pure::variants - Connector for IBM Engineering Test Management Manual

pure-systems GmbH

Version 6.0.5.685 for pure::variants 6.0

2024

Table of Contents

1. Introduction

pure::variants - Connector for IBM Engineering Test Management enables Quality Manager users to manage test artefacts variability using pure::variants. By coupling pure::variants and Quality Manager, knowledge about variability and variants can be formalized, shared and automatically evaluated. This enables getting answers for questions about valid combinations of test artefacts in product variants quickly; permits easy monitoring of planned and released product variants at the test artefacts level and also permits very efficient production of variant-specific test plans out of the test artefacts repository.

The manual is available in online help inside the installed product as well as in printable PDF format. Get the PDF here.

1.1. About this manual

The reader is expected to have basic knowledge about and experiences with both tools, IBM Rational Quality Manager and pure::variants. Please consult their introductory material before reading this manual.

1.2. Software Requirements

The following software has to be present on the user's machine in order to support the pure::variants - Connector for IBM Engineering Test Management:

IBM Rational Quality Manager:	IBM Rational Quality Manager 6.0.6.1 - 7.0.3 is required. Compatibility with other IBM Rational Quality Manager releases is not guaranteed.
Oracle Java Runtime:	At least Java Runtime Environment 1.8 is required.

The pure::variants - Connector for IBM Engineering Test Management is an extension for pure::variants and is available on all supported platforms.

IBM Rational Quality Manager and pure::variants communicate using the HTTP(S) protocol.

1.3. Installation

Please consult section **pure::variants Connectors** in the **pure::variants Setup Guide** for detailed information on how to install the connector (menu **Help** -> **Help Contents** and then **pure::variants Setup Guide** -> **pure::variants Connectors**).

2. Using the Connector

2.1. Starting pure::variants

Depending on the installation method used either start the pure::variants-enabled Eclipse or under Windows select the **pure::variants** item from the **program** menu.

If the **Variant Management** perspective is not already activated, do so by selecting it from **Open Perspective -> Other** in the **Window** menu.

2.2. Authentication

To use the the connector it is always required to be logged-in to the Quality Manager application. Currently there are two authentication mechanisms supported:

- Form-based
- OpenID Connect (for Single-Sign-On)

For both mechanisms the user will be prompted with a login dialog, which expects the user credentials. In case of Single-Sign-On a browser-based login dialog will be shown.

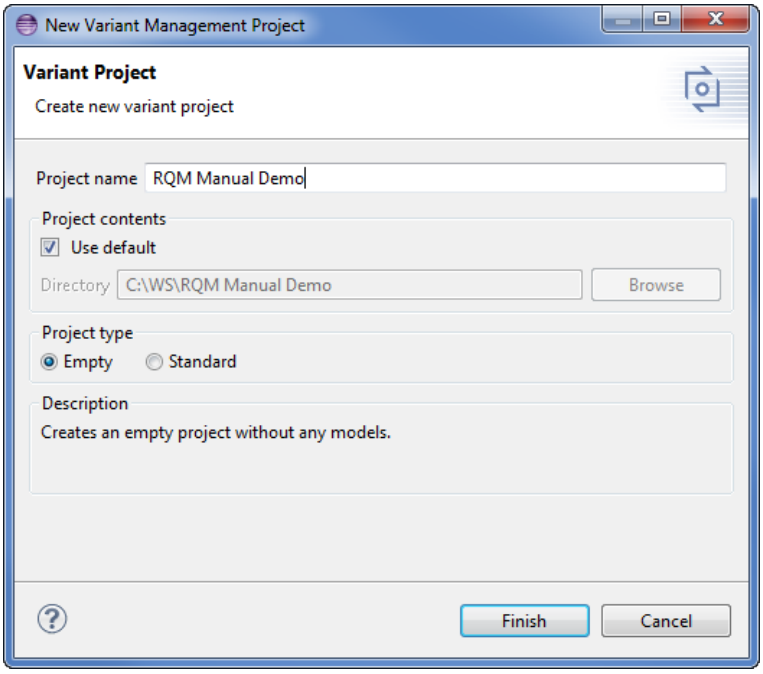
2.3. Creating the Initial Model(s)

The first step is always to create the corresponding family model for the relevant Quality Manager test plans of a project. These initial family models serve as starting points for using existing variability information. The import procedure has to be executed **only once** for a Quality Manager test plan. Each Quality Manager project with selected test plans is represented by one pure::variants family model.

The Jazz user required some specific permissions to import as well as to transform Test artifacts into pure::variants. See Section 3.6, “Minimal permissions required for ETM stream and variant enumeration transformation”

Before the actual import can be started, a Variant Management project has to be created, where the imported models will be stored. Select **Project** from **New** in the **File** menu. Choose **Variant Projects** below **Variant Management** in the first page of the "New project" wizard. Choose a name for the project and select **Empty** as project type (see Figure 1, “Creating an empty Variant Management project for Quality Manager test plan import”)

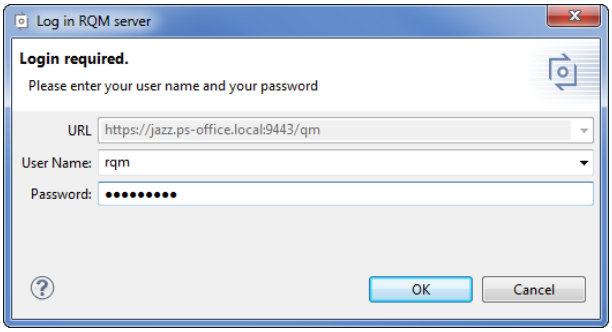
Figure 1. Creating an empty Variant Management project for Quality Manager test plan import



Import is started by selecting the import action either in the context menu of the Project view or with **Import** menu in the **File** menu. Select **Variant Models or Projects** and press **Next** . On the following page select *Import RQM Test Plans* .

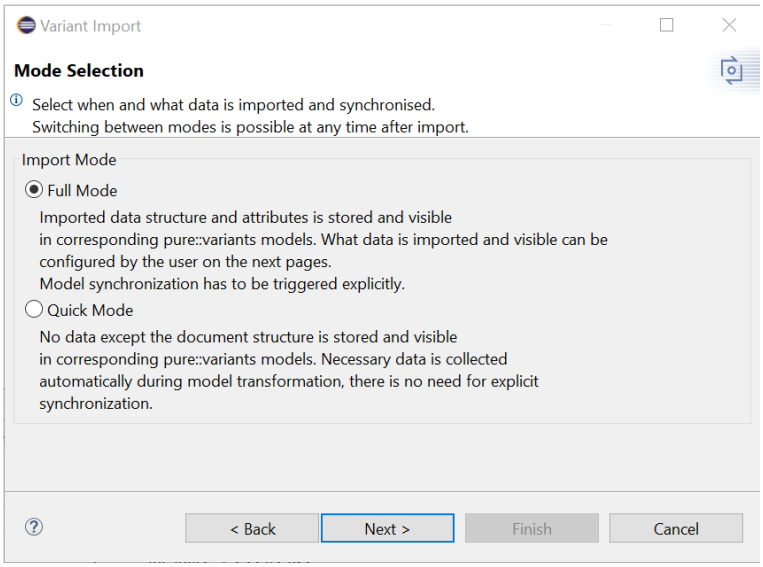
The server selection page appears and asks for Quality Manager server address. You can test the connection to the server pressing *Test Connection*. A login dialog pops up asks for your Quality Manager credentials, username and password, as input. You authenticate to server by pressing "OK" and you see the first page of wizard.

Figure 2. Login in Quality Manager server



With the second page you can decide whether you want to perform a full import of your Quality Manager test plan (**Full Mode**), or if you want just to import the test plan header (**Quick Mode**).

Figure 3. The mode selection page in the Quality Manager import wizard

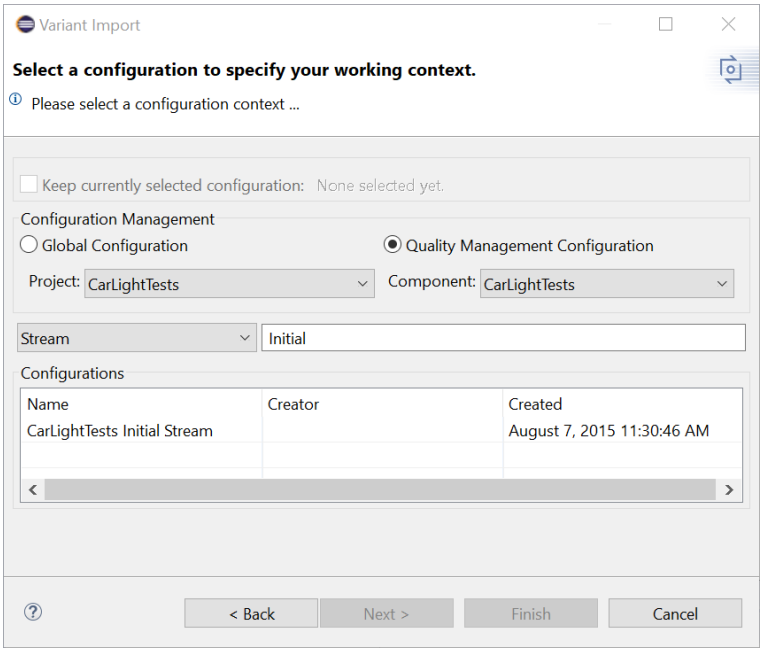


By using **Full Mode** all data of the selected Quality Manager test plan can be imported. Which data will be imported can be configured by the user on the next pages. The imported data is stored and visible in corresponding pure::variants models. The user can use the imported data to preview the variability of Quality Manager test plan within pure::variants. The models need to be explicitly synchronized before transforming a variant.

The Quick Mode is used to just import information pure::variants needs to access the Quality Manager test artefacts during synchronization and transformation. The result are empty models in import. Each module is represented by a pure::variants model containing the link information to the related Quality Manager test plan. In **Quick Mode** there is no need to explicit synchronize the models imported from Quality Manager, this is done automatically during transformation.

The next page of the import dialog shows the selection page for the configuration, the test plan(s) should be imported from. You can import the test plan(s) from a Global Configuration or Quality Manager Configuration. Furthermore you can choose between stream, baseline and other configuration types.

Figure 4. Quality Manager Configuration to import from

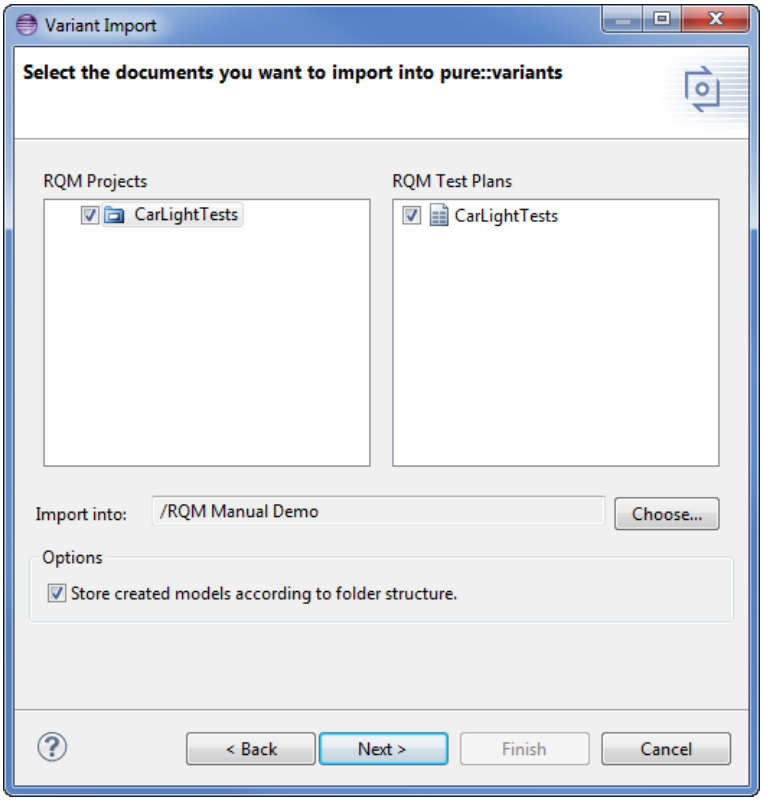


Note

Choosing a component is only required since Jazz 6.0.3

The complete list of projects of the Quality Manager repository is shown. Navigate to the project containing the test plans of interest and select the check boxes on the right side. Selecting a check box on the left side marks all test plans for import. Make sure that the import target location given next to **Import into:** is correct. The location can be changed using the **Choose** button.

Figure 5. The test plan selection page in the Quality Manager import wizard



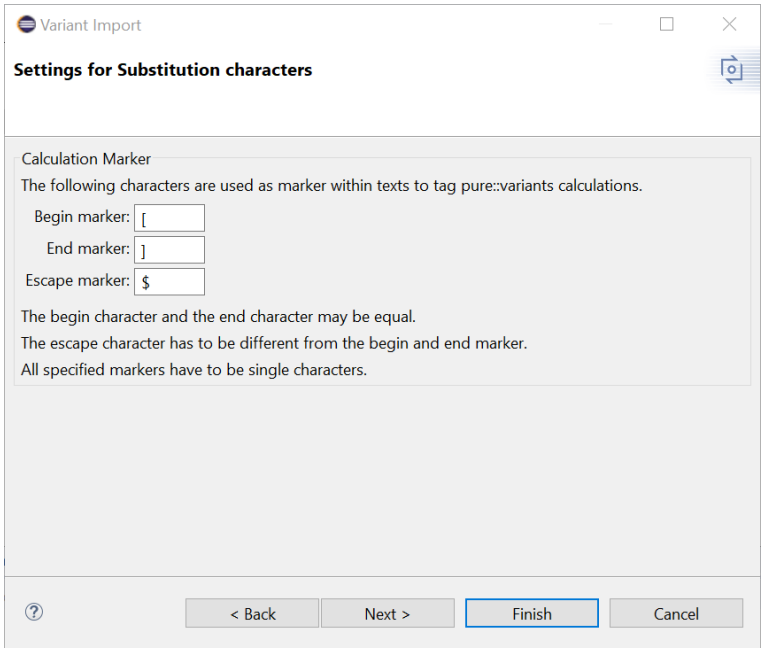
On the next page you can optionally define the custom markers for Partial Text Substitution. Since pure::variant version 5.0.6, the substitution is supported on texts (e.g. test artifact title and description) and rich texts (e.g. test artifact attribute content texts and tables). More on how to use the Substitution markers is described in Section 3.2, “Calculations within test artefacts”

The writing of calculations to test artefacts will work out of the box if it is enabled (set to true) in the transformation module parameter (see Figure 15, “Transformation Configuration”). The standard markers are as follows:

- *Opening character is* ***[***
- *Closing character is* ***]***
- *Escape character is* ***$***

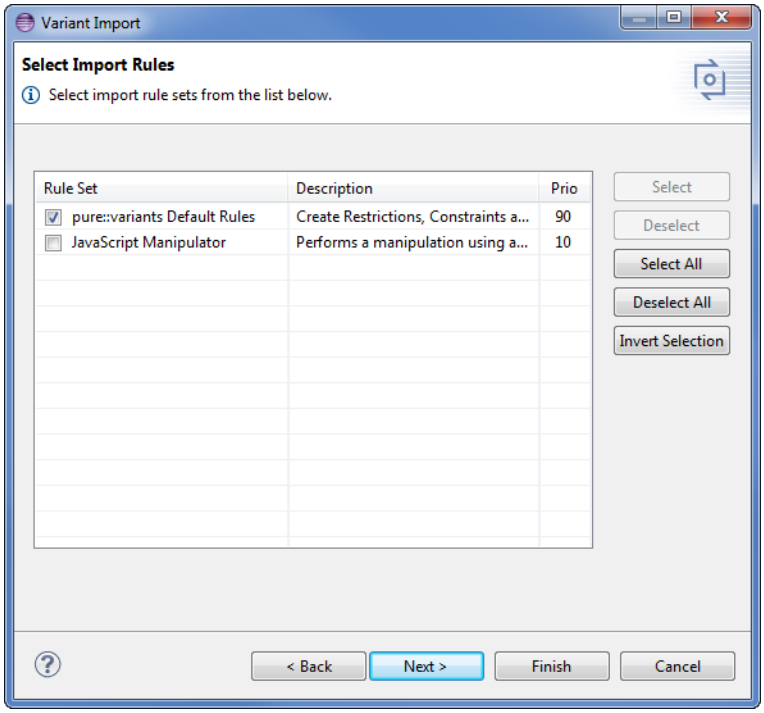
If custom markers are entered, the wizard page will validate and show error message in case of a forbidden character is used. There is also possibility to change these markers later when you synchronize the model. (see Section 2.8, “Updating Models from Quality Manager”)

Figure 6. Quality Manager Partial Text Substitution settings



On the following page the Import Rules are shown. On this page you can select sets of Import Rules, which will be used to manipulate the resulting model after import. Import Rule Sets can be used to create specific pure::variants model elements like restrictions or constraints from Quality Manager test artefacts information.

Figure 7. Select Import Rules to use during import

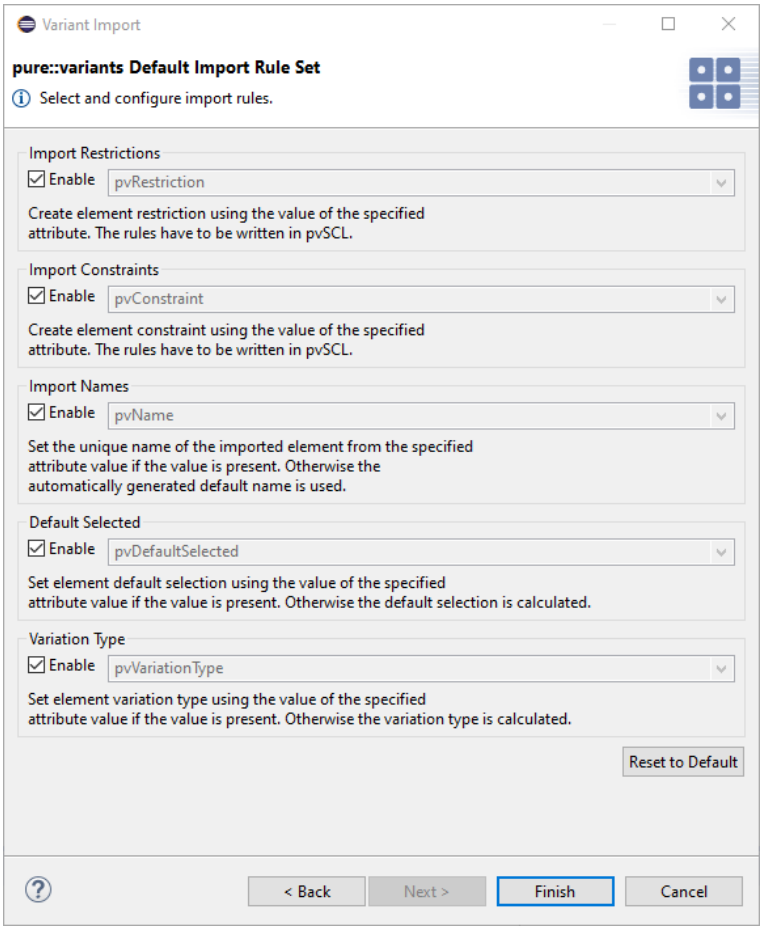


The last pages are showing the settings of the selected Import Rule Sets. For the **pure::variants Default Rules** you can choose which attribute value will be used for creating restrictions and constraints and setting the default selection on elements and which attribute value will be used as unique name for elements. Also you can disable the creation of restrictions, constraints and using specific attribute for unique name, default selection and variation type.

Note

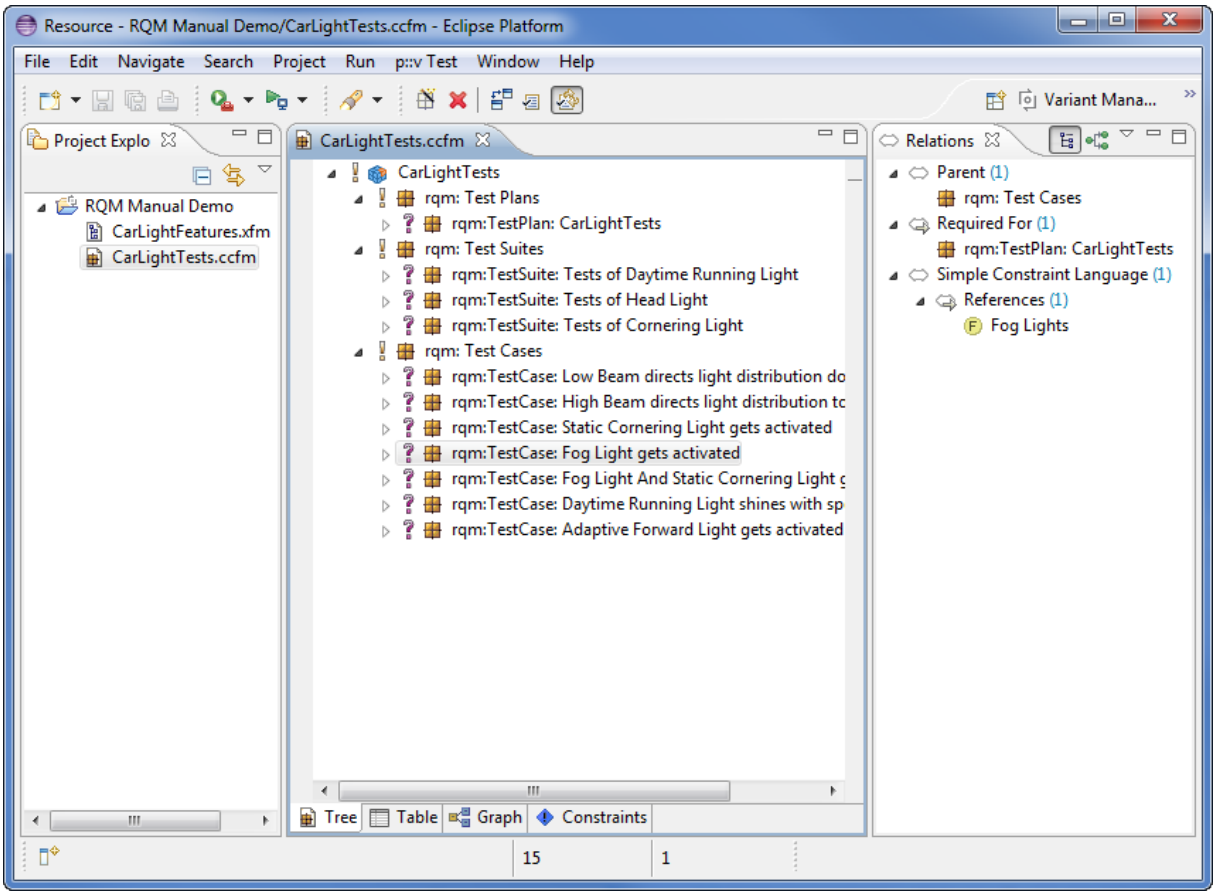
Non-standard custom attribute names are currently not supported. The drop down menus are therefore disabled. Supported names and types are shown in Table 3, “List of custom attribute in Quality Manager to use for pure::variants”

Figure 8. Settings for the pure::variants Default Import Rule Set



The import result will be visible in the Variant Project view. If nothing shows up, use the item **Refresh** in the project's context menu (right mouse click) or press **F5** after selecting the project in the view. Each module is now represented by one pure::variants model. Models can be opened by double-clicking on them or selecting **Open** in the context menu. Figure 9, “Result of initial import of a Quality Manager test plan” shows the typical layout of a Quality Manager test project after import.

Figure 9. Result of initial import of a Quality Manager test plan



The Quality Manager test plan is imported as family model and its test artefacts, like test suites and test cases, are represented as components. The hierarchical structure found in Quality Manager test plans is not reflected identically in the family model. Due to the fact, Quality Manager allows the referencing of test cases to test suites and test plans at the same time, pure::variants Relations are used to represent the relation between test artefacts.

If element attributes are not visible in your model view, you should enable attribute display via the context menu (**Tree Layout** and select **Attributes**).

All test artefact elements are imported as *mandatory* or default selected option (if a restriction was defined in Quality Manager) unless variability information was provided in the Quality Manager test artefacts. This is explained in more detail in Section 3.1, “Adding Variability Information in Quality Manager”.

Table 1. Overview of representation of Quality Manager entities in pure::variants

Quality Manager Entity	pure::variants Representation
Project	family model
Test Plan	component in family model
Test Suite	component in family model
Test Case	component in family model
Test Artefact title	elements visible name (long names are shortened)
Test Artefact attribute `pvRestriction`	element restriction in pvSCL language
Test Artefact attribute `pvConstraint`	element constraint in pvSCL language
Test Artefact attribute `pvVariationType`	element variation type. Valid input values are either `mandatory` , `or` , `optional` , `alterna\`

	`tive` , `ps:mandatory` , `ps:or` , `ps:optional` , or `ps:alternative` .
Test Artefact attribute `pvName`	element unique name. Instead of generating the unique name from Quality Manager object properties, pure::variants will use the defined name as unique name. The name should be a valid unique name (see pure::variants User Guide for more information). If the name is not valid, pure::variants will generate a valid name from it. Uniqueness is not enforced during import, but later shown as model problem in the model editors.
Test Artefact attribute `pvDefaultSelected`	element default selection state. Valid input values are either `off` or `on` . Case is irrelevant, so `OFF` or `Off` are also valid input values.
Test Artefact id	element attribute `id`
Test Artefact resource identifier	element attribute `resourceId`
Test Artefact attribute	element attributes with type ps:string

2.4. Using Variability Information from Quality Manager test artefacts

During the synchronization, pure::variants can optionally use some information present in the Quality Manager test artefacts to create the pure::variants model representation. During synchronization all information including element hierarchy, restrictions and constraints are compared and if different from the information stored in the pure::variants model, shown as mergeable difference.

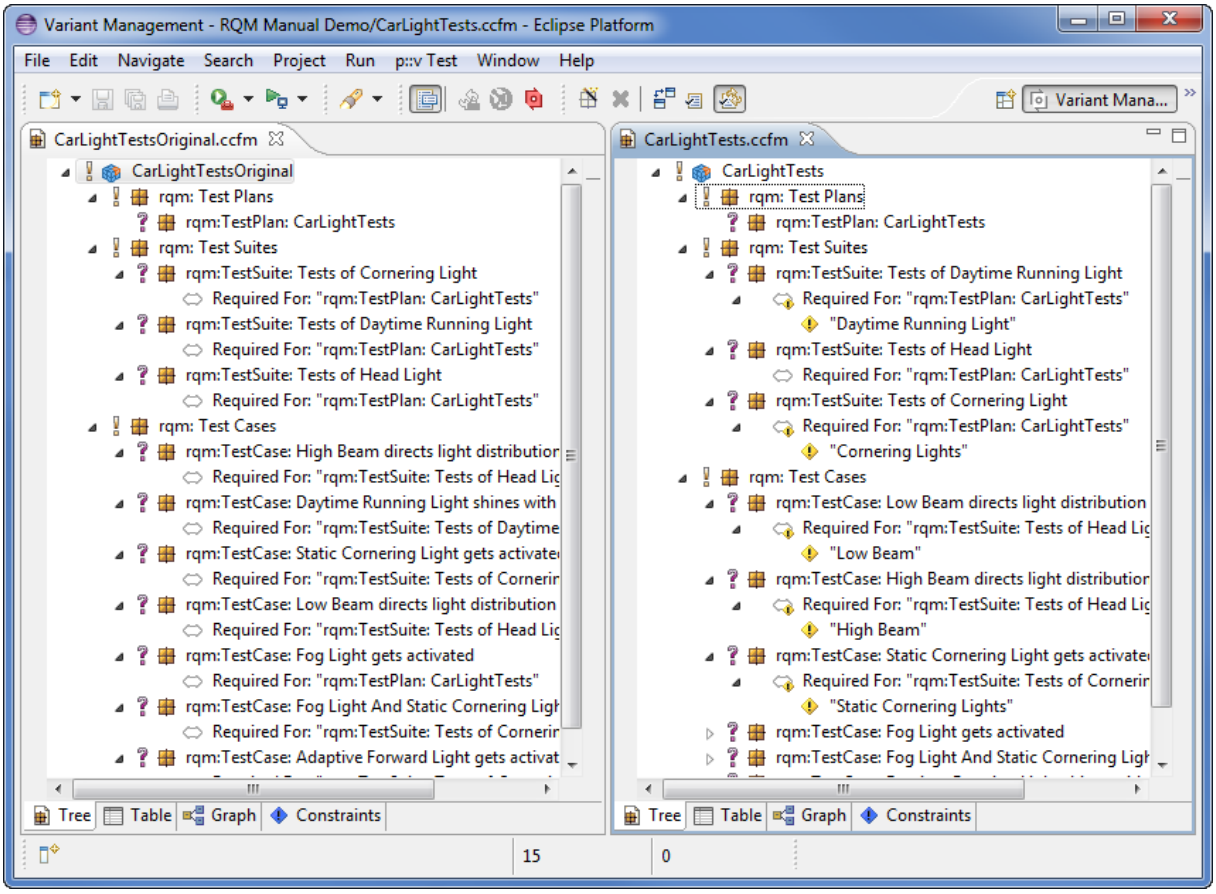
Which and how variability information can be represented in Quality Manager test artefacts is explained in more detail in Section 3.1, “Adding Variability Information in Quality Manager” .

2.5. Adding Variability Information

The pure::variants models created from a Quality Manager test project can be changed to represent the necessary variability information. Every aspect of the pure::variants element representing a Quality Manager test artefact can be changed. Removal or modification of attribute values imported from Quality Manager is also possible. However, these changes will not affect the original Quality Manager test artefacts. During synchronization the user is given the choice to restore the original attribute (value) as read from the Quality Manager test artefacts. Removal of test artefacts is also possible, but again, these changes will be shown during synchronization with Quality Manager. It is usually recommended to keep all the test artefacts and attributes that have been imported from Quality Manager.

Relations like `ps:conflicts` (mutual exclusion) or `ps:requires` may be added as well as restrictions. Restrictions are the most flexible way to express variability rules since they can refer to attribute values and also contain arbitrary calculations. See the pure::variants User Manual for more information on the available relations and the restrictions language.

Figure 10. Original import (left) and modified family model (right)

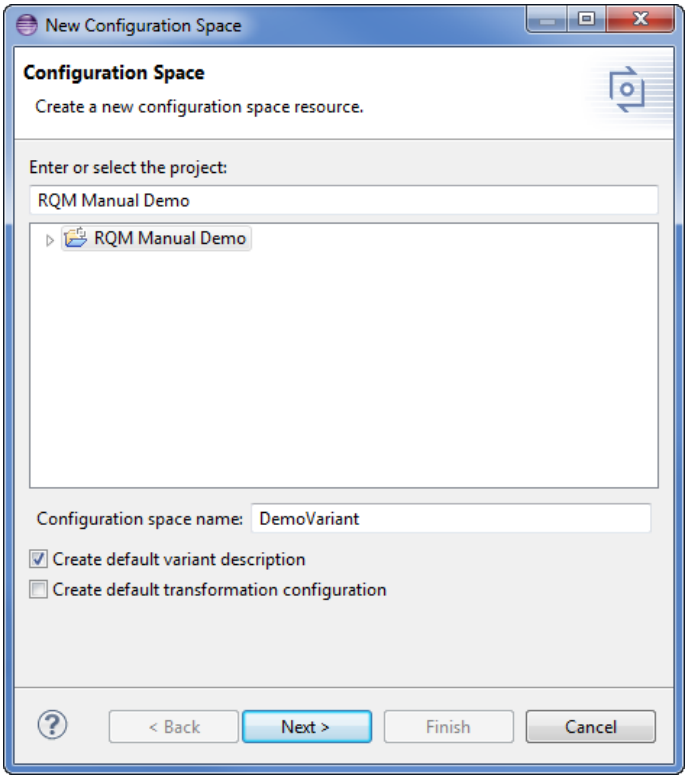


2.6. Defining a Variant

The next step is the definition of the actual variants of interest. Since the variability model usually permits the definition of a very large number of variants, pure::variants keeps track only of those variants which are of interest for the users. Typically this number is much smaller than the number of possible variants.

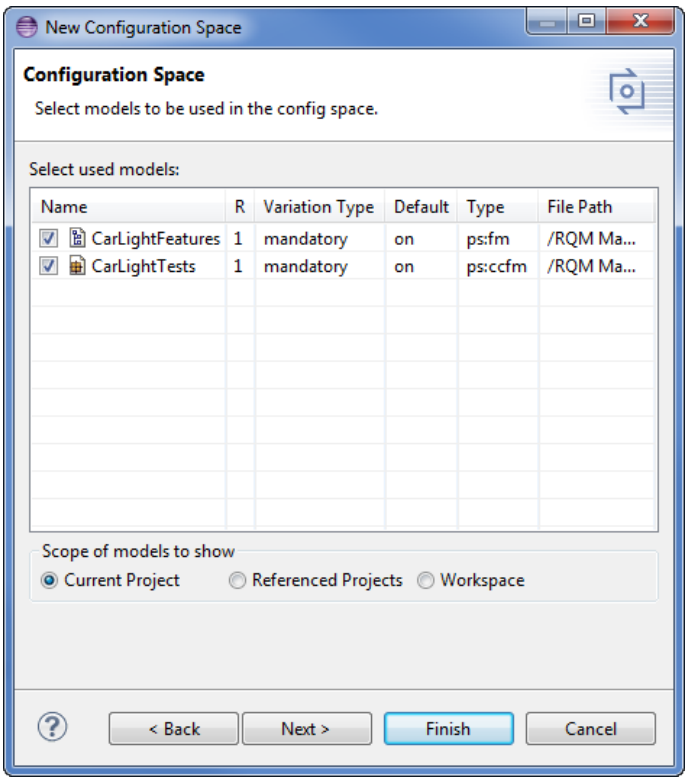
Variants are stored as separate entities called *Variant Description Models* (VDM). A VDM always belongs to a specific *Configuration Space* . Thus before defining variants, a configuration space has to be created. Select the project containing the imported models in the Variant Projects view and open the context menu. Below the item **New** select **Configuration Space** . A wizard is opened. On the first page (Figure 11, “The Configuration Space Wizard, page 1”), enter a name for the configuration space. The name has to follow strict rules (no spaces, no special characters). Uncheck the box before **Create standard transformation** , since for pure test project/plan models the standard transformation does not provide any relevant functionality (See the pure::variants User Manual for more information on transformations).

Figure 11. The Configuration Space Wizard, page 1



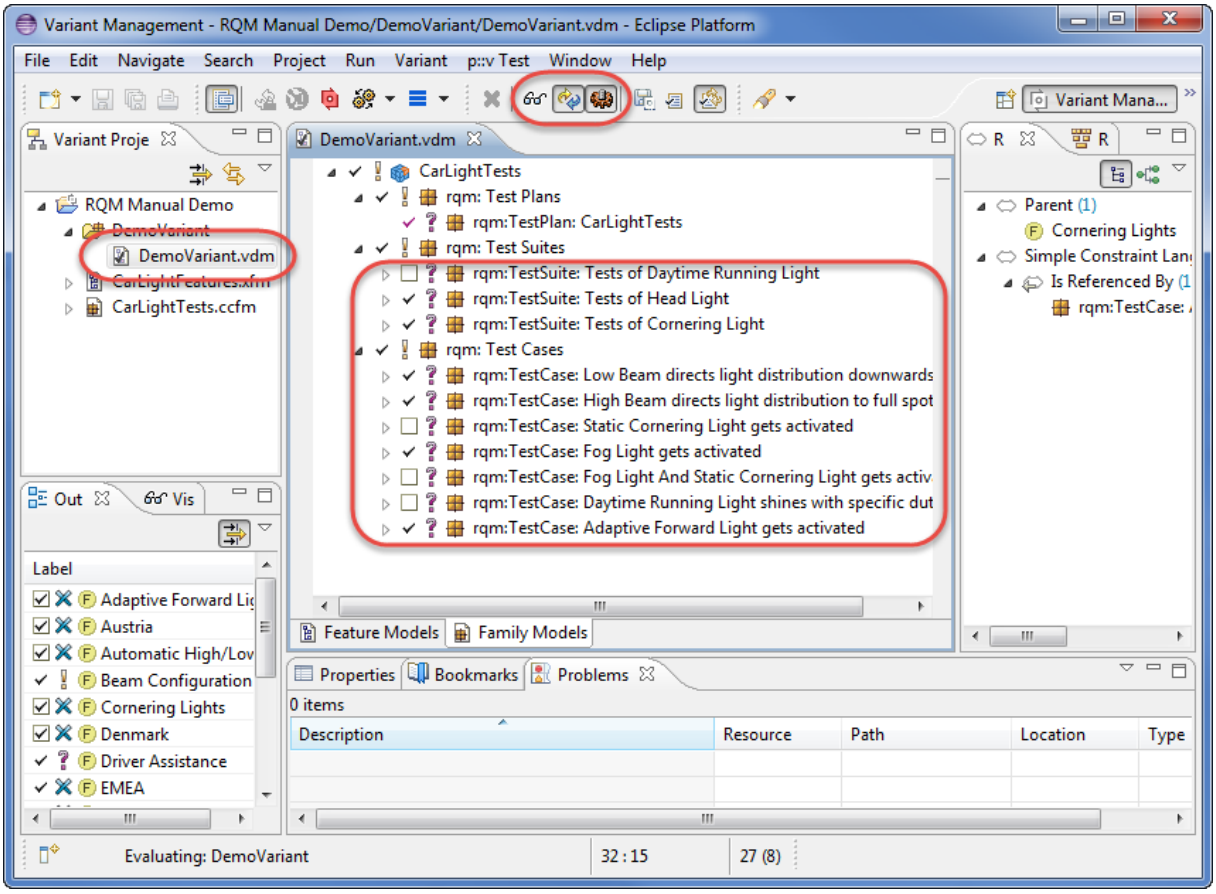
The next page is used to specify which models are to be included in this configuration space. Select here all models that represent the Quality Manager test project/plans of interest. In the example below just one model is selected. Now press the **Finish** button.

Figure 12. The Configuration Space Wizard Model Selection Page



The resulting project structure is shown in Figure 13, “Initial Configuration Space Structure” . The `DemoVariants.vdm` is created and immediately opened. It resembles the structure of the previously defined model(s), but has a checkbox in front of each element to permit the user to select elements for this variant by clicking on it. The buttons marked in the toolbar control the evaluation of configurations. The left-most button ( ) initiates a manual check of the variant configuration. The middle button ( ) toggles between manually checking and automatic checking after changes to the VDM. Finally the right-most button ( ) toggles the auto-resolver on or off. The auto-resolver provides automatic resolution of configuration problems where possible.

Figure 13. Initial Configuration Space Structure



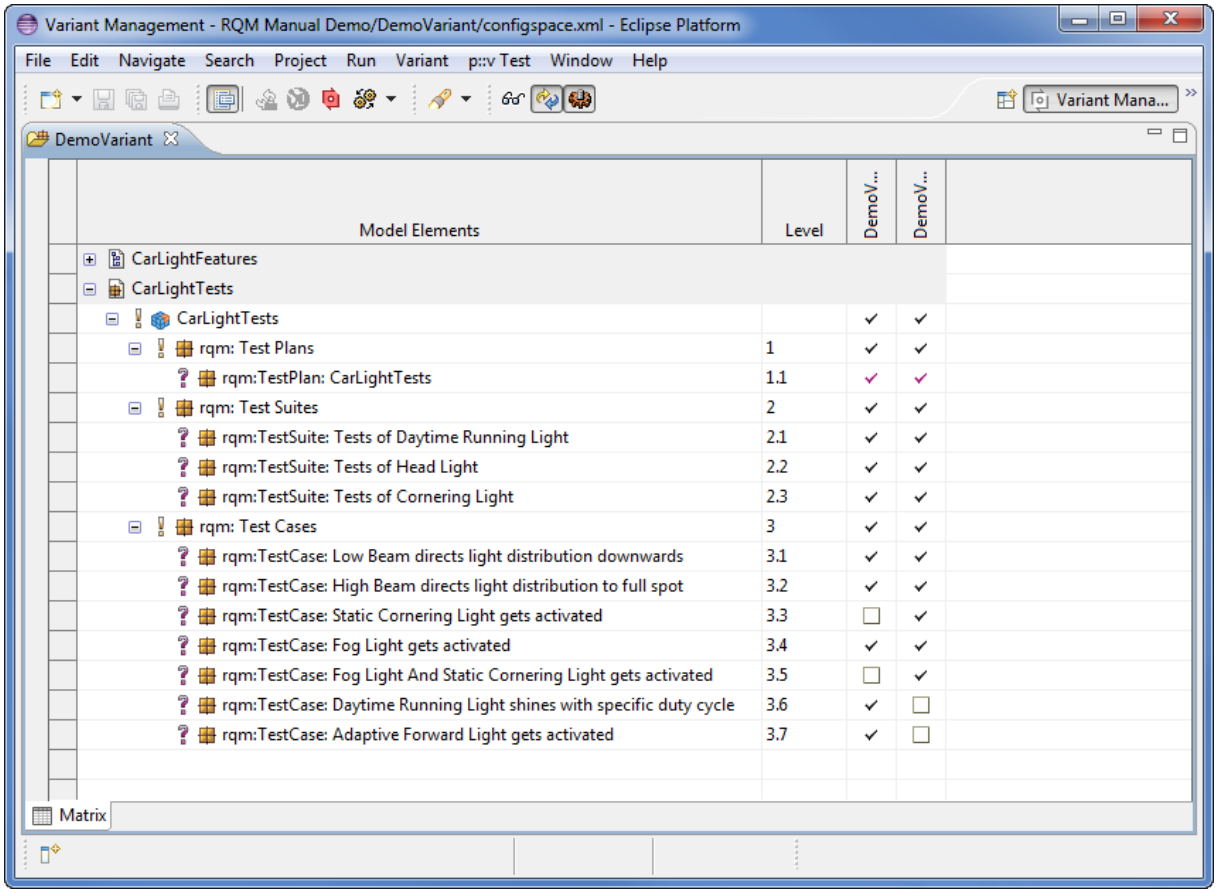
Tip: For small to medium sized models (up to several thousand elements, depending on the speed of the processor), it is convenient to turn on both autochecking and autoresolving by clicking on the respective toolbar buttons.

Problems may be indicated by pure::variants during the selection of test artefact elements. There are several places where problems are shown. Firstly, the Problems view (usually located in the lower right part of the pure::variants perspective) lists all problems such as incompatible elements or a missing selection from *alternative* elements. In addition, problems are shown in the model editor directly in front of the element causing the problem. A tool-tip (which can be seen by moving the mouse over the icon) explains the problem, and the context menu for the problem (right mouse button) provides possible fixes for the problem. E.g. for conflicting elements the fix is to deselect either one or the other element.

Each variant can be represented in its own VDM. To create a new VDM, either select **Clone** from the context menu (in Variant Project view) of an existing variant or use **New->Variant Model** in the context menu of the configuration space. When a valid variant is configured, it can be stored and exported to Quality Manager. The next section explains this in detail.

Variants may be compared at the element level by using the matrix editor (see Figure 14, “Matrix Editor for comparing variants”). This editor is activated by double-clicking on the enclosing configuration space icon. The **Table Layout** item in the context menu can be used to customize the list of variants to compare and the **Show Elements...** and **Filter** items in the same menu can be used to select elements of interest.

Figure 14. Matrix Editor for comparing variants



2.7. Transforming a Variant

Variants stored in a variant description model can be made available in Quality Manager. The Connector currently supports one way of representing variants: *attribute-based*,*stream-based*.

Attribute-Based Variant Representation

In the attribute-based representation we define an custom attribute for each Quality Manager Test artifacts types which are part of the Quality Manager project. This transformation modus adds the name of the variant if the Test artifact is part of the variant. The name of this attribute can be user defined. Default is *pvVariants*. To define custom attributes navigate in the Quality Manager to *Manage project properties* from the Quality Manager project's gear icon, then switch to *Custom Attributes* tab. The custom attribute *pvVariants*/user defined attribute name must be defined for Test Plan, Test Suite, Test Case, Test Script and Manual Test Step artifact types. Currently the Text types such as *Text (Small)*, *Text (Medium)*, *Text (Large)* are supported.

Note

String length for *Text (Small)* is up to 250 characters, *Text (Medium)* is up to 1000 characters, *Text (Large)* is up to 32000 characters.

Stream-Based Variant Representation

In the stream-based representation each variant is represented by a new Quality Manager stream. The new variant stream is derived from the stream/baseline the Quality Manager test project/plan was imported from. The Quality Manager test projects/plans involved in the transformation process, are variant-specifically tailored on this variant stream.

Note

The Test Artefacts are not deleted if they are referenced by other Test Artefacts, which are not part of the configuration space. E.g. If a Test Case is referenced by one Test Plan but also used in another Test Plan, which is not part of the configuration space, it would not be deleted from the project. Hence it is only removed from the Test Plan, which is part of the configuration space.

Note

The transformation moves the deleted Test artifacts to the "Trash" of that variant stream, not permanently delete them. If the Jazz user has "Permanently Delete" permission in the ETM application, they can delete the artifacts permanently from the "Trash" manually. To see the "Trash", from the ETM application page, open the variant stream from the top right corner->Click the gear icon->Trash. Select artifacts types from the drop down menu.

For the support of Global Configurations, the Global Configuration Transformation module must be configured additionally, see *pure::variants OSLC Base Components Manual* of the *pure::variants - OSLC Base* feature extension.

Note

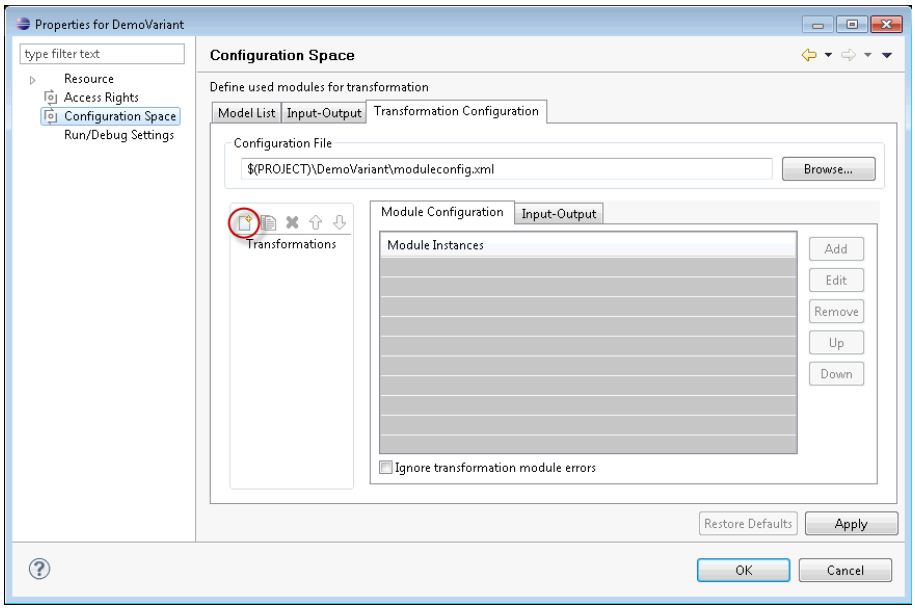
The Jazz user required some specific permissions to import as well as to transform Test artifacts into pure::variants. See Section 3.6, “Minimal permissions required for ETM stream and variant enumeration transformation”

Preparing a Transformation

To transform a variant, first a *Transformation Configuration* has to be created. To create a Transformation Configuration open pull down menu of **Transformation** button in the tool bar ( ) and choose *Open Transformation Config Dialog...*

The configuration space property dialog opens and the *Transformation Configuration* tab is shown. Next step is to add a new *Module Configuration* by clicking the marked tool bar item (see Figure 15, “Transformation Configuration”). Now add a new Module to the Module Configuration, using the **Add** button.

Figure 15. Transformation Configuration



A new Dialog comes up. Choose **IBM Rational Quality Manager** and enter a name. The next page shows some parameters. The **Modus** parameter specifies one of the variant result representations, as described above. One of the following modes can be selected:

- *Variant Enumeration* (see the section called "Attribute-Based Variant Representation") adds the current variant for selected requirements to the enumeration attribute.
- *Create Streams* (see the section called "Stream-Based Variant Representation") derives a new stream - the variant stream - from the initial stream. The initial stream represents your product line and is chosen in Import resp. Synchronization wizard (see Figure 17, "Synchronize model"). In context of the variant stream, the Quality Manager Test Plans involved in the transform process are tailored accordingly to the selection configuration. For the support of Global Configurations, the Global Configuration Transformation module must be configured additionally, see *pure::variants OSLC Support Manual* of the *pure::variants - OSLC Support* feature extension.

Table 2. *Modus*-related transformation module parameters:

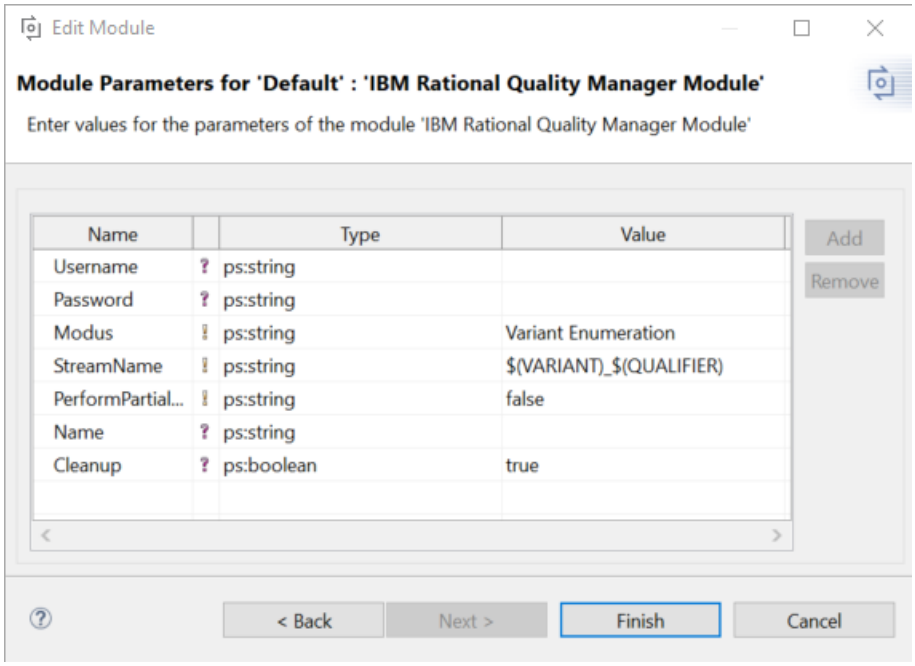
Parameter	Description	Variant Enumeration	Create Streams
Username	The user name for connecting to Quality Manager server.	Optional	Optional
Password	The password for connecting to Quality Manager server.	Optional	Optional
StreamName	Specifies the name for the derived variant stream. The default stream name is a concatenation of the variant name and the date and time the transformation was started.	Unsupported	Required
PerformPartialTextSubstitution	If **true** is selected, the partial text substitution is performed.	Unsupported	Optional
Name	Specifies the name for the enumeration attribute. If not set, the standard name (*pvVariants*) is used. Please note that the attribute with the defined name and type **Text** needs to be existing in your Quality Manager project custom attribute for each type of Test artifacts before the transformation is started.	Optional	Unsupported
Cleanup	If **true** is selected, all existing variant attributes are removed before exporting the current variant.	Optional	Unsupported

For automatic generation of the stream name all standard variant path variables can be used. There are three additional variables

- $(BASELINE)
  - Adds the name of the Baseline, which was used during test plan import or the name of the baseline, which is created as source for the new stream during stream creation.
- $(COMPONENT)
  - Adds the name of the Component the imported test plan is located in.

- $(STREAM)
  - Adds the name of the Stream, which was used during test plan import.

Figure 16. Module Parameter Page

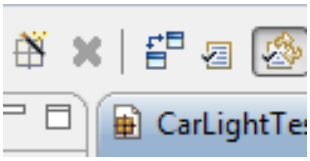


After finishing the dialogs, the transformation can simply be used by clicking on the Transformation button in the tool bar and choosing the transformation configuration in the pull down menu.

2.8. Updating Models from Quality Manager

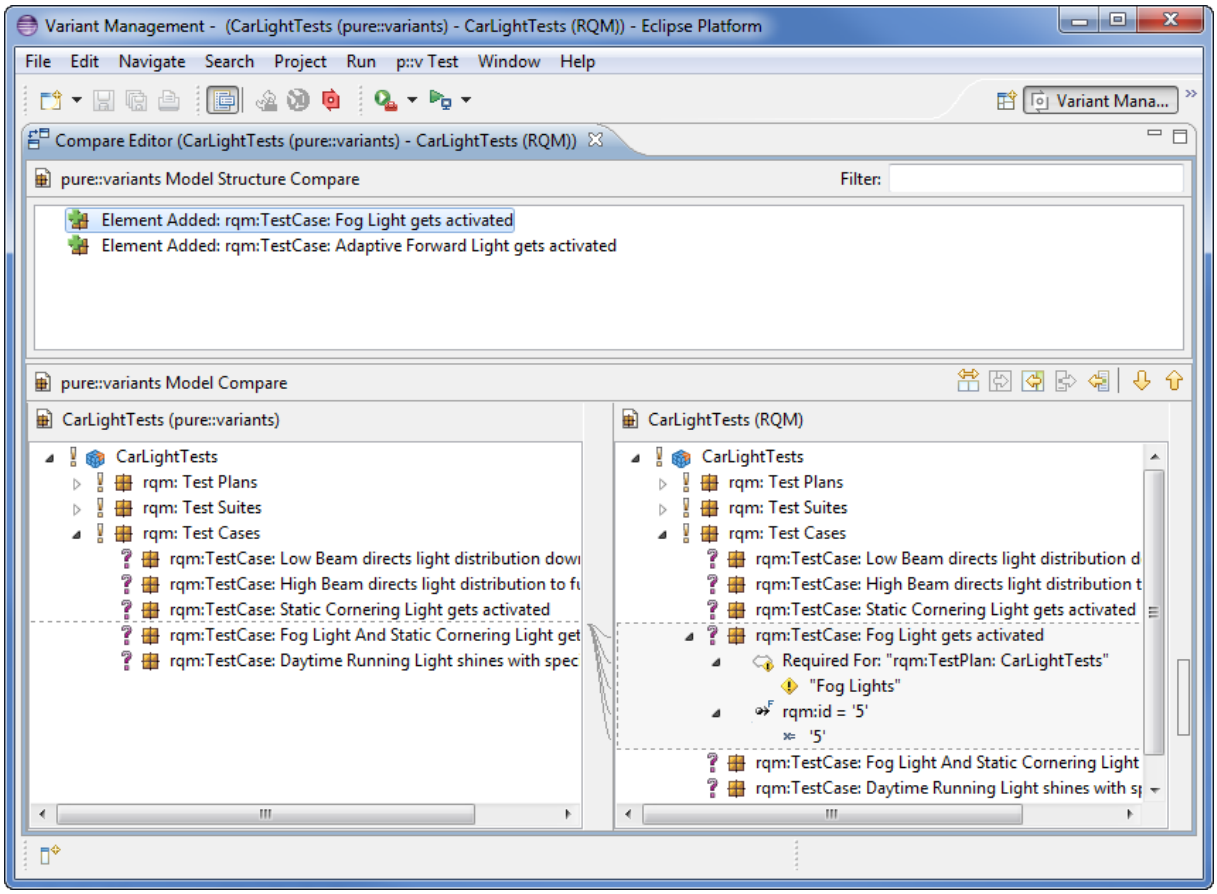
Since there is no live connection between the Quality Manager database and pure::variants, it is necessary to update the pure::variants models with information from Quality Manager whenever relevant changes have been made. To facilitate the synchronization, pure::variants provides a **Synchronize** action. To start the update, open the model representing the Quality Manager test project/plan and press the **Synchronize** button in the tool bar (see Figure 17, “Synchronize model”). pure::variants will connect to Quality Manager and present the so called Compare Editor for pure::variants models (see Figure 18, “Model update from Quality Manager in Compare Editor”).

Figure 17. Synchronize model



The compare editor is used throughout pure::variants to compare model versions but in this case is used to compare the Quality Manager data (displayed in the lower right side) with the current pure::variants model (lower left side). All changes are listed as separate items in the upper part of the editor, ordered by the affected elements. Selecting an item in this list highlights the respective change in both models. In the example, the changed attribute values are marked with boxes and connected with their respective counterparts in the other model.

Figure 18. Model update from Quality Manager in Compare Editor



The Merge toolbar (see marked area between upper and lower editor windows at right) provides tools to copy single or even all (non-conflicting) changes as a whole from the Quality Manager model to the family model.

3. Advanced Topics

3.1. Adding Variability Information in Quality Manager

This subsection introduces the custom attributes of the Quality Manager. By using these attributes, variation information can be added to the Quality Manager model. User has to define these custom attributes in Quality Manager. The type for the custom attributes must be Text. Currently *Text (Small)*, *Text (Medium)*, *Text (Large)* types are supported. The custom attributes can be added for Test Plans, Test Suites, Test Cases, Test Scripts, Keywords (reused Test Scripts) and Test Steps (from Jazz 6.0.2).

Note

String length for *Text (Small)* is up to 250 characters, *Text (Medium)* is up to 1000 characters, *Text (Large)* is up to 32000 characters.

Table 3. List of custom attribute in Quality Manager to use for pure::variants

pure::variants attributes	Standard names to use in Quality Manager	Type
Restriction	`pvRestriction`	`Text (Small), Text (Medium), Text (Large)`
Constraint	`pvConstraint`	`Text (Small), Text (Medium), Text (Large)`

Unique Name	`pvName`	`Text (Small), Text (Medium), Text (Large)`
Default selection state	`pvDefaultSelected`	`Text (Small), Text (Medium), Text (Large)`
Variation Type	`pvVariationType`	`Text (Small), Text (Medium), Text (Large)`

Defining an Element Variability Type

A Quality Manager attribute named `pvVariationType` can be used to provide pure::variants with information about the intended variability type of an object. For each object with this attribute, the attributes value is matched against the four possible variability types for elements (use strings `mandatory` , `alternative` , `or` , `optional` , `ps:mandatory` , `ps:alternative` , `ps:or` , `ps:optional`) during initial import or synchronization. If this attribute is not defined or contains any other value than listed above, the default value of `ps:mandatory` or `ps:optional` (if the element has a restriction defined in attribute `pvRestriction`) is used.

Defining an Element Name

Each imported Quality Manager object gets an automatically generated pure::variants unique name. This name can be used in restrictions, constraints and calculations in pure::variants. If the automatically generated name is not suitable, it is possible to define a different unique name in Quality Manager.

If the Quality Manager attribute `pvName` is defined and not empty, this value is used as unique name. To prevent later problems the name is checked during import and synchronization. In case of violation of the naming conventions pure::variants automatically converts the name to a compatible name. However, pure::variants does not prevent creation of non-unique names during import and synchronization. If duplicate names are existing, they will be marked in the model editor.

Defining Element Restrictions

A Quality Manager attribute named `pvRestriction` can be used to generate a pure::variants restriction for the related pure::variants element. The language used for restriction definition is pvSCL. The pvSCL language is described in detail in the pure::variants User Guide.

Restrictions (as usual in pure::variants) may refer to elements in the same model or in any other model used together with the defining model in a configuration space. For example, a test artefact element should only be selectable, if a feature with unique name "MyFeature" or the feature "MyOtherFeature" is selected, simply use " `MyFeature or MyOtherFeature` " as value for the restriction attribute `pvRestriction` .

Defining Element Constraints

A Quality Manager attribute named `pvConstraint` can be used to generate a pure::variants constraint for the related pure::variants element. The language used for restriction definition is pvSCL. The pvSCL language is described in detail in the pure::variants User Guide.

Constraints (as usual in pure::variants) may refer to elements in the same model or in any other model used together with the defining model in a configuration space. So if the selection of a test artefact should imply the selection of two other elements with the names "MyTestcase" or the feature "MyOtherTestcase", simply use `SELF IMPLIES MyFeature or MyOtherFeature` as value for the restriction attribute `pvConstraint` .

Defining Element Default Selection

Each imported Quality Manager object gets an default selection state.

This is calculated using the following rules:

- If variation-type is mandatory or optional the element gets default selected.
- If the variation-type is undefined and a restriction or constraint is defined, the element gets optional and default selected.

- In all other cases the element is default selected off.

If this does not fit your needs you can specify the default selection for each element by using the Quality Manager attribute `pvDefaultSelected` .

Allowed values are (ignoring case) `on` and `off` .

3.2. Calculations within test artefacts

Titles, description, other attribute with rich text contents and Execution variables of Quality Manager test artifacts may have variable parts, while the most part of the text will remain equal in all of your variants. (Project Execution Variables are not supported). In the below table all supported attributes of the Quality Manager attribute are listed (see Table 4, “List of system attributes per Test Artifact type in Quality Manager to use for pure::variants partial text substitution.”). In QM, these system attributes substitution texts can be added by clicking "*Add content*".

Table 4. List of system attributes per Test Artifact type in Quality Manager to use for pure::variants partial text substitution.

Quality Manager Test Artifacts	Supported attributes for Partial Text Substitution
Test Plan	• Title • Description • Business Objectives • Test Objectives • Risk Assessment • Sampling Strategy • Test Conditions • Test Identification • Test Strategy • Test Bed • Software Test Environment Details • Instruments and Test Equipment • Test Plan Scope • Test Data
Test Suite	• Title • Description • Test Suite Design • Pre-Condition • Post-Condition • Expected Results • Risk Assessment
Test Case	• Title

	• Description • Test Case Design • Risk Assessment • Pre-Condition • Post-Condition • Expected Results • Notes • Requirements • Test Case Scope • Test Description • Test Preparation
Test Script	• Title • Description
Test Step	• Description

In this case you can add a pvSCL statement to be evaluated by pure::variants. The statements will be replaced with actual values of your variant by pure::variants.

A pvSCL statement starts with an opening marker '[' followed by an pure::variants pvSCL calculation rule and is ended with an closing marker ']'. To escape a statement the escape character is used '$'. This will prevent pure::variants from evaluating and replacing the escaped statement.

Example: *[Speed->Max] km/h.* in an Quality Manager test artifact description will be replaced with the value of attribute *"Max"* on Feature *"Speed"* in the exported variant. The result could be: *100 km/h.*

Escaping the rule in the previous example, like *$[Speed->Max] km/h.* , forces pure::variants to ignore the rule. The result, would be *[Speed->Max] km/h.* in that case. The rule is not changed, but the escape character is removed.

Variability in HTML tables

To add variability to HTML tables there needs to be a explicit row and column to hold the variability information. This column and line can be added anywhere in the table, but needs to hold the specified keyword, that is also used to indicate a restriction on e.g. a requirement. By default, this keyword is **pvRestriction.**

Figure 19. Example HTML table

Color	Technology	Static Cornering Light	
White	LED	increased fog light brightness	LED
Yellow	LED	increased fog light brightness	NOT(EU) AND LED
White	Halogen	-	Halogen
Yellow	Halogen	-	NOT(EU) AND Halogen
		CorneringStaticLights	**pvRestriction**

As depicted in the example table, the highlighted pvRestriction cells describe the variability for their respective row and column. The variability information of a specific cell in the table is the AND product of the restriction value of its row and its column. In the example the whole column "Static Cornering Lights" will only be part of the variant, if the feature CorneringStaticLights was selected. The cell below the heading in that column will be included in a variant if CorneringStaticLights AND LED was selected.

The variability information cells (e.g. the marked, yellow pvRestriction cells in the example) can be included in a variant if keepConstraint is set to true (where supported), but must be included for partial transformations, since the information is needed to later create the 100% variants.

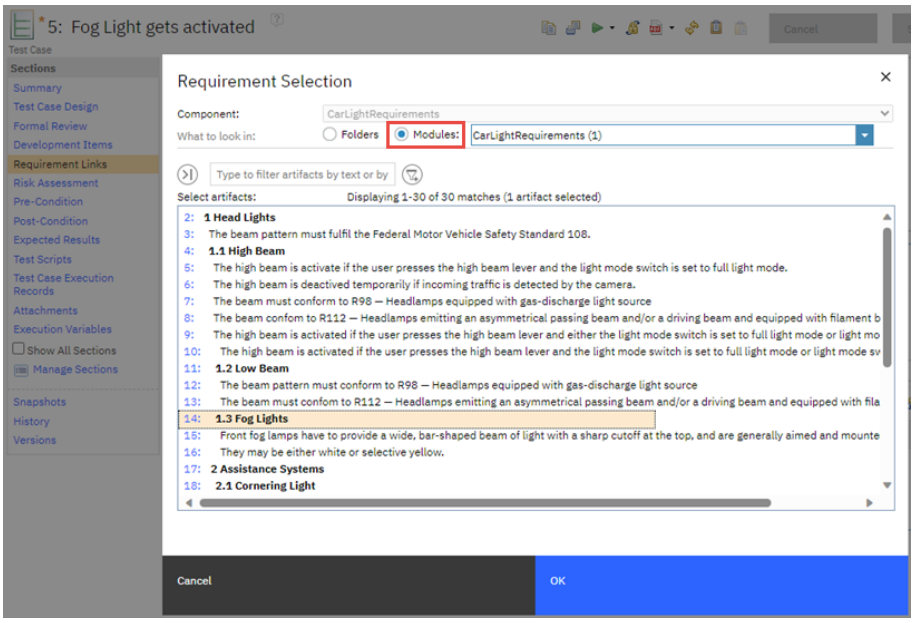
Calculations will also be computed if they are marked with the respective open and close characters, and nested tables, so tables with cells, that themselves again hold a table, are supported and comply to the same rules as described above. But a ** tag is not supported.

3.3. Link Propagation between DOORS Next and Test Managment

The link propagation allows to evaluate the dependency relations between DOORS Next's requirements and Test Management's test cases, with the usage of pure::variants relation types. Therefore, the user can specify for the ***Validate(s) Requirements*** link type an appropriate pure::variants relation type, as equivalent. When importing *test plans* or *requirements modules*, the user should activate the link propagation.

Please be aware, to only create links to requirements, using ***module-scoped links*** (red rectangle), as shown below:

Figure 20. Create link in Test Management



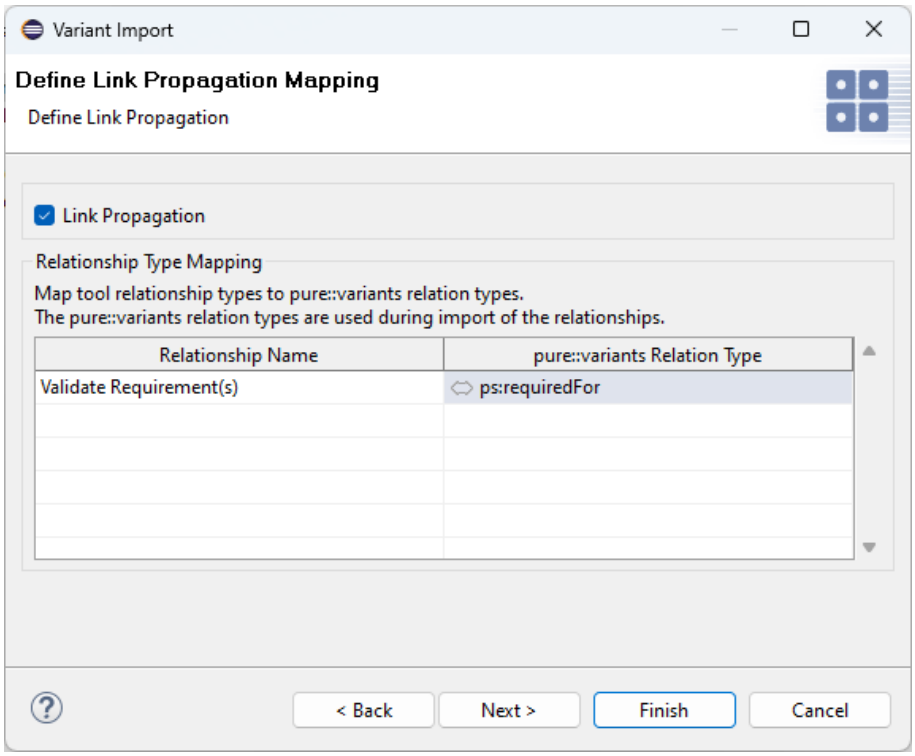
For details of importing a DOORS Next requirements module, see the pure::variants - Connector for IBM DOORS Next manual.

Note

For this link propagation to Test Management, you only need to activate the Link Propagation option while doing the import/synchronization of a DOORS Next requirements module. There is no need to define any link type mapping in the DOORS Next requirements module itself, using the pure::variants DOORS Next integration.

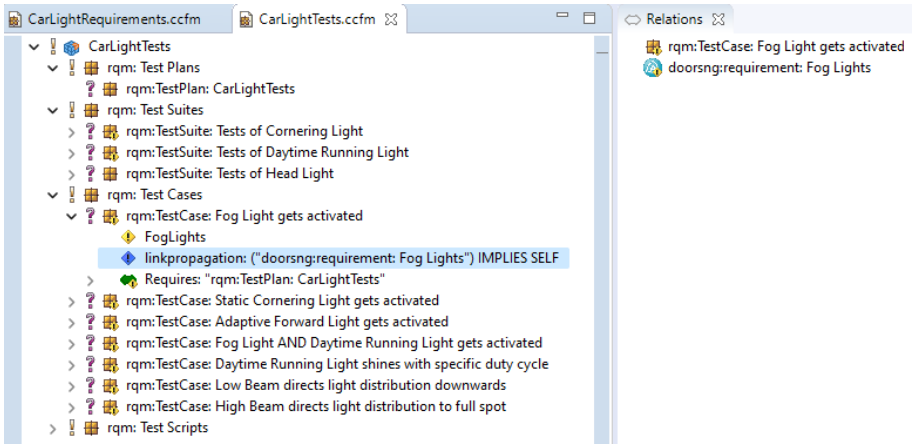
The user must import of relevant test plan(s), with the link propagation enabled and a suitable mapping to pure::variants relations.

Figure 21. Import Wizard - Link Propagation



In the following screenshot, you can see the imported family model, which contains the adapted modelling for proper evaluation:

Figure 22. Family Model - Adapted to link propagation support



For applying the link propagation, the user must import all DOORS Next requirements modules and Test Management's test plans, which are interconnected by links. The family models must be imported from the same ***Global Configuration***, and these must be either present in ***Full Mode*** or ***Quick Mode***, but *not mixed*, in the Configspace.

The Link Propagation is currently only applicable with the ***Stream-based transformation*** mode. The transformation modes are not supported.

3.4. ANT transformation and synchronization

While the transformation and synchronization is supported for Quality Manager test projects/plans a valid authentication is required. There are two possibilities for ANT transformation. Either the user credentials (username and password) are provided in the Quality Manager transformation module or defined as environment variables.

Therefore define *PV_RQM_USER* for username and *PV_RQM_PASSWORD* for password. For ANT synchronization only environment variables are suitable.

3.5. Quality Manager update capability

As already motivated in section *Variant Update* in *pure::variants User's Guide*, the Quality Manager supports the updating of exported variants, i.e., the application of changes done in the product line after the variant was exported without reverting user changes meanwhile done in the exported variant. In case of Quality Manager, the Stream-based transformation approach creates always a new stream for the product variant. This allows the user to transfer his changes from the previously derived variant stream to the newly created one. The functionality for transferring/merging to the new variant stream is provided by Quality Manager (Merge Configuration). Please see the documentation of Quality Manager for further information. In case of transforming a Quality Manager test project, referenced in a Global Stream, the stream is created as described previously. If having included transformation modules, which also support Global Streams, in the same transformation configuration, only one global variant stream is generated. This requires that all variability-aware models, participating in the transformation process, originate from the same global stream.

3.6. Minimal permissions required for ETM stream and variant enumeration transformation

Roles assigned to a user must at least have the following permissions to successfully import and transform Test artifacts. The user permissions can be checked in the ETM application page from a Project Area->Permissions. For more information on this topic, please consult with your Jazz admin or IBM ETM manual.

Table 5. List of mandatory permissions for different transformation modes

		Create Stream	**Variant Enumeration**
Baseline	Create	Required	
Stream	Create	Required	
Save Keyword	Delete	Required	
	Edit	Required	Required
Save Test Case	Delete	Required	
	Edit	Required	Required
Save Test Case Execution Record	Delete	Required	
	Edit	Required	Required
Save Test Plan	Delete	Required	
	Edit	Required	Required
Save Test Script	Delete	Required	
	Edit	Required	Required
Save Test Suite	Delete	Required	
	Edit	Required	Required
Save Test Suite Execution Record	Delete	Required	
	Edit	Required	Required
XML Export		Required	Required
XML Import	Delete	Required	Required
	Post		

4. Troubleshoot

The Quality Manager allows the user to create test artefacts via the REST API e.g. migrating data from other platforms. These artefacts were deleted in the resulted variant stream regardless the selections in the variant model.

The Quality Manager family models are compatible with each other (Synchronization/Transformation) regardless the pure::variants version if none of the artefacts were created via the REST API. With pure::variants version 5.0.4, Quality Manager family models are not backward compatible with the older versions. That means, if a family model is imported/synchronized using pure::variants newer than 5.0.3 and contains REST API created artefacts, it will not be compatible for further synchronization or transformation using an older pure::variants version. In this case, user will see "Test Plan not found" error. Once the model is backward incompatible, it will remains incompatible even the artefact with generated id are deleted from server side.

On the other hand, if an older version family model contains REST API created artefacts, to transform it properly user has to synchronize the family model at least once if using pure::variants version newer than 5.0.3. The transformation process shows a warning if synchronization is needed.

4.1. Connection Issues - Timeouts, Interrupts, etc.

Since pure::variants Connector are heavily relying on integration to IBM's Rhapsody Model Manager tool, as there has to be lot of network communication, which may work out better or worse depending on the company's infrastructure setup. If there are infrastructural problems, like slow network connectivity or slow server deployments, you may overcome these issues, by defining the following parameters:

- Activate retry strategy: *`PV_HTTP_CLIENT_REQUEST_RETRY=true`*
- Increase retry count (default: 3): *`PV_HTTP_CLIENT_REQUEST_RETRY_COUNT=4`*
- Extend connection timeout (in milliseconds, default: 120000): *`PV_HTTP_CLIENT_CONNECTION_TIMEOUT=120000`* (equal to 2 minutes)
- Extend response timeout (in milliseconds, default: 120000): *`PV_HTTP_CLIENT_READ_TIMEOUT=120000`* (equal to 2 minutes)

These parameters can be defined in the *eclipse.ini* file of your pure::variants installation (directly after line **-vmargs**), as follows:

```
...
-vmargs
-DPV_HTTP_CLIENT_REQUEST_RETRY=true
-DPV_HTTP_CLIENT_REQUEST_RETRY_COUNT=4
-DPV_HTTP_CLIENT_CONNECTION_TIMEOUT=240000
-DPV_HTTP_CLIENT_READ_TIMEOUT=240000
...
```

Note

Please ensure to prefix the parameter names with **-D**.