case can be achieved by Cloning a RIDO that has RI details.)
- If the RIDO has Graph Label Name set, then the Graph Label ID must not be set (it must be new), and the graph below the RIDO must satisfy certain structural constraints (see below);
- If the RIDO has children, then Graph Label Name must be set;
- Any RIDO children must not be changed (and must not be new).

For both Update and Create, the Graph Label, or rather the graph fragment that represents it, must satisfy a number of conditions (some of which have already been mentioned earlier):

- The RIDO should not have more than one of each of the Semantic, Structure or Other relations (note, it does not need to have all three);
- Any Composite (And) or Alternate (Or) nodes must have two or more children;

Remember that RIN models can be saved locally by the RIT; so you can save a model as work-in-progress, exit the RIT, then return to it later to continue working on it, and only upload to the Registry once you are satisfied that it is complete.

5.10 Updating to a Registry

This operation can be used to update changes to a RIDO back to its source Registry (or Registries). As described in Section 5.1, a RIDO is the basic building block in a RIN and aggregates different types of information objects from the Registry. Thus, a RIDO can contain details of both a RepInfo object and its Graph Label, which could come from different Registries.

To Update, select the RIDO (by clicking on it; a dashed outline should appear), then:

- Right click on it to bring up the context menu, or
- Select "Update in Registry" from the main RepInfo menu item

If “Update in Registry” is not active in the menu, then the RIDO does not meet basic criteria for the operation. Update is only available on RIDOS that are changed but not new.

The RIT will check that the RIDO and its subgraph (if any) satisfy the update constraints. If it fails, a brief report dialog is shown, e.g.

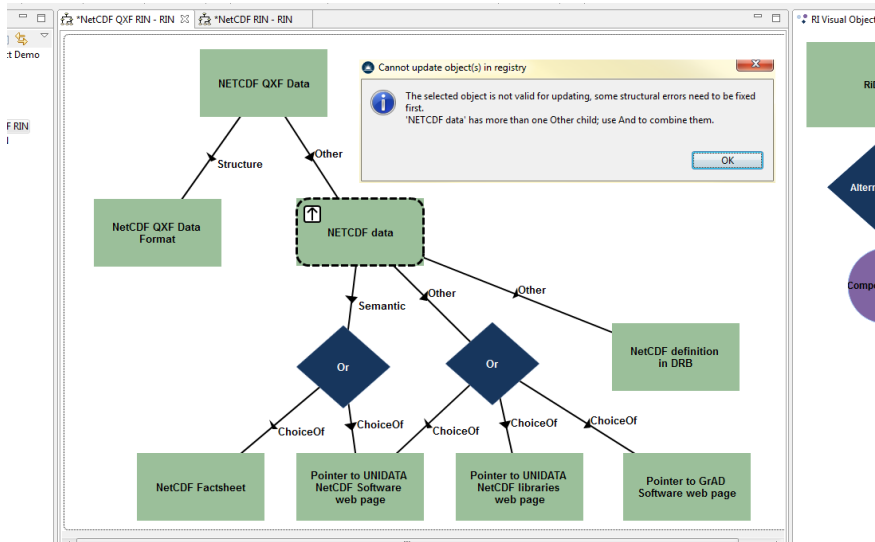


Figure 59 Error message on failing update validation.

(To help pinpoint an error, the error message will normally refer to a specific RIDO by its display name. In some cases (including for And/Or nodes) this is not possible, and the internal RIDO ID will be used instead. Pausing the mouse over a graphical object will bring up a pop-up display of some of its properties, including its internal ID, which could be useful when tracking down validation failures.)

If the RIDO passes the validation check, the RIT automatically determines which objects are to be updated, and in which Registries; and then displays a dialogue with the information, e.g.:

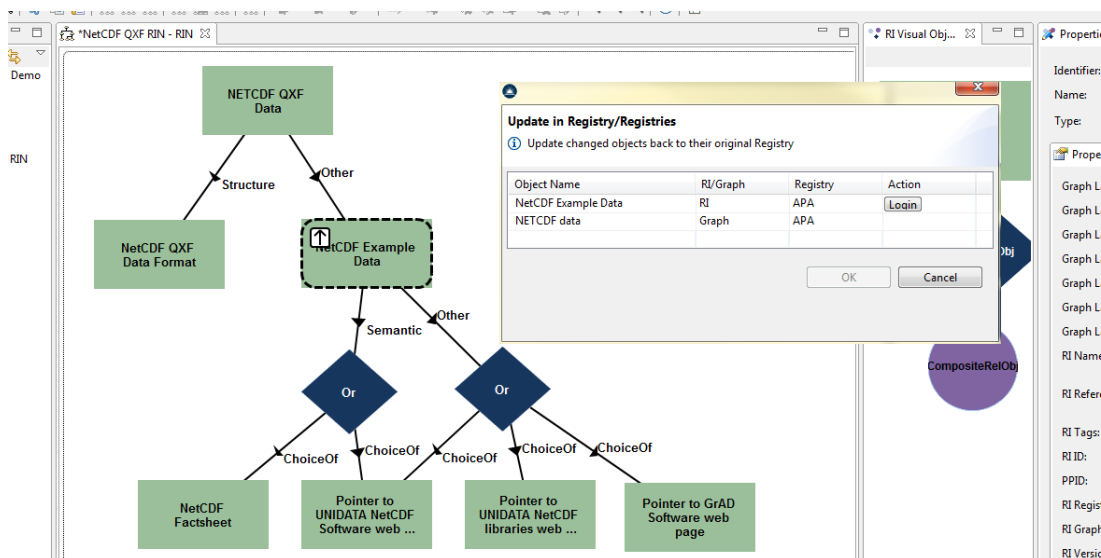


Figure 60 Update dialogue.

If Update finds an object that is new, the dialog will include a row that allows the user to choose the target registry in which to create it. This is shown in Figure 61; here, we have added a third child with new RI to the And under the Semantic branch (adding the child has changed the top RIDO's Graph Label, so it is also flagged as changed). When we select the top RIDO and choose Update, the dialog includes an entry for the new child's RI, with a drop-down list that we can (and must) use to choose its target Registry.

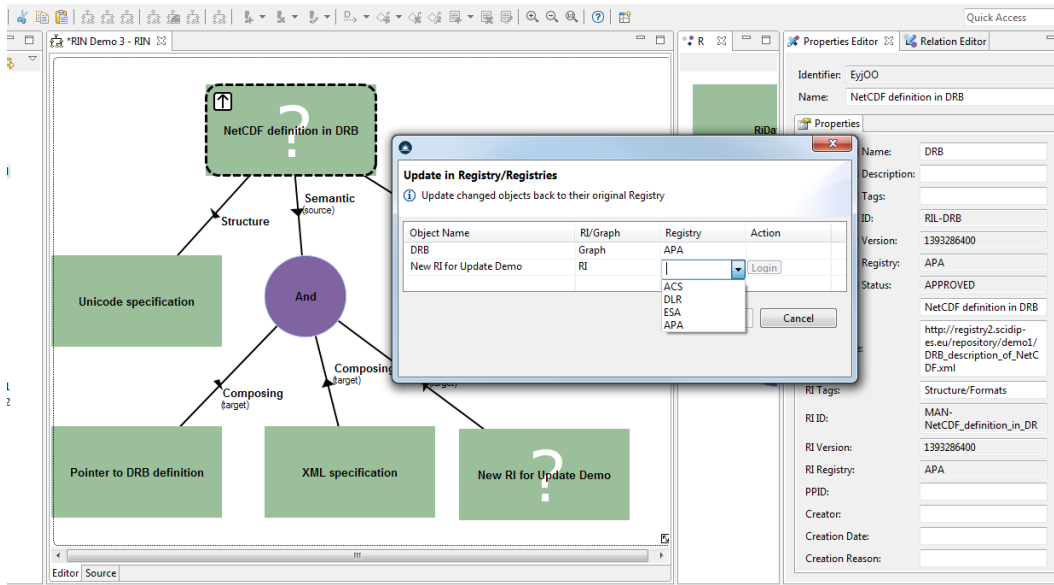


Figure 61 Update dialog with new RI

Users must have write authority and be logged in for each of the Registries involved to carry out update. If this is not the case, the dialogue will display a Login button on the first row that refers to that Registry; and the OK button will be disabled.

Clicking the Login button brings up a login dialog for users for to supply credentials for that Registry (the default Registry Serializer implementation uses simple name and password).

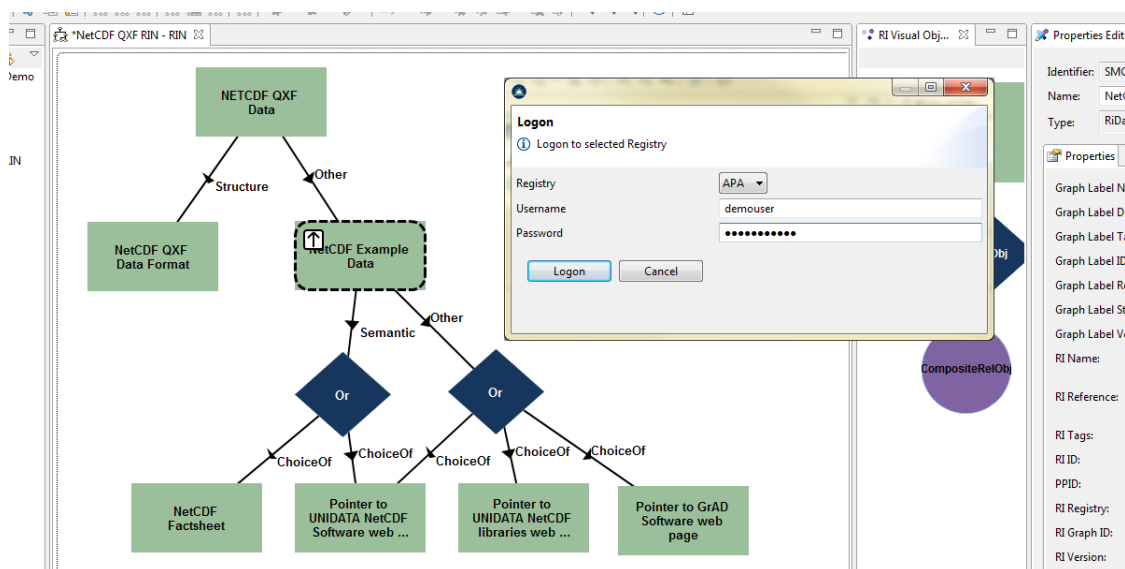


Figure 62 Registry login dialogue.

Once authenticated for write access, the Login button for that Registry disappears. When all registries are write-authenticated, the OK button is enabled.

Click OK to perform the Update for the selected RIDO in their source registries.

A short error message is displayed if the operation failed. The PST log file may contain more details (see Section 7).

As Update will change existing objects in the Registries, care should be taken to minimise the disturbance to others who may be using the current versions of the objects.

When Update is performed, each updated object is assigned a new version number (which the RIT records in the RI and Graph Label Version fields). However, if an RI object or Graph Label is updated via one RIDO, the version number in any other RIDO that refers to the same object will *not* be updated. As part of the Upload validation includes a check that the version number is current, this means that such out-of-date RIDOS cannot be uploaded. At present, if Upload reports that a RIDO (or its objects) are out of date, the only solutions are to Clone the RIDO (so that new objects will be created from it, see later) or to remove it from the RIN.

5.11 Creating in a Registry

This operation can be used to upload a new RIDO's objects to a Registry.

To Create, select the RIDO (by clicking on it; a dashed outline should appear), then:

- Right click on the RIDO to bring up the context menu, or
- Select "Create New in Registry" from the main ReplInfo menu item.

The RIT will check that the RIDO and its subgraph (if any) satisfy the creation constraints. If it fails, a brief report dialog is shown, e.g.

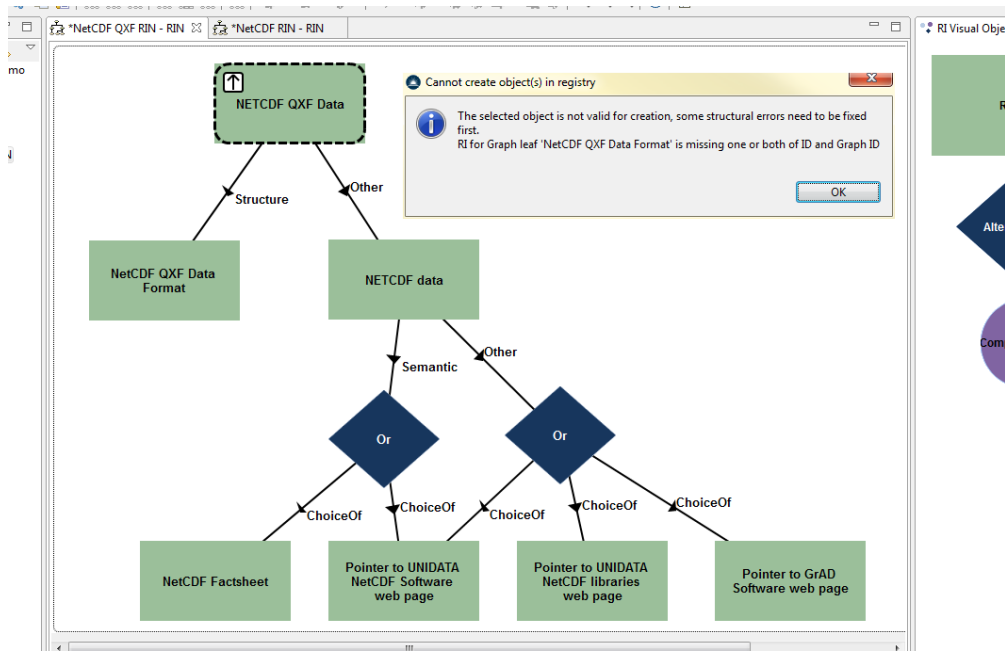


Figure 63 Error message on failing creation validation.

If the RIDO passes the validation, the RIT then asks the user to choose a target Registry for creating the RIDO's objects:

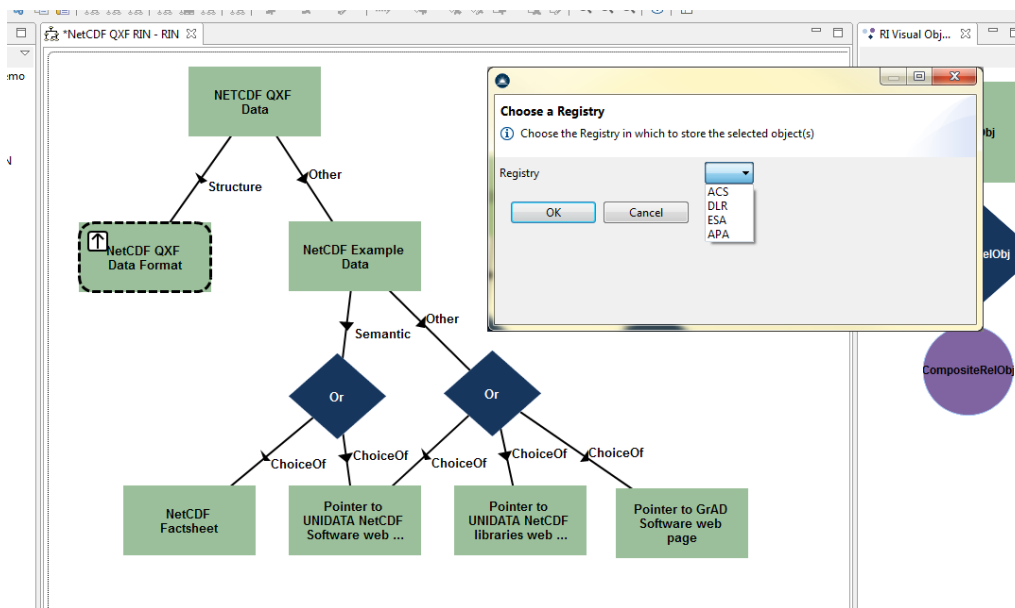


Figure 64 Choose target registry dialogue.

If the user does not have write authority for the chosen Registry, they will be invited to log in:

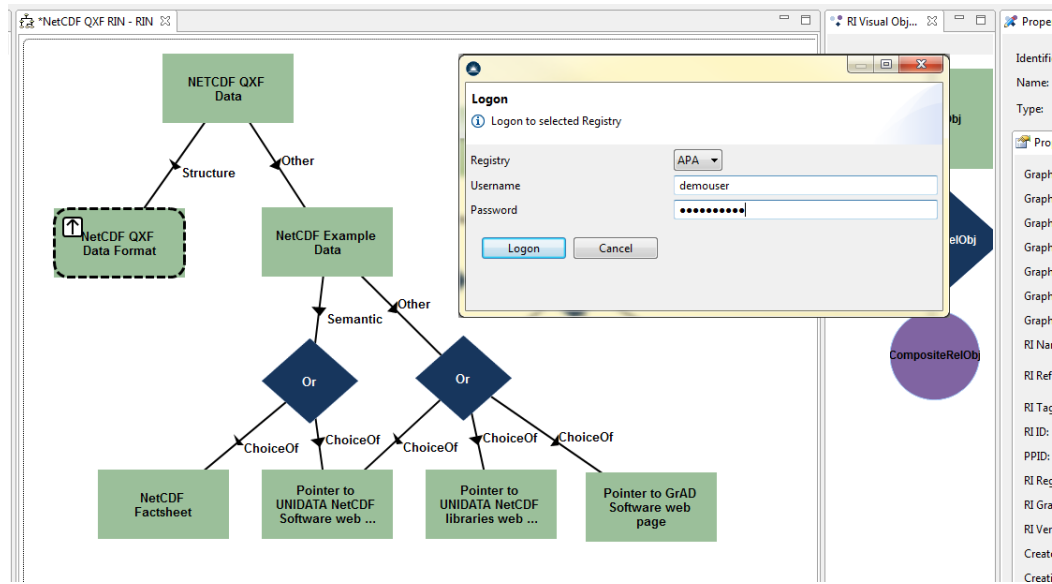


Figure 65 Registry login dialogue.

Once the user has authenticated for write access, the RIDO's RI and/or Graph Label are created in the chosen Registry. The RIDO details are updated to reflect this (RI ID, RI Registry and/or Graph Label ID and Graph Label Registry will be set).

Note: It may be necessary to de-select and then re-select the RIDO to force a refresh of the Properties Editor.

A short error message is displayed if the operation failed. The PST log file may contain more details (see Section 7).

5.12 Cloning a RIDO

When we retrieve RI or a Graph Label from a Registry and then make modifications, we might decide that we would like to save the modified object as a new object, rather than updating the original object. (This may be forced on us, for example we may not have write access to the original Registry, or the RIDO may be out of date as the RI or Graph Label have been updated from elsewhere.)

The Clone operation allows us to do this. Performing Clone on a RIDO “detaches” its RI and Graph Label details from the original Registry by clearing the IDs (and other fields such as Registry, Status and Version). The resulting RIDO is now entirely new, and (assuming the validation conditions are met) can be Created to upload new objects to a selected Registry.

Note that Clone does not create a copy of the RIDO in the RIN; it creates new copies of the RIDO's objects.

Clone can be applied to any RIDO, so long as it is not entirely new (in which case, there is nothing to clone). To Clone a RIDO, select it, and choose “Clone RIDO” from the context menu (Figure 66).

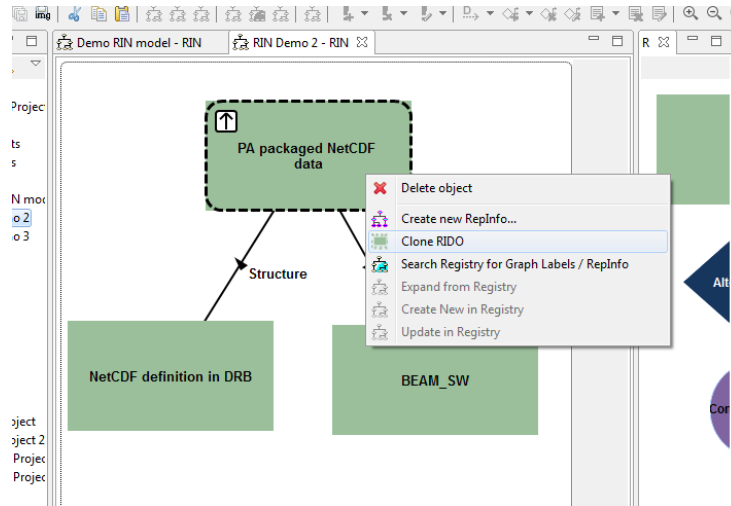


Figure 66 Cloning a RIDO

Afterwards, the ID (and Version, Registry and Status) fields will be cleared / reset; the RIDO will be flagged as Changed (as it now contains new objects). (It may be necessary to de-select and re-select the RIDO to update the Properties panel and see the changes.) Both the RI and Graph Label (if present) are now new. If the RIDO has a parent, then it will now be flagged as changed as well.

When a RIDO is Cloned, we should consider changing the Name or other details of its RI or Graph Label so that (once created) they can be distinguished from the originals in subsequent Searches.

6 Reference Manual

6.1 Keyboard shortcuts

HOTKEY	DESCRIPTION
Ctrl + S	Save changes in the current editor
Ctrl + M	Maximise the editor view

6.2 Command-line commands

N/A. PST/RIT is a GUI tool.

6.3 Public APIs

None available. PST/RIT is a GUI tool.

7 Troubleshooting Common Issues

The PST/RIT records information, warnings and errors in a file in its workspace (the folder chosen by the user on application start up). The file is in `.metadata\.log`. The information contains error traces more relevant to developers rather than end-users.

7.1 Toolbar menu items are not enabled properly

When changes are made in a model editor are not always picked up by the toolbar menu items. For example, when Use Solution... is used on a PNM model that had no Solution selected before, Solution-based menu items (e.g. Show cost of current solution) may still not be enabled. There is a workaround for this, which is to switch to another editor (e.g. by opening a Risk), and then go back to the original PNM editor.

7.2 RIN model: RIDO property panel may not be updated

After some operations on a selected RIDO, the RIDO's property panel may not be updated to reflect changes. Click away from the RIDO and then select it again; this should cause the panel to update.

Annex A. References

[CON1] Conway, E., Dunckley, M.J., Giaretta, D. and McIlwrath, B. 2009. Preservation Network Models: Creating Stable Networks of Information to Ensure the Long Term Use of Scientific Data. *Proc. Ensuring Long-Term Preservation and Adding Value to Scientific and Technical Data*, del Castillo, Madrid, Spain, 01-03 Dec 2009. Available at http://epubs.cclrc.ac.uk/bitstream/4314/PV09_Conway_PNM.pdf

Annex B. Figures and Tables

B.1. List of Figures

Figure 1 Example PNM for BADC MST data.	8
Figure 2 Overview of PS/RIT design.	9
Figure 3 PS/RIT project model.....	10
Figure 4 Defining a PST workspace.	14
Figure 5 Main GUI components – PST perspective.	15
Figure 6 Create a new preservation project using the main menu.	16
Figure 7 Create a preservation project using the toolbar icon.	16
Figure 8 Create a new preservation project using the context menu.	16
Figure 9 New project dialogue.	17
Figure 10 Project explorer.....	18
Figure 11 Schema viewer.	18
Figure 12 New PNM Model dialog.	19
Figure 13 The PNM drawing canvas.....	19
Figure 14 PNM drawing canvas with two PNM objects.....	20
Figure 15 Two PNM objects with a ‘depA’ relationship.....	20
Figure 16 PNM properties editor.	21
Figure 17 Relation editor.....	22
Figure 18 Objective set editor.....	22
Figure 19 Objective editor.....	23
Figure 20 Working with objective sets via the relation editor.	24
Figure 21 Choose objective set dialogue.	24
Figure 22 Decision object/s associated with the project.....	25
Figure 23 Decision editor.	26
Figure 24 Risks associated with a PNM object.....	27
Figure 25 Export a PNM model.	28

Figure 26 Example of a Transformation object.....	29
Figure 27 Example PNM with alternative preservation solutions.	30
Figure 28 An incomplete PNM solution showing undecided nodes.	31
Figure 29 PNM solution showing the user chosen nodes.....	32
Figure 30 PNM solution showing more than one alternative nodes are chosen.	32
Figure 31 About to show cost of current solution	33
Figure 32 Include or exclude Transformation costs?.....	34
Figure 33 Cost (including transformations)	34
Figure 34 Saving a solution.	35
Figure 35 Displaying the cost of a chosen solution.....	35
Figure 36 Displaying cost for the solution with Vista included.....	36
Figure 37 The RIT perspective.	37
Figure 38 A simple RIDO.....	37
Figure 39 A small RIN.	38
Figure 40 Simple RIN with child selected.....	39
Figure 41 Graph Label with more complex structure	39
Figure 42 Complex RIN with multiple Graph Labels	40
Figure 43 Create a new RIN model.	41
Figure 44 Search RepInfo Registry dialogue.	41
Figure 45 Registry Search result.....	42
Figure 46 RIN representing the retrieved MERIS L1 data Graph Label.....	43
Figure 47 Search on a RIDO that has a Graph Label (but no RI)	43
Figure 48 RIN after expanding the GIS3D Model Example leaf RIDO.	44
Figure 49 Specify a category for the new Representation Information object.	45
Figure 50 Dialogue for choosing or adding a tool for creating Representation Information.	45
Figure 51 Configure local tool dialogue.	46
Figure 52 New Representation Information dialogue.	47
Figure 53 Editing a RIN using the visual editor.	48

Figure 54 Search result for ‘netcdf’.....	49
Figure 55 NetCDF RIL added to RIN.....	49
Figure 56 The completed NetCDF QXF data RIN.....	50
Figure 57 A Changed RIDO	51
Figure 58 Confirmation dialog for replacement of an RI’s Graph Label	52
Figure 59 Error message on failing update validation.	55
Figure 60 Update dialogue.	55
Figure 61 Update dialog with new RI	56
Figure 62 Registry login dialogue.	56
Figure 63 Error message on failing creation validation.	58
Figure 64 Choose target registry dialogue.	58
Figure 65 Registry login dialogue.	59
Figure 66 Cloning a RIDO.....	60

B.2. List of Tables

No table of figures entries found.

Annex C. Terminology

ACRONYM	DESCRIPTION
AIP	Archival Information Package
CPID	Curation Persistent Identifier
Graph Label	Synonym for RIL
GUI	Graphical User Interface
PNM	Preservation Network Model
PST	Preservation Strategy Toolkit
RI, RepInfo	Representation Information
RIL	RepInfo Label
RIT	RepInfo Toolkit

Annex D. Sample PNM Schema file

Ontology {

```
ObjectiveAttributes {
    name {
        editable=yes
    }
    title {
        editable=yes
    }
    description {
        editable=yes
    }
    preservationFunction {
        editable=no
        1=render
        2=describe
        3=transform
        4=implement
    }
    uniqueIdentifier {
        editable=yes
    }
    version {
        editable=yes
    }
    location {
        editable=yes
    }
    seeAlso {
        editable=yes
    }
}
```

```
RiskAttributes {  
    name {  
        editable=yes  
    }  
    title {  
        editable=yes  
    }  
    description {  
        editable=yes  
    }  
    seeAlso{  
        editable=yes  
    }  
    rating {  
        editable=yes  
        numerical=yes  
    }  
}
```

```
DecisionAttributes {  
    name {  
        editable=yes  
    }  
    title {  
        editable=yes  
    }  
    description {  
        editable=yes  
    }  
    created {  
        editable=yes  
    }  
}
```

```
seeAlso{
    editable=yes
}
decision {
    editable=yes
}
failure {
    editable=no
    1=critical
    2=partial
    3=within tolerance
}
}
```

```
PnmObjectProperties {
    title{
        editable=yes
    }
    uniqueIdentifier{
        editable=yes
    }
    version{
        editable=yes
    }
    description{
        editable=yes
    }
    seeAlso{
        editable=yes
    }
    cost{
        editable=yes
        numerical=yes
    }
}
```

```
}  
transformation object{  
  editable=no  
  1=no  
  2=yes  
}  
transformation cost{  
  editable=yes  
  numerical=yes  
}  
cost model ref{  
  editable=yes  
}  
}
```

```
Types {  
  FunctionalObj=DigitalDataObj  
  OtherObj=FunctionalObj  
  SemanticObj=FunctionalObj  
  StructuralObj=FunctionalObj  
}
```

```
Dependencies {  
  FunctionalRel {  
    start=DigitalDataObj  
    end=FunctionalObj  
  }  
  StructuralRel {  
    start=DigitalDataObj  
    end=StructuralObj  
  }  
  SemanticRel {  
    start=DigitalDataObj
```

```
        end=SemanticObj
    }
    otherRel {
        start=DigitalDataObj
        end=OtherObj
    }
    RelatedObject {
        start=DigitalDataObj
        end=DigitalDataObj
    }
    CompositeRel {
        start=DigitalDataObj
        end=CompositeRelObj
    }
    Composing {
        start=CompositeRelObj
        end=DigitalDataObj
    }
    AlternateRel {
        start=DigitalDataObj
        end=AlternateRelObj
    }
    ChoiceOf {
        start=AlternateRelObj
        end=DigitalDataObj
    }
    Choiceof {
        start=AlternateRelObj
        end=CompositeRelObj
    }
}
}
```


Annex E. Sample RIN schema file

Ontology {

ObjectiveAttributes {

name {

 editable=yes

}

title {

 editable=yes

}

description {

 editable=yes

}

preservationFunction {

 editable=no

 1=render

 2=describe

 3=transform

 4=implement

}

uniqueIdentifier {

 editable=yes

}

version {

 editable=yes

}

location {

 editable=yes

```
}  
seeAlso {  
    editable=yes  
}  
}
```

```
RiskAttributes {  
    name {  
        editable=yes  
    }  
    title {  
        editable=yes  
    }  
    description {  
        editable=yes  
    }  
    seeAlso{  
        editable=yes  
    }  
    rating {  
        editable=yes  
        numerical=yes  
    }  
}
```

```
DecisionAttributes {  
    name {  
        editable=yes
```

```
}  
title {  
    editable=yes  
}  
description {  
    editable=yes  
}  
created {  
    editable=yes  
}  
seeAlso{  
    editable=yes  
}  
decision {  
    editable=yes  
}  
failure {  
    editable=no  
    1=critical  
    2=partial  
    3=within tolerance  
}  
}
```

```
PnmObjectProperties {  
ril.displayName{  
    editable=yes  
}  
}
```

```
ril.description{  
    editable=yes  
}
```

```
ril.cpid{  
    editable=no  
}
```

```
ril.registry{  
    editable=no  
}  
ril.categories{  
    editable=yes  
}
```

```
ril.typeCpid{  
    editable=no  
}  
ril.version {  
    editable=yes  
}  
ril.status{  
    editable=yes  
}
```

```
manifest.resourceName{  
    editable=yes  
}
```

```
manifest.location{  
    editable=yes  
}
```

```
manifest.rilCpid{
```

```
    editable=yes
  }
  manifest.registry{
    editable=yes
  }
manifest.cpid{
  editable=yes
}
  manifest.categories{
    editable=yes
  }
manifest.ppid{
  editable=yes
}
  manifest.version{
    editable=yes
  }
pdi.creator{
  editable=yes
}
pdi.reason{
  editable=yes
}
pdi.date{
  editable=yes
}
  ri.baseType{
    editable=yes
```

```
}  
ri.cpid{  
    editable=no  
}  
ri.typeCpid{  
    editable=no  
}  
}  
  
Types {  
    FunctionalObj=RiDataObj  
    OtherObj=FunctionalObj  
    SemanticObj=FunctionalObj  
    StructureObj=FunctionalObj  
}  
  
Dependencies {  
    Structure {  
        start=RiDataObj  
        end=RiDataObj  
    }  
    Semantic {  
        start=RiDataObj  
        end=RiDataObj  
    }  
    Other {  
        start=RiDataObj  
        end=RiDataObj
```

```
}  
StructureAnd {  
    start=RiDataObj  
    end=CompositeRelObj  
}  
SemanticAnd {  
    start=RiDataObj  
    end=CompositeRelObj  
}  
OtherAnd {  
    start=RiDataObj  
    end=CompositeRelObj  
}  
CompositeRel {  
    start=RiDataObj  
    end=CompositeRelObj  
}  
Composing {  
    start=CompositeRelObj  
    end=RiDataObj  
}  
StructureOr {  
    start=RiDataObj  
    end=AlternateRelObj  
}  
SemanticOr {  
    start=RiDataObj  
    end=AlternateRelObj
```

```
}  
OtherOr {  
    start=RiDataObj  
    end=AlternateRelObj  
}  
AlternateRel {  
    start=RiDataObj  
    end=AlternateRelObj  
}  
ChoiceOf {  
    start=AlternateRelObj  
    end=RiDataObj  
}  
Choiceof {  
    start=AlternateRelObj  
    end=CompositeRelObj  
}  
}  
}
```

Annex F. Sample registryserializer.ini file

```
[network]  
proxySet = true  
http.proxyHost = wwwcache.rl.ac.uk  
http.proxyPort = 8080
```