pure::variants - Connector for IBM Engineering Systems Design Rhapsody Manual

Parametric Technology GmbH

Version 7.0.0.685 for pure::variants 7.0

2025

Table of Contents

1. Introduction

1.1. What is pure::variants - Connector for IBM Engineering Systems Design Rhapsody?

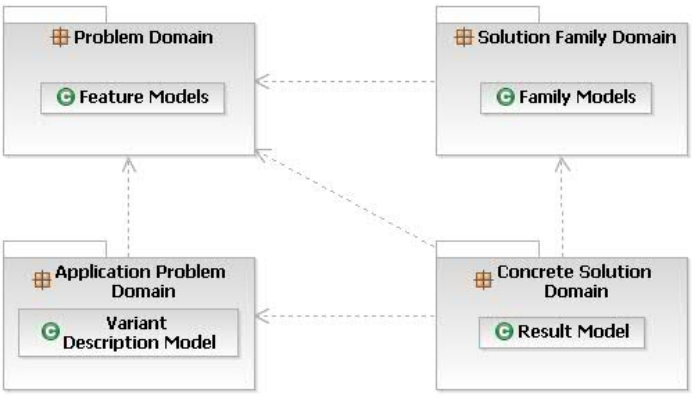
pure::variants - Connector for IBM Engineering Systems Design Rhapsody enables use of product line variability concepts in IBM Rational Rhapsody projects. It allows to maintain one master project from which different project variants are created automatically by selecting features from Feature Models in pure::variants. So instead of having to merge changes in slight variations of the base UML models, the change is applied once to the master project and then all relevant variants are automatically generated by pure::variants.

Figure 1, “Overview of family-based software development with pure::variants” shows the four cornerstone activities of software product line development and the models used in pure::variants as the basis for these activities.

When building the infrastructure for your Product Line, the problem domain is represented using hierarchical Feature Models. The solution domain, i.e. the concrete design and implementation of the software family, is implemented as UML Family Models.

The two model types used for Application Engineering, i.e. the creation of product variants, are complementary to the models described above. The Variant Description Model (VDM), containing the selected feature set and associated values, represents a single problem from the problem domain. The Variant Result Model describes a single concrete solution drawn from the solution family.

Figure 1. Overview of family-based software development with pure::variants



pure::variants manages the knowledge captured in these models and provides tool support for co-operation between the different roles within a family-based software development process:

- The *domain analyst* uses the pure::variants Feature Model editor and UML/SysML models in IBM Rational Rhapsody to build and maintain the problem domain model containing the commonalities and variabilities in the given domain.
- The *domain designer* uses UML models to describe the variable family architecture and to connect it via appropriate rules to the Feature Models.
- The *application analyst* uses a Variant Description Model to explore the problem domain and to express the problems to be solved in terms of selected features and additional configuration information. This information is used to derive a Variant Result Model from the UML model(s) in IBM Rational Rhapsody.
- The *application developer* generates a member of the solution (feature selections and variant specific IBM Rational Rhapsody Models) from the Variant Result Model by using the transformation engine.

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 Systems Design Rhapsody:

IBM Rational Rhapsody:	9.0.0 - 10.0.0 is required. Compatibility with other Rhapsody releases is not guaranteed.
IBM Rational Model Manager:	For the Jazz server, Rhapsody Model Manager (Architecture Management) 6.0.6.1 - 7.0.3 is required. On the client machine, Engineering Workflow Management client (RTC) 6.0.6.1 - 7.0.3 is required. In EWM, the Rhapsody Integration needs to be installed. In Rhapsody, the Rhapsody Model Manager add-on is required. For details, see pv setup guide under section pure::variants - Connector for IBM Engineering Systems Design Rhapsody.

The pure::variants - Connector for IBM Engineering Systems Design Rhapsody is an extension for pure::variants and is available on all supported windows platforms.

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**).

Apart from the connector, also the pure::variants Integration for IBM Rational Rhapsody needs to be installed, and if you plan to transform Rhapsody Model Manager (RMM) projects, RTC still needs to be prepared.

Installing the pure::variants Integration for IBM Rational Rhapsody

Please consult section **pure::variants Integrations** 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 Integrations**).

To use the Integration, it still needs to be added to your Rhapsody project:

1. Open a Rhapsody project
2. Select **File** > **Add Profile to Model...**
3. Select the file "pvRhapsody.sbs" from the Integration installation directory

Now the Integration should be available for the given project. You can open the Integration window at **Tools** > **pure::variants**. If the Integration window does not show, see the section called “What to do when the Integration Window Does not Open?”.

Per default the Integration is loaded when opening the Rhapsody project. If you want to only load the Integration when clicking **Tools** > **pure::variants**, you can edit file `pvRhapsody.prp` in the pure::variants Integration for Rhapsody installation folder. Open the file with a text editor and set property `showonstart` to `False`. After restarting Rhapsody, the Integration window should only open after clicking **Tools** > **pure::variants**.

Preparing pure::variants

You need to specify your rhapsody.ini location in the pure::variants preferences, otherwise the transformation will fail. You can set the rhapsody.ini location in the pure::variants preferences (see **Window** > **Preferences** > **Variant Management** > **Connector Preferences** > **Connector for IBM Rational Rhapsody**).

With older Rhapsody versions, the transformation runs also without setting the rhapsody.ini location - it simply starts up the Rhapsody version that was installed last. Nevertheless, we recommend to set the ini location, since it speeds up the transformation when Rhapsody takes a long time to start, and it allows to choose which Rhapsody installation is used during transformation (in case you have multiple Rhapsody versions installed, see also the section called “Transforming with a specific Rhapsody version”). For automation or continuos integration environments the rhapsody.ini location can also be set with the*PV_RHAPSODY_INI_PATH* environment variable.

You can also optionally specify the RTC eclipse.ini file location and the lscm.bat file location in the pure::variants preferences, otherwise the transformation will look for it at the default location. You can set these in the pure::variants preferences (see **Window** > **Preferences** > **Variant Management** > **Connector Preferences** > **Connector for IBM Rational Rhapsody**). The RTC executable path can also be set with the *PV_RTC_EXEC_PATH* environment variable. For the RTC eclipse.ini path the *PV_RTC_INI_PATH* environment variable is used and for the lscm.bat path the *PV_RTC_LSCM_BAT* environment variable can be used.

When using Rhapsody 9.0 or above, it is possible to run a transformation of RMM projects in offline mode. This means that the transformation is carried out without starting RTC/EWM client. At the above mentioned preference page, you can select the checkbox to enable this feature. Alternatively, you can set an environment or Java system variable that is named **PV_RHAPSODY_RMM_OFFLINE_MODE** to true to set the offline mode.

1.4. About this manual

The reader is expected to have basic knowledge about and experiences with pure::variants. The pure::variants manual is available in online help as well as in printable PDF format here .

2. Using pure::variants - Connector for IBM Engineering Systems Design Rhapsody

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 **Windows Start** 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. How pure::variants - Connector for IBM Engineering Systems Design Rhapsody Works

Before the IBM Rational Rhapsody project is extended with variability information a corresponding feature model project should be set up in pure::variants - Connector for IBM Rational Rhapsody. In this pure::variants project the features to control variability in the IBM Rational Rhapsody projects are maintained. To add variability information to UML models, the concept of UML constraints is used. Special constraints are used to mark up optional elements and connections in an UML model such as classes, states, transition, class attributes and class members. To identify these constraints a stereotype (<<pvRestriction>>) is used. As an addition to using constraints, UML tags or requirements can be used in a similar fashion as constraints.

The constraint language is pvSCL (pure::variants Simple Constraint Language, see pure::variants User's Guide), which provides very simple and intuitive syntax for expressing feature model conditions. For instance, to make a class optional and include it only when the feature `WindSpeed` is not selected in pure::variants, the corresponding pvSCL rule is simply naming the feature inside the not operator: `not(WindSpeed)` .

To create variants of the master Rhapsody project, Variant Description Models (VDMs) have to be created in the pure::variants project. Each VDM contains the feature selection for one project variant. The transformation of a project variant will create a Rhapsody project variant in a specified output location. All variable elements with failing constraint have been removed from this project variant.

2.3. Preparing the IBM Rational Rhapsody Project

Local Rhapsody Project Preparation

To be able to add pure::variants restriction, value calculations, and value choices to elements, the VariantManagement profile has to be added to your project. To do that, follow these steps:

1. Open a Rhapsody project
2. Select **File** -> **Add Profile to Model...**
3. Select the file "VariantManagement.sbs" from the Integration installation directory (see Section 1.3, “Installation” for instructions on how to install the Integration)

Now the profile should be available for the given project.

Rhapsody Design Manager Project Preparation

Externally vs. Actively-Managed Mode

Apart from local Rhapsody projects (.rpy files), the pure::variants - Connector for IBM Rational Rhapsody also allows transformation of Rhapsody Design Manager projects. The Design Manager helps users in working collaboratively on the same Rhapsody project. It supports two different data management modes: *Externally-managed* or *actively-managed* mode.

When models are managed externally, users work offline with the Rhapsody model and need to check in the edited model via the Rational Rhapsody Design Manager Import Engine. When models are managed actively, users work directly with the model located on the Design Manager server.

Both data management modes are supported by the pure::variants - Connector for IBM Rational Rhapsody. To transform actively-managed models, please follow the instructions in the section called “Moving your Rhapsody project file to the Rhapsody Design Manager server” and the section called “Adding IBM Rational Rhapsody Design Manager Server Project”.

Externally-managed models can be transformed just like local Rhapsody models. After transformation, the variant can be imported to the Design Manager using the Rational Rhapsody Design Manager Import Engine. Since

externally-managed models are treated just like local projects until they are checked in, this section about preparing Design Manager projects is only relevant for actively-managed Rhapsody projects.

Moving your Rhapsody project file to the Rhapsody Design Manager server

To move a Rhapsody file project to the Design Manager server it is required to have the Variant Management profile created on the Design Manager server first.

1. Open the Rhapsody project to move to server

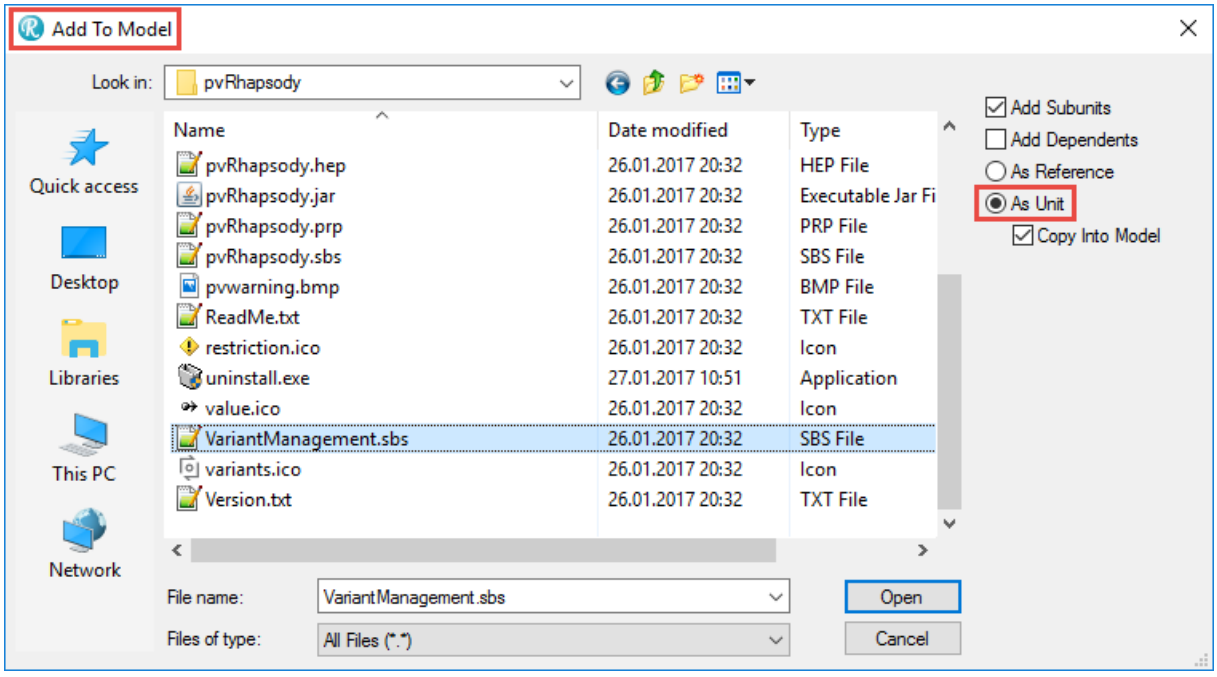
2. Add the VariantManagement profile to your model. Unlike for local Rhapsody projects, you need to call **File -> Add to Model...** and select to add the profile **as Unit**. This ensures that the profile is imported with write access, which is needed for the next step (see below figure).

3. In the Rhapsody project browser, right-click the added VariantManagement profile and select **Design Manager->Create/Update Domain...**, and follow the instructions of the wizard.

4. Now you can move the project via **Tools->Design Manager->Move to Design Manager...**

This has to be done only once per Designer Manager server. For the next project that is located on the same Design Manager server, it suffices to call **File -> Add Profile to Model...** and select the VariantManagement profile from the dropdown box.

Figure 2. Adding the VariantManagement Profile to your Rhapsody project



Enabling the pure::variants Integration in your Actively-Managed Project

To start using the pure::variants Integration in your project, the necessary files first need to be copied to a place that Rhapsody can find on all computers on which the Design Manager project is opened. To this end, follow these steps:

1. Find out the location of your Rhapsody Profiles folder. Depending on how Rhapsody was installed, this can be located at one of the following directories:

   - C:/Users/[username]/IBM/Rational/Rhapsody/[Rhapsody version number]/Share/Profiles

   - C:/ProgramData/IBM/Rational/Rhapsody/[Rhapsody version number]/Share/Profiles

- [Rhapsody installation folder]/Share/Profiles (Only possible if Rhapsody was not installed in folder 'Program Files')

2. Copy the pure::variants Integration for Rhapsody installation folder to the Rhapsody Profiles folder. Common installation directories of the Integration for Rhapsody are:

   - C:\Program Files\Parametric Technology\pv_Enterprise_6.0\com.ps.consul.eclipse.ui.rhapsody.integration or

   - C:\Program Files (x86)\Parametric Technology\com.ps.consul.eclipse.ui.rhapsody.integration

3. Rename the copied folder 'com.ps.consul.eclipse.ui.rhapsody.integration' to 'pvRhapsody'. Otherwise Rhapsody will not be able to find the pure::variants Integration after restarting.

4. Open your Rhapsody Design Manager project.

5. Select **File**->**Add Profile to Model...** and press '**...**'.

6. In the file selection dialog, select the pvRhapsody.sbs that is located in the pvRhapsody folder you copied (see steps 1-3).

7. Now the pure::variants integration window should open.

2.4. Adding Variability to IBM Rational Rhapsody Projects

Variability information in a Rhapsody project can be represented in multiple ways. Either a *pure::variants restriction* can be added to an element, which defines whether the restricted element will be part of the transformed variant, or the value of certain Rhapsody elements can be set through a *pure::variants value calculation* or *pure::variants value choice*.

Adding pure::variants Restrictions

pure::variants restrictions (pvRestrictions) can be represented by UML constraints, UML tags or by tagged or constrained UML requirements. Through each approach, pvRestrictions can be added to UML elements, which are evaluated during transformation. If multiple pvRestrictions are applicable for one element, only one pvRestriction needs to be true in order to keep the element in a variant.

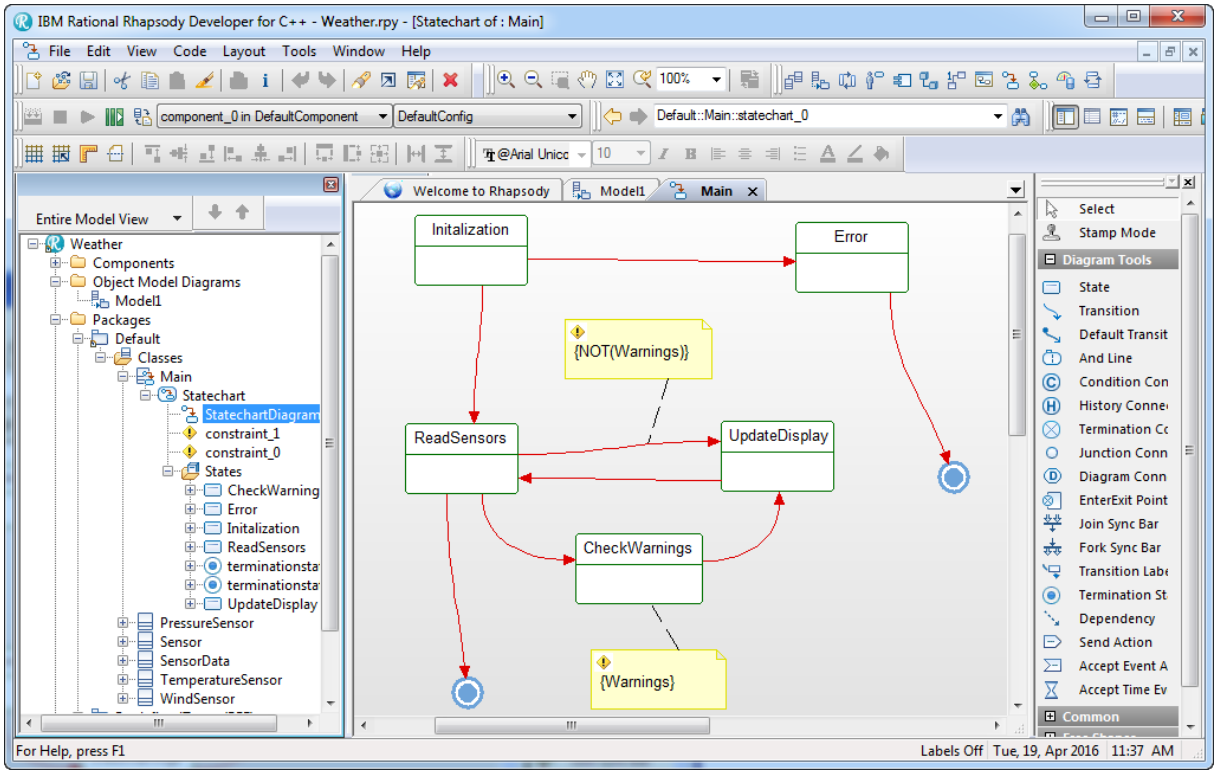
Using Constraints

Representing pvRestrictions through UML constraints is the approach that is supported best. When using this method, either the native IBM Rational Rhapsody context menus can be used to add constraints, or the pure::variants Integration for IBM Rational Rhapsody. Using the Integration has the advantages that auto completion is provided when adding constraints, pvSCL rules are checked for errors, and visualizations help to preview variants or find errors.

To use the native IBM Rational Rhapsody context menus to add a pvRestriction constraint to an element, select the element in the Rhapsody project browser. Use **Add New**->**VariantManagement**->**pvRestriction** from the context menu to add a new constraint. Enter the constraint in pvSCL syntax (see pure::variants User's Guide for details) into the specification. The*pvRestriction* will be applied to the owning element unless it is anchored to other elements. In this case it is applied to all anchored elements. Another way to add constraints manually is by dragging a constraint from the**Common** toolbox and anchoring it to the elements that should be restricted. Then the stereotype of the element has to be set to <<pvRestriction>>.

To use the pure::variants Integration for IBM Rational Rhapsody to add variability to an element, you first need to add the Integration to the IBM Rational Rhapsody project (see Section 1.3, “Installation”) and then use the element's context menu to open a pure::variants constraint editor (see the section called “Adding and Editing Variability Information”). Further information about how to use the Integration can be found in Section 2.5, “Using the pure::variants Integration for IBM Rational Rhapsody ”.

Figure 3. Sample UML Model with pvRestrictions



Using Tags

Another way to add a pvRestriction to a Rhapsody model element is by adding a `pvRestriction` tag to it. This can be done either through the context menu of the relevant element in the model browser or through its**Features** dialog. The created tag has to have the name `pvRestriction` and the desired pvSCL rule as value.

Using Requirements

Optionally, a pvRestriction that is already added to a requirement (e.g., by a constraint or a tag) can be propagated to all elements that are dependent on the requirement. To enable this behavior, the pure::variants Integration for IBM Rational Rhapsody has to be used for editing the Rhapsody master project.

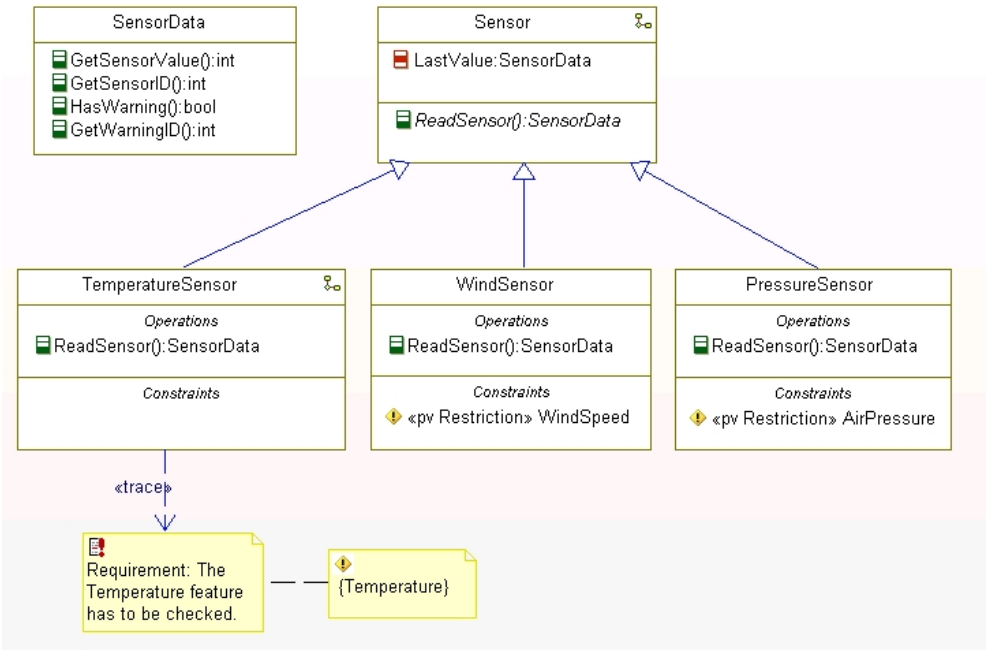
Note

This functionality has to be explicitly turned on in the Integration's preference dialog. You can also adjust propagation settings there. (see the section called “Setting Propagation Preferences”).

Figure 4, “Sample UML Model with requirement used for propagating variability information” shows how the propagation can be used to restrict class *Temperature*. The requirement is enabled/disabled by the constraint connected to it. So if **Temperature** is selected in the current variant the requirement is enabled and all model elements that are connected with dependency links.

The dependency links have to fit the stereotypes selected in the preferences, and the link direction has to be to the requirement.

Figure 4. Sample UML Model with requirement used for propagating variability information



Adding Variability to Rhapsody Element Values

It is possible to add variability directly to the value of some Rhapsody element types, such that the value is replaced in the transformation result. Via *pure::variants value calculations* variability can be added to the values of Rhapsody tags, attributes, or value properties. Or you can use*pure::variants value choices* to add variability to Rhapsody tag values.

Using pure::variants Value Calculations

A pure::variants value calculation is represented by a UML constraint with the stereotype `pvValueCalculation`. During transformation the pvValueCalculation's specification is evaluated and the result is set as the value of its parent element (which can be a tag, an attribute or a value property). The pvValueCalculation's specification has to be pvSCL-conform. Therefore, it is highly recommended to use the pure::variants Integration to add new pvValueCalculations.

When the pure::variants Integration is enabled (see Section 1.3, “Installation”), you can add a pvValueCalculation through the context menu **Add New p::v Value Calculation** or edit an existing pvValueCalculation through its context menu **Edit p::v Value Calculation**. **Add New p::v Value Calculation** is supported on all tags, attributes or value properties. See the section called “Adding and Editing Variability Information” for instructions on how to use the pvSCL editor.

Using pure::variants Value Choices

For example, when the tag for which the value should be set is of a type that cannot be represented by a pvValue-Calculation (e.g., of type 'Class'), you can use a pure::variants value choice instead.

A pure::variants value choice is represented by a UML constraint with the stereotype `pvValueChoice`. Below the pvValueChoice constraint, multiple UML tags can be defined and restricted with pure::variants restrictions. During transformation first the restrictions on the child tags are evaluated, so that only those child tags remain, which have restrictions that evaluate to true. Then the value of the parent tag is set to the value of the first remaining child tag. Please note that pvValueChoices are only supported for tags of multiplicity "1" and that the pvValueChoice's child tag needs to be of the same type and multiplicity as the pvValueChoice's parent tag. Furthermore, the pvValueChoice's child and parent tag must always have a value set.

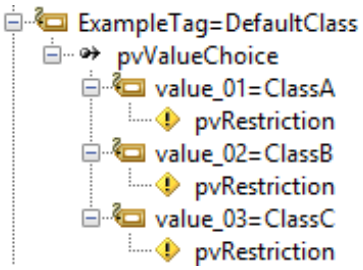
Note

The order that is used to decide which of the pvValueChoice's child tags is the first is always the order that is shown in Rhapsody. If the option **View->Browser Display Options->Enable Ordering** is enabled in your Rhapsody project, the user-defined order is used. Otherwise, child tags are processed in lexicographic order.

Example: In Figure 5, “Tag of Type 'Class' annotated with a pure::variants Value Choice”, the parent tag 'ExampleTag' has a pvValueChoice constraint with three child tags 'value_01' to 'value_03'. Each child tag is restricted. Let's assume that during transformation the restriction of 'value_01' evaluates to false, while the other value's restriction evaluates to true. In that case the value of 'ExampleTag' will be set to 'ClassB', since 'value_02' is the first remaining child of pvValueChoice. Furthermore, the pvValueChoice will be removed from the variant.

When instead all three child tags ('value_01' to 'value_03') evaluate to false, the value of 'ExampleTag' will stay 'DefaultClass', since no child tag of the pvValueChoice remains and the tag's original value is used as default value.

Figure 5. Tag of Type 'Class' annotated with a pure::variants Value Choice



To create new pvValueChoices and add new restricted child tags it is highly recommended to use the pure::variants Integration. When the pure::variants Integration is enabled, you can add a pvValueCalculation through the tag's context menu **Add New p::v Value Choice**, and add a new restricted value through the pvValueChoice's context menu **Add New p::v Value with Constraint**. **Add New p::v Value with Constraint** will first copy the pvValueChoice's parent tag as a child of the pvValueChoice element and open the pvSCL editor for adding a new pvRestriction constraint to the copied tag. See the section called “Adding and Editing Variability Information” for instructions on how to use the pvSCL editor.

2.5. Using the pure::variants Integration for IBM Rational Rhapsody

To support users of the pure::variants - Connector for IBM Engineering Systems Design Rhapsody, the Integration provides visualizations (see the section called “Visualizing Variability Information”) and an editor for variability information (see the section called “Adding and Editing Variability Information”). The instructions for installing the Integration and showing it in your Rhapsody project can be found in Section 1.3, “Installation”.

First Use

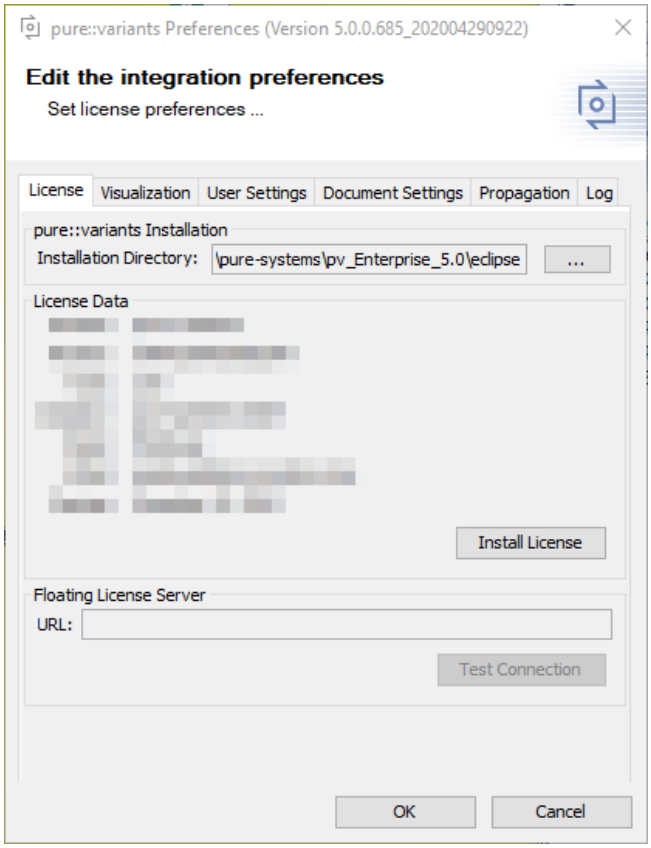
When you first use the Integration after installation, it is necessary to check whether the license preferences are correct. To this end, open the Integration preferences dialog via the button in the Integration window.

A dialog opens that shows the path to your pure::variants installation and your license information (see Figure 6, “Preferences Dialog”). If any of the information is missing, you need to enter it. Use the **...** button in the **pure::variants Installation** group to enter the installation directory, and the **Install License** button to specify your license.

If you are using a floating license and the URL in the **Floating License Server** group is not set already, you need to enter the URL. To test if the connection to the floating license server is established, press the button **Test Connection**.

Now you can use the Integration.

Figure 6. Preferences Dialog



Connecting with pure::variants Models

For editing variability information and viewing visualizations, it is necessary to connect your Rhapsody project with one or more pure::variants models. The following types of pure::variants models can be loaded:

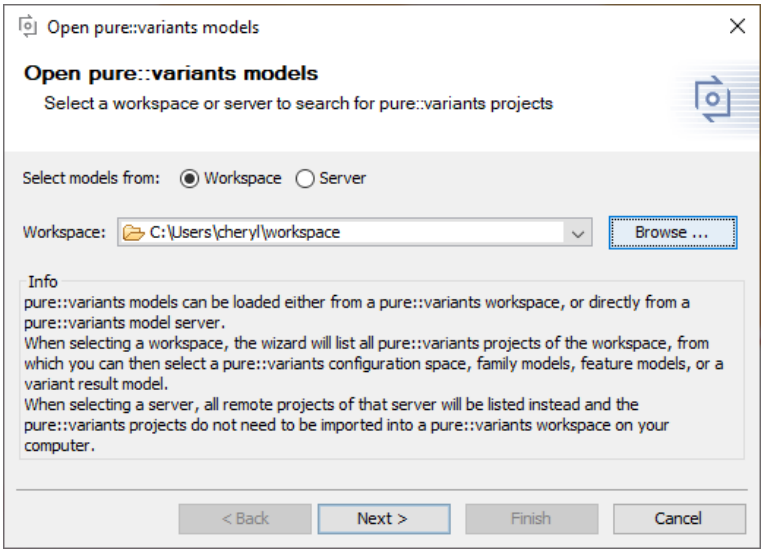
- Recommended: pure::variants configuration spaces, which enable selection of contained variant description models (.vdm)
- pure::variants variant result models (.vrm)
- pure::variants feature models (.xfm)
- pure::variants family models (.ccfm)

pure::variants models can be opened from two different sources: Either from a *pure::variants/Eclipse workspace* or from a *pure::variants model server*.

Opening models from a workspace

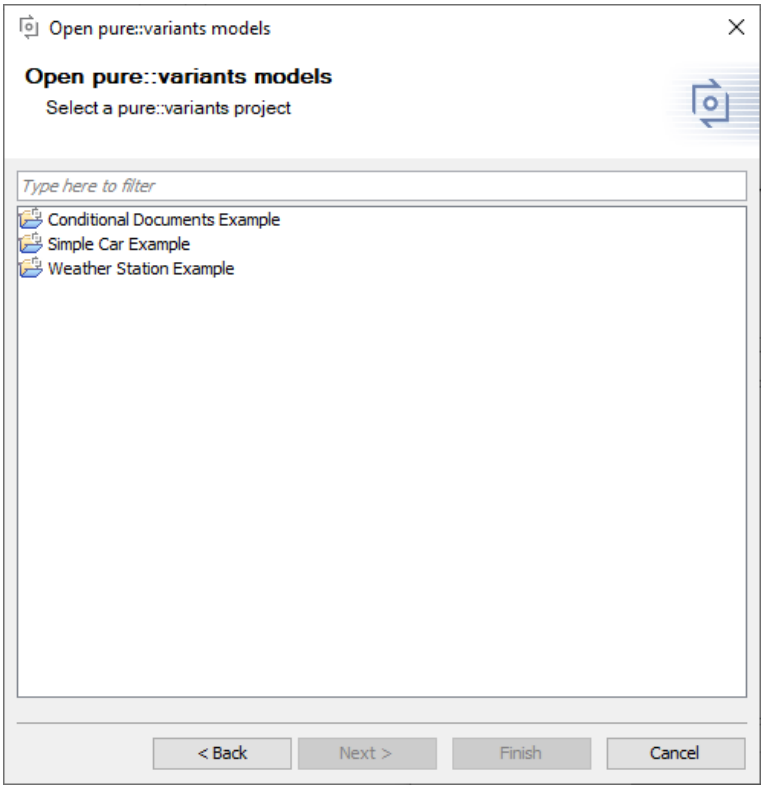
To open a model or configuration space, press on the Integration window. This will open a wizard, which first allows choosing the source (workspace or server). Choose *workspace* and then browse to find your pure::variants workspace folder. Already known workspaces are listed in the workspace dropdown box. If you later need to add or remove a workspace from the list, you can go to tab *User Settings* of the Integration preferences (accessible via).

Figure 7. Mode selection page



On the next page, all projects are listed that are located in the selected workspace folder or that are linked into the pure::variants/Eclipse workspace.

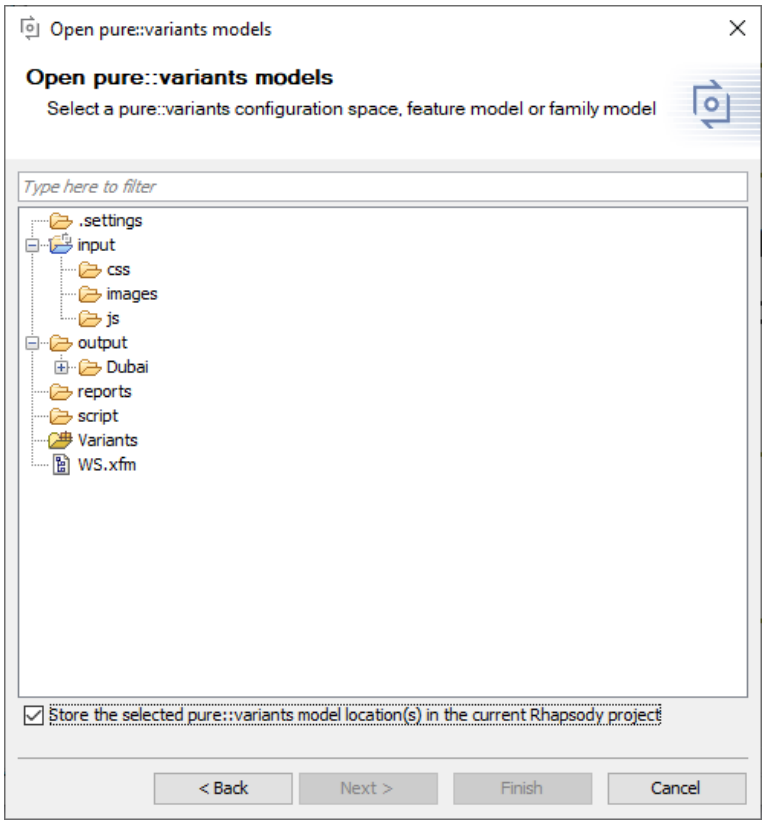
Figure 8. Project selection page



Select one and on the next page choose the model(s) or configuration space you want to open.

Checkbox *Store the selected pure::variants model location(s) in the current Rhapsody project* allows you to save the selected model locations in the current Rhapsody project, so that these models will be opened again once any user opens this Rhapsody project. If you do not select the checkbox, the model locations will only be stored on your computer. For details, see the section called “Saving and Loading pure::variants Models”.

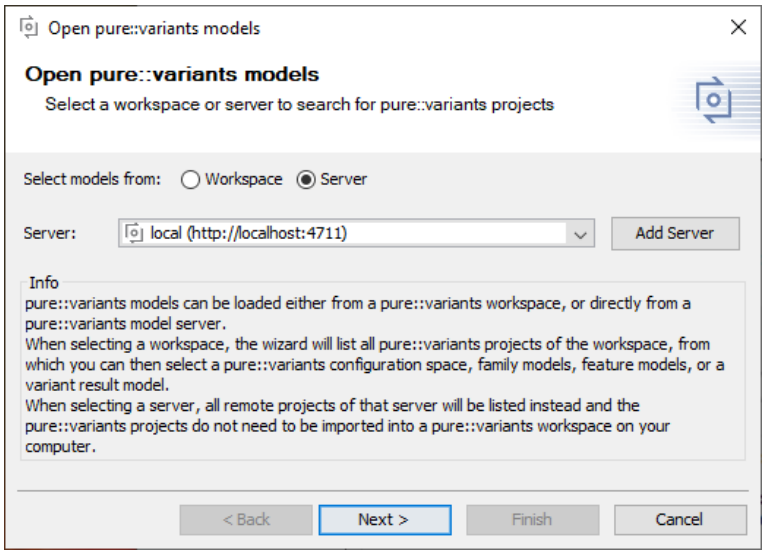
Figure 9. Model selection page



Opening models from a model server

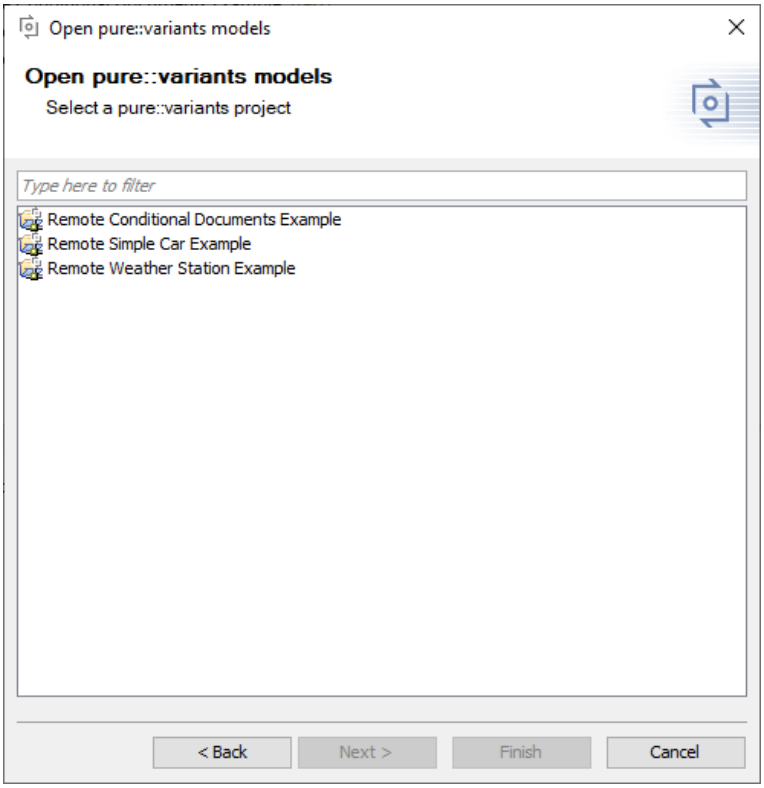
To open a model or configuration space directly from a pure::variants model server, also press . On the first page of the wizard, choose *server* and add the server address via button **Add Server**. Like in pure::variants, new servers need a name and the server address (e.g., https://yourserveraddress:443). Any known servers are listed in the server dropdown box. If you later need to add or remove a server from the list, open the Integration preferences by pressing . On tab *User Settings*, you can add or remove servers.

Figure 10. Mode selection page



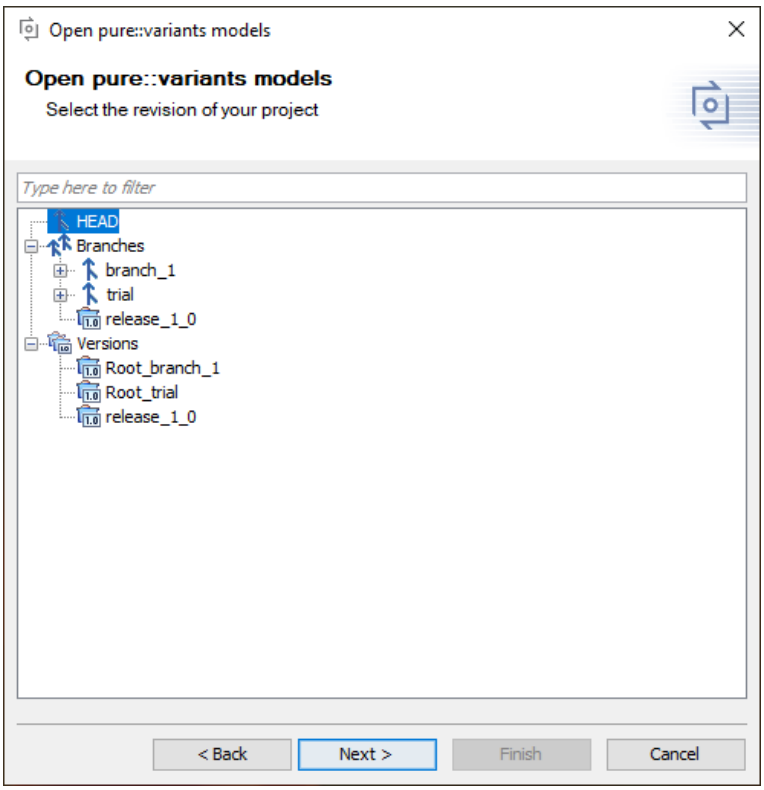
On the next page of the wizard, all projects of the server that the current user has read access to are listed.

Figure 11. Project selection page



Select one and on the next page choose the project's revision (branch or tag) from which you want to load a model.

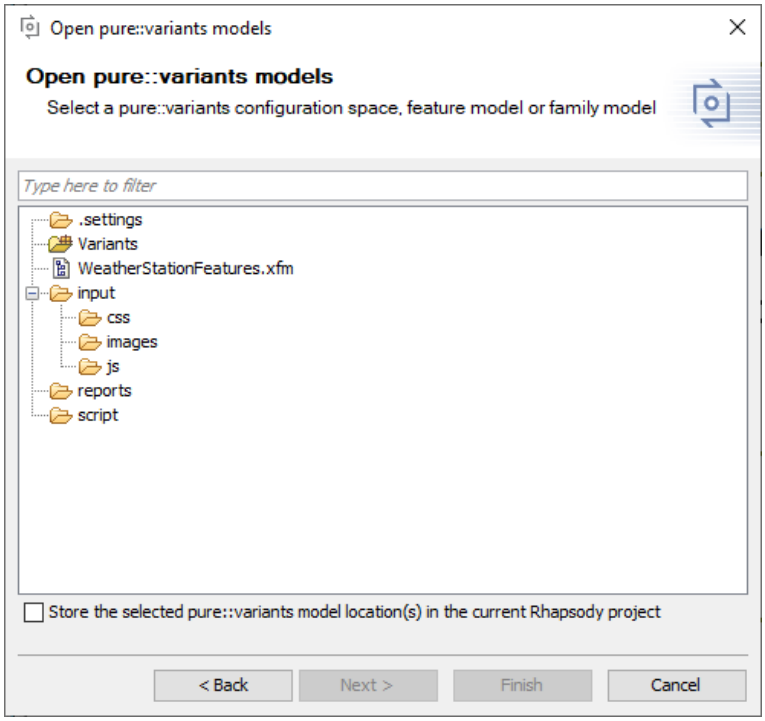
Figure 12. Project selection page



Finally, on the last page select the model(s) or configuration space you want to open.

Checkbox *Store the selected pure::variants model location(s) in the current Rhapsody project* allows you to save the selected model locations in the current Rhapsody project, so that these models will be opened again once any user opens this Rhapsody project. If you do not select the checkbox, the model locations will only be stored on your computer. For details, see the section called “Saving and Loading pure::variants Models”.

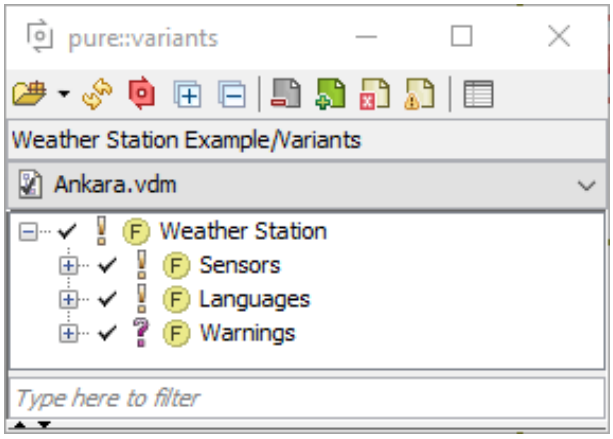
Figure 13. Model selection page



Opening pure::variants Configuration Spaces

To open a pure::variants configuration space, use the wizard as described above. On the last page, select a configuration space folder. Now the Integration window should show all used models of your configuration space. Please note that family models (.ccfm) are not opened per default. You can enable loading family models in the Integration preferences on the **Visualization** tab. After selecting a variant from the dropdown list, selections should be shown in front of features. To ease usage of configuration spaces with many variants, the latest opened variants are shown at the top of the list.

Figure 14. Configuration Space with Selected Variant



[1]You can create a variant result model in pure::variants by clicking the **Save Result to File** button that is shown in the toolbar of a variant description model.

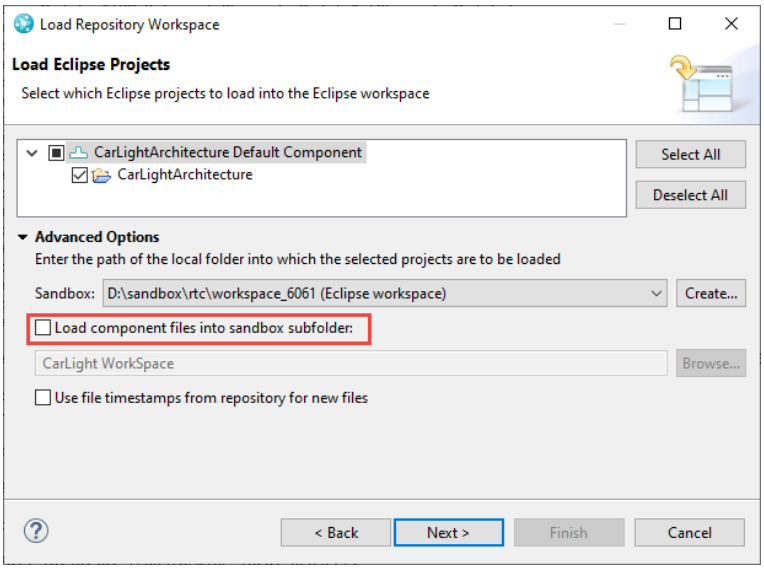
Opening Other pure::variants Models

Other pure::variants models, such as variant result models[1] (.vrm), feature models (.xfm), and family models (.ccfm) can also be opened via . Please note that family models (.ccfm) are not listed per default. You can enable loading family models in the Integration preferences on the **Visualization** tab.

Opening pure::variants Models Checked Out via Rational Team Concert

Please note that pure::variants models checked out in an RTC repository will only work if the component files are checked out directly in the workspace. To do that, "Load component files into sandbox subfolder" needs to be deselected when loading the repository workspace.

Figure 15. Load Repository Workspace



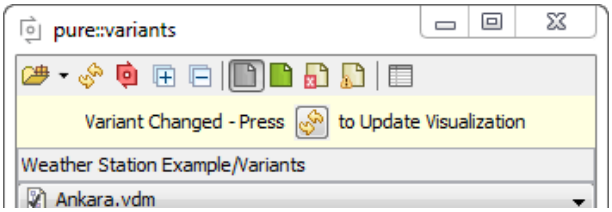
Live Connection with pure::variants

Since pure::variants 4.x, changes of the loaded pure::variants models are propagated live to the Integration. For example, directly after editing the name or changing the selection of a feature the loaded models are updated in the Integration window. To enable this live update, the following prerequisites need to be fulfilled:

- the opened model needs to be located in an Eclipse workspace
- the changes have to be done on the same Eclipse workspace using pure::variants 4.x or later
- either a configuration space, feature model or family model needs to be loaded (Variant result models can only be updated automatically when the .vrm file is saved)

If a visualization is active when a loaded model has changed, a pane is shown that informs you about a pending visualization update (see Figure 16, “Information about Pending Visualization Update”). When pressing the pane's refresh button, the visualization is updated.

Figure 16. Information about Pending Visualization Update



When the used models of a configuration space have changed or a new variant model was added to the configuration space, a live update of the currently loaded models is not possible. In this case, you can press to manually reload all pure::variants models and refresh the current visualization. To unload all models and free the pure::variants license, press .

Saving and Loading pure::variants Models

To ease the work with pure::variants Integrations, the last loaded model locations are saved, so that the model will be opened again automatically, next time you start the tool. Per default, these model locations are saved only for you on your local machine. If you want to save the last loaded model locations for all users, who are using a certain Rhapsody project, you can select checkbox *Store the selected pure::variants model location(s) in the current Rhapsody project* on the last page of the open model wizard. Then everytime a user opens that Rhapsody project, the pure::variants models stored in the document will be opened instead of the locally stored models (if they can be found on the user's machine).

Furthermore, a list of the latest loaded models can be accessed via the small arrow next to the button.

Please note that models are saved relative to your current workspace, which is the workspace where your current model is located in or linked to. Therefore, you may be asked for your current workspace location when loading a model from a different workspace, or a model that is not located in a workspace (but may be linked into a workspace). Hint: If you want to know where exactly the loaded model is located, you can hover over the name of the model. A tooltip will show the full path of your currently loaded configuration space or model.

If you do not want to load a model again on startup, you can clear the stored model locations from the user and/or document settings. For details, see the section called “Manage Settings”.

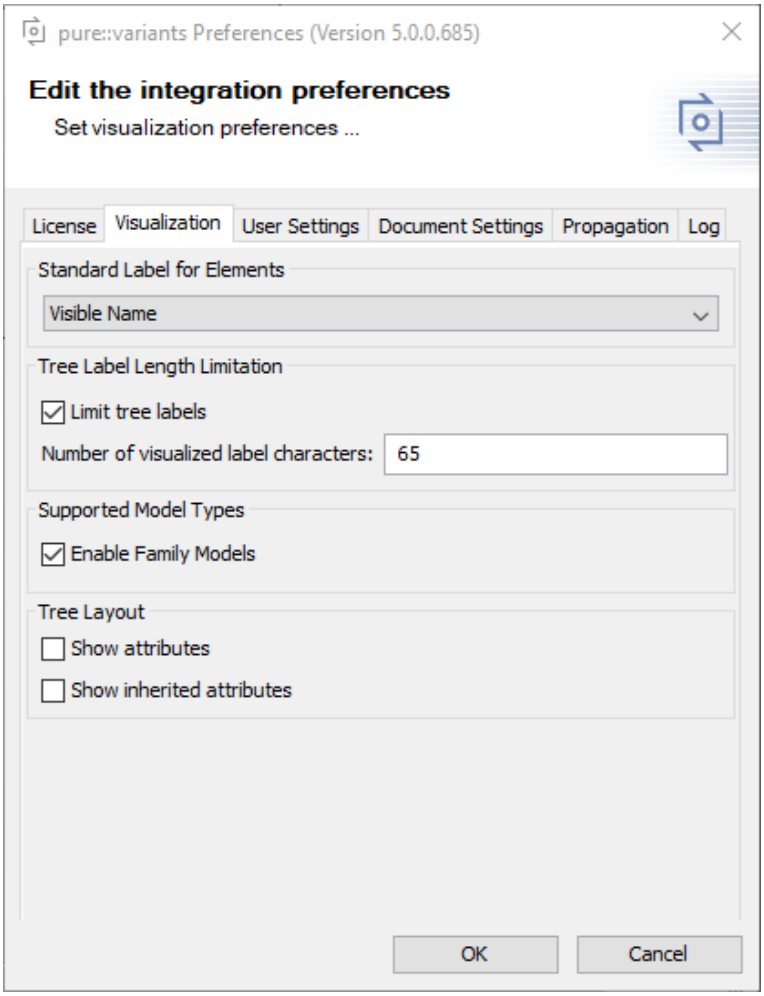
Removing Old Model References

Since the loading of models has changed extensively in pure::variants Integrations of version 4.x, Integrations now save pure::variants models differently in the current document than they did in version 3.2.x. Therefore, references to models loaded with version 3.2.x may still exist in your document, but are not used. This may be as intended if you still also need to open the document with an Integration of version 3.2.x. However, if this is not necessary you can remove these references from your document: Open the Integration preferences and press the **Remove References** button in the lower part of the **Log** tab (see Figure 28, “Preferences Dialog Log Tab”).

Model Visualization Preferences

In the Integration preferences, you can set how pure::variants models will be displayed and which model types are supported. To do that, open the Integration preferences by pressing and go to the **Visualization** tab (see Figure 17, “Preferences Dialog Visualization Tab”). The first dropdown box enables you to set how elements in the pure::variants model view are labeled. Furthermore, you can limit how many characters are shown for each element in the tree, enable or disable the loading of family models, and set whether attributes are shown in the model tree. To also show attributes inherited from parent elements, select **Show inherited attributes**.

Figure 17. Preferences Dialog Visualization Tab



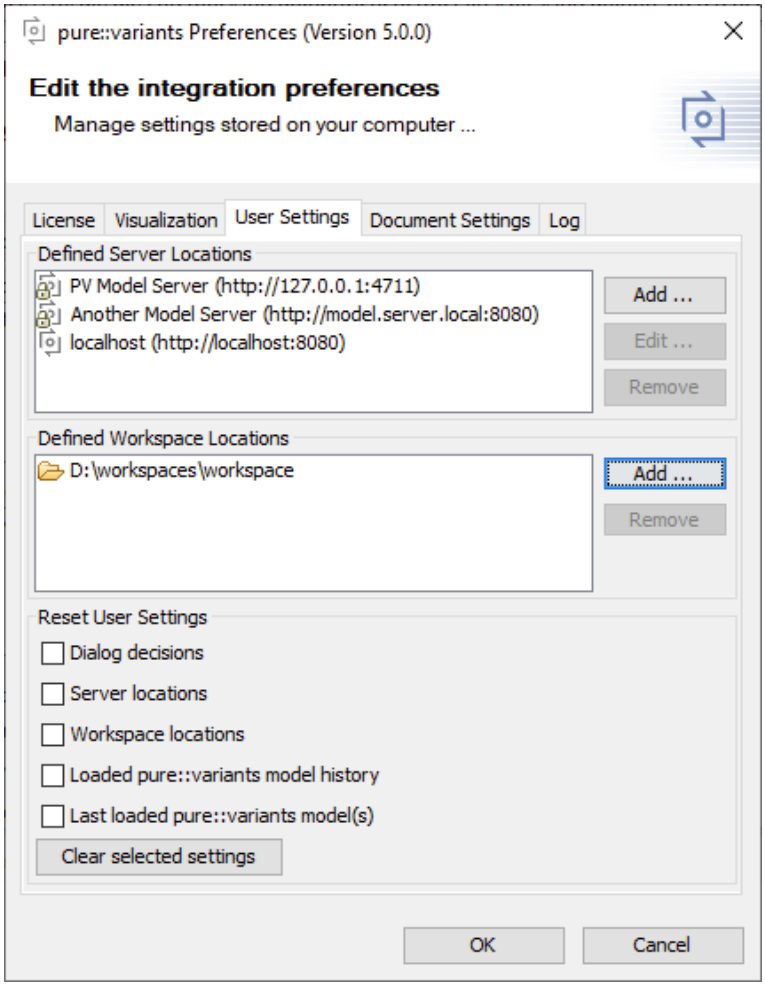
Manage Settings

In the Integration preferences, you can also manage settings stored by the Integration. You can, for example, add or remove known workspace and server locations or clear certain settings. To do that, open the Integration preferences by pressing .

Here, you can find tabs *User Settings* and *Document Settings*.

On tab *User Settings*, you can manage settings that are stored only on your local machine, specifically for your user. (see Figure 18, “Preferences Dialog User Settings Tab”). This includes server and workspace locations, dialog decisions, the history of previously loaded pure::variants models, and so on. The first section, **Defined Server Locations** enables you to add, edit and remove the server locations that are not locked. The second section, **Defined Workspace Locations** enables you to add and remove the workspace locations. The last section **Reset User Settings** enables you to clear the selected settings. For example, you can select checkbox *Last loaded pure::variants model(s)* and press the clear button, to make sure no pure::variants model is loaded from the user settings at startup of the Integration.

Figure 18. Preferences Dialog User Settings Tab



On tab *Document Settings* you can manage all settings that are stored in the current Rhapsody project. For example, you can clear the last loaded pure::variants model from the current document, so that, when you or other users next open the current Rhapsody project, the Integration will load no pure::variants model stored in the document.

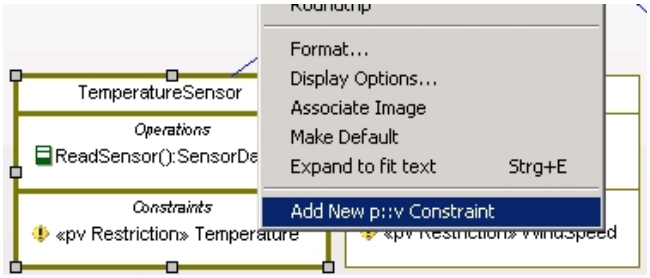
Adding and Editing Variability Information

To add and maintain the variability information of a IBM Rational Rhapsody project more efficiently and less error-prone, the Integration provides a pvSCL editor.

Accessing the Constraint Editor

You can open it through the context menu of various UML elements. In Figure 19, “Accessing the Constraint Editor” the constraint editor is being accessed to add a new pvRestriction constraint to a class. You can also do this for other UML elements, such as states, use cases, objects, etc. Furthermore, you can edit pure::variants constraints (**Edit p::v Constraint**), or convert other constraints to pure::variants constraints and edit them using the constraint editor (**Convert to p::v Constraint**). Adding or editing pure::variants Value Calculations is also possible (see the section called “Using pure::variants Value Calculations”).

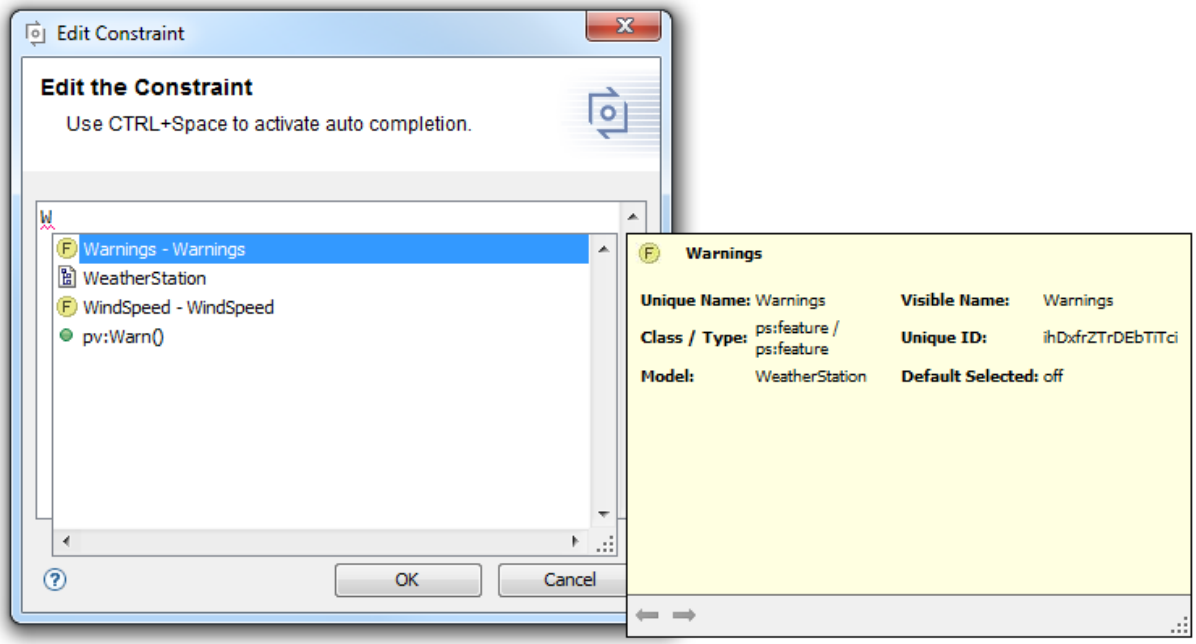
Figure 19. Accessing the Constraint Editor



Working with the Constraint Editor

The constraint editor features auto completion and syntax highlighting (see Figure 20, “Using the Constraint Editor”). You can automatically complete words by pressing CTRL + space. If more than one word is possible, a list containing possible keywords, features and pvSCL functions is displayed.

Figure 20. Using the Constraint Editor



Error Check

When you are done with editing the pure::variants constraint and closing the editor the entered expression is checked for errors. Errors in pvSCL expressions are reported if the expression's syntax is not pvSCL compliant, or if an element is unknown based on the loaded pure::variants models. Unknown elements are highlighted in red.

Calculations within Requirement Specification

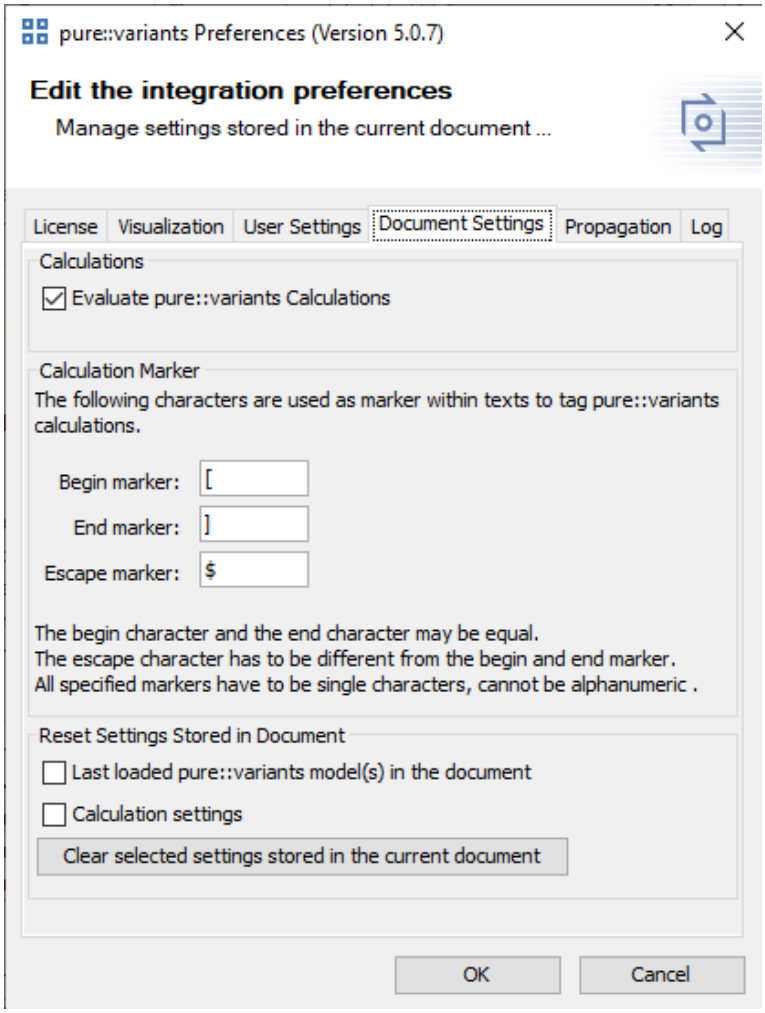
Texts in Requirement specifications may have variable parts, while the most part of the text will remain equal in all of your variants.

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, if you perform *partial text substitution* during the transformation of your variant. The actual values are also shown during the preview performed with the pure::variants Integration for IBM Rational Rhapsody.

A pvSCL statement starts with an opening marker (default: '[') followed by an pure::variants pvSCL calculation rule and is ended with an closing marker (default: ']'). To escape a statement the escape character is used (default:

"$"). This will prevent pure::variants from evaluating and replacing the escaped statement. The characters used for tagging a calculation are configurable. See Figure 21, “Calculation Settings”.

Figure 21. Calculation Settings

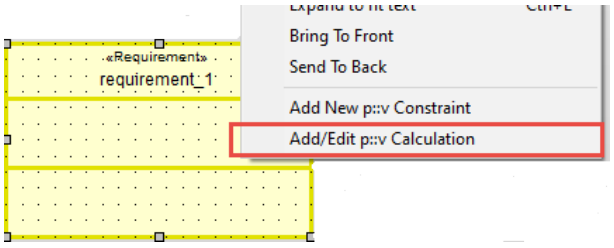


Example: *The maximum allowed speed is [Speed->Max] km/h.* in a requirement, it will be replaced with the value of attribute *"Max"* on Feature *"Speed"* in the exported variant. The result could be: *The maximum allowed speed is 100 km/h.*

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

After the markers are added to the requirement, the calculation editor can be opened via the context menu -> Add/Edit p::v Calculation. See Figure 22, “Add/Edit p::v Calculation”

Figure 22. Add/Edit p::v Calculation



Visualizing Variability Information

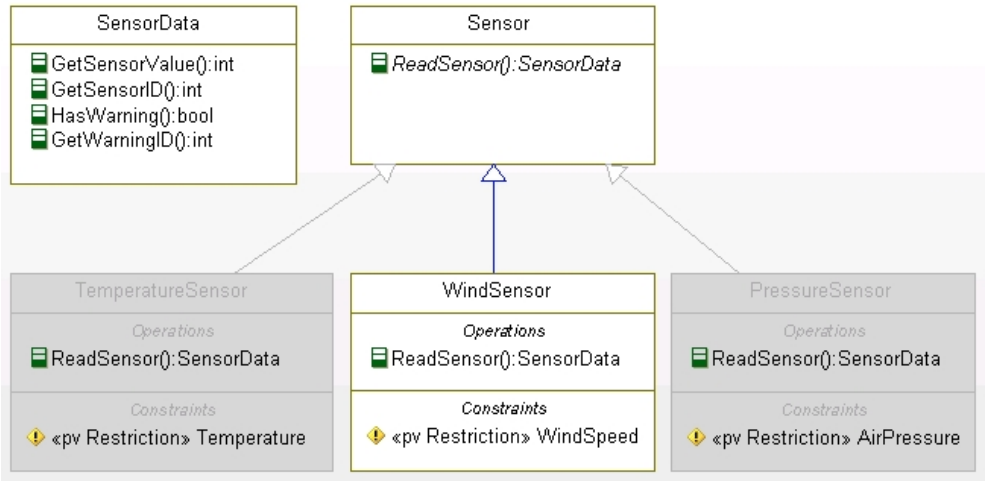
When you apply visualizations they affect all diagram elements of the loaded IBM Rational Rhapsody project. There are two types of visualizations: Preview visualizations and visualizations of errors in variability information. All visualizations color elements in the diagram and a small overlay is added to the element's icon. This icon overlay is also visible in the Rhapsody project browser.

Previews

Through preview visualizations you can preview the results of a transformation with pure::variants. For this visualization you need to load a variant model, based on which the visualizations are generated. There are two types of previews: Preview of excluded elements and preview of included elements.

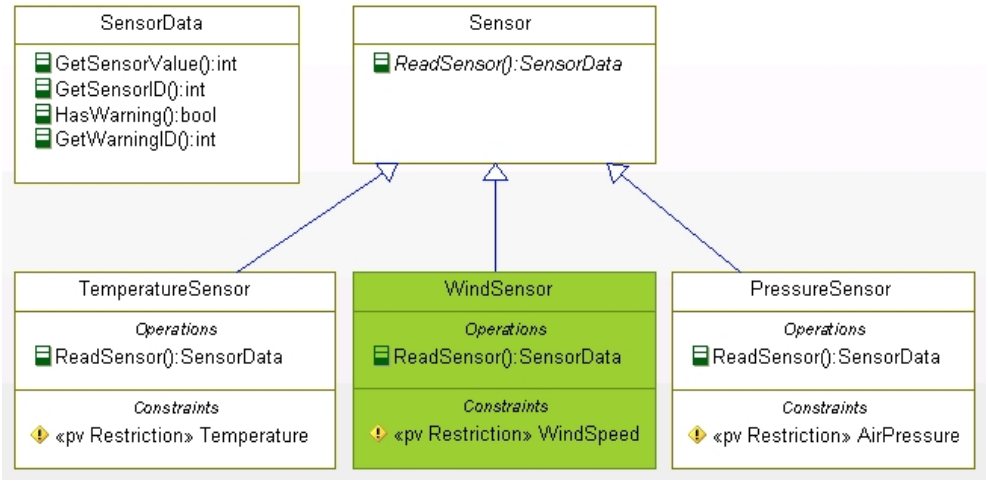
When you select the preview of excluded elements by pressing  , all elements that would be excluded in the corresponding variant are grayed out. Figure 23, “Preview of excluded elements” shows such a visualization in a class diagram. The classes TemperatureSensor and PressureSensor are grayed out, since the assigned features Temperature and AirPressure are not selected in the loaded variant model.

Figure 23. Preview of excluded elements



When you apply the preview of included elements by pressing  , all elements that would be included in the variant, and that are related to variability information, are highlighted in green. Figure 24, “Preview of included elements” shows such a preview. The class WindSensor is highlighted, since feature *WindSpeed* is selected in the loaded VRM. Sensor and SensorData, on the other hand, are not highlighted, because they are not related to variability information.

Figure 24. Preview of included elements



Problem Visualizations

Problem visualizations indicate where problems in variability information exist (problems in the pvSCL expression of pure::variants constraints). To be able to view problem visualizations, a pure::variants model needs to be loaded (see the section called “Connecting with pure::variants Models”). There are two types of problems: Elements with pvSCL errors are highlighted in red ( ), whereas elements with pvSCL warnings are highlighted in yellow with a red border ( ). Figure 25, “Error Visualization” shows an example for the error visualization. The class WindSensor contains the constraint "Not(WindSpeed)", which is a syntactic error, since the keyword "Not" has to be in uppercase letters. Thus, the class is highlighted in red.

Figure 25. Error Visualization

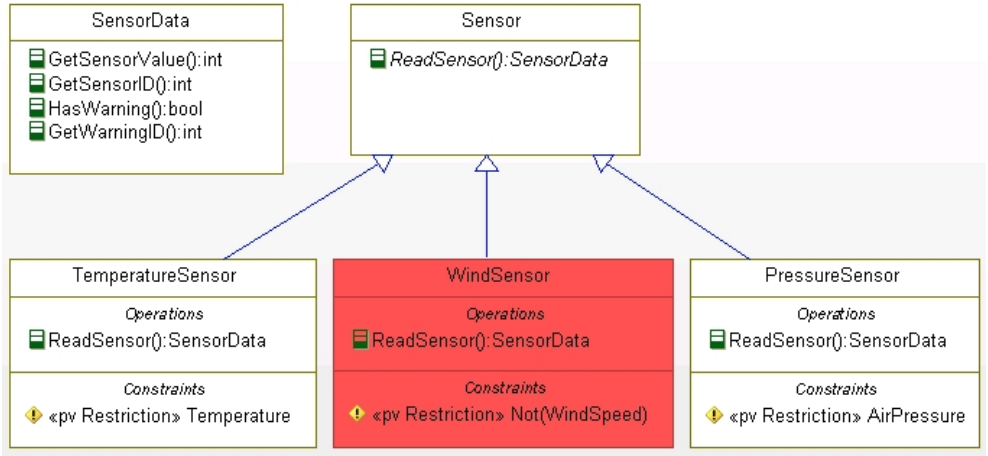
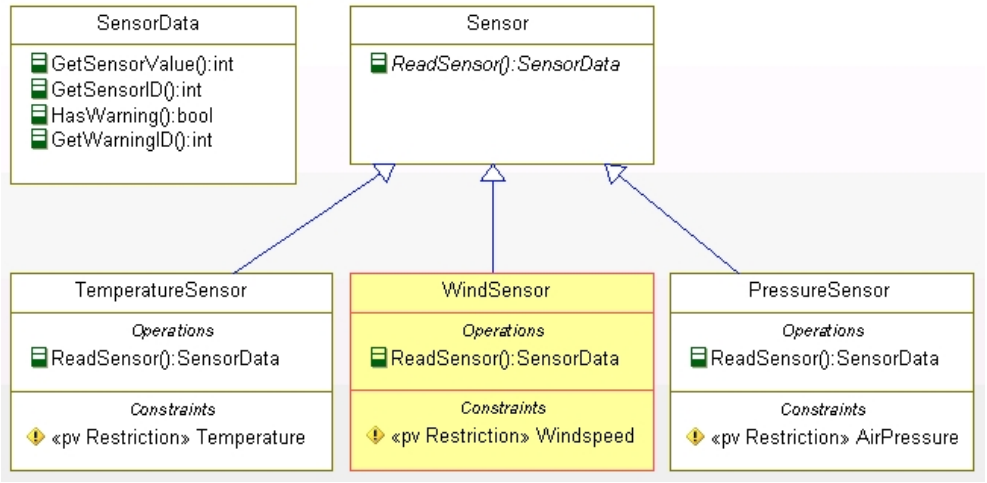


Figure 26, “Warning Visualization” shows the warning visualization. This time, the class WindSensor is highlighted, because it contains the constraint "Windspeed", which is semantically incorrect, since the loaded feature model does not contain a feature Windspeed (but it does contain a feature WindSpeed). You can easily identify this typing error through the warning visualization.

Figure 26. Warning Visualization



Customizing Preview

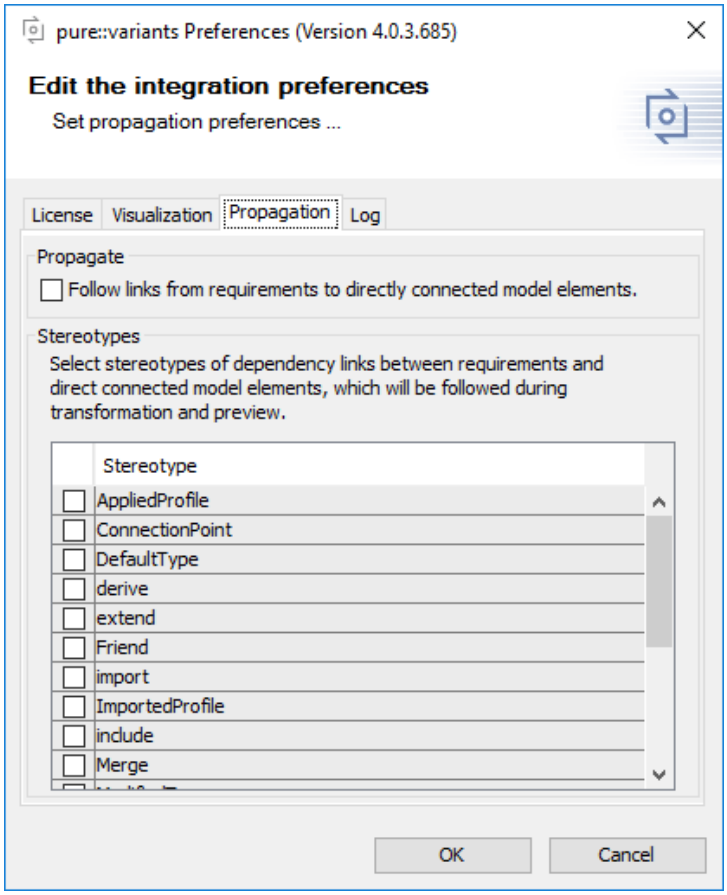
You can easily customize the colors that are used for each visualizations or change the icon overlay that is shown in the project browser. To this end, open pvRhapsody.prp in the pure::variants Integration for Rhapsody installation folder. For example, to change the fill color of one of the visualizations, you need to edit the RGB values given in double-quotes in the line starting with `Property Fill.FillColor`. To change the icon overlays, you can edit the existing .bmp icons. Please note that only bitmaps (.bmp) are supported, and that all parts of the image that are white are shown as transparent in the icon overlay.

Setting Propagation Preferences

As described in the section called “Using Requirements”, variability that is already added to a Rhapsody requirement can be propagated to all elements that are dependent on the requirement. To edit the propagation settings, open the Integration preferences by pressing , and go to the **Propagation** tab (see Figure 27, “Propagation Preferences”). Here you can enable or disable the propagation by selecting the first check box. Furthermore, you can adjust for which stereotypes of the dependencies the propagation will be performed. Per default the propagation is only performed for dependencies with the stereotype <<trace>>.

Please note, that this will affect the outcome of both visualization and transformation.

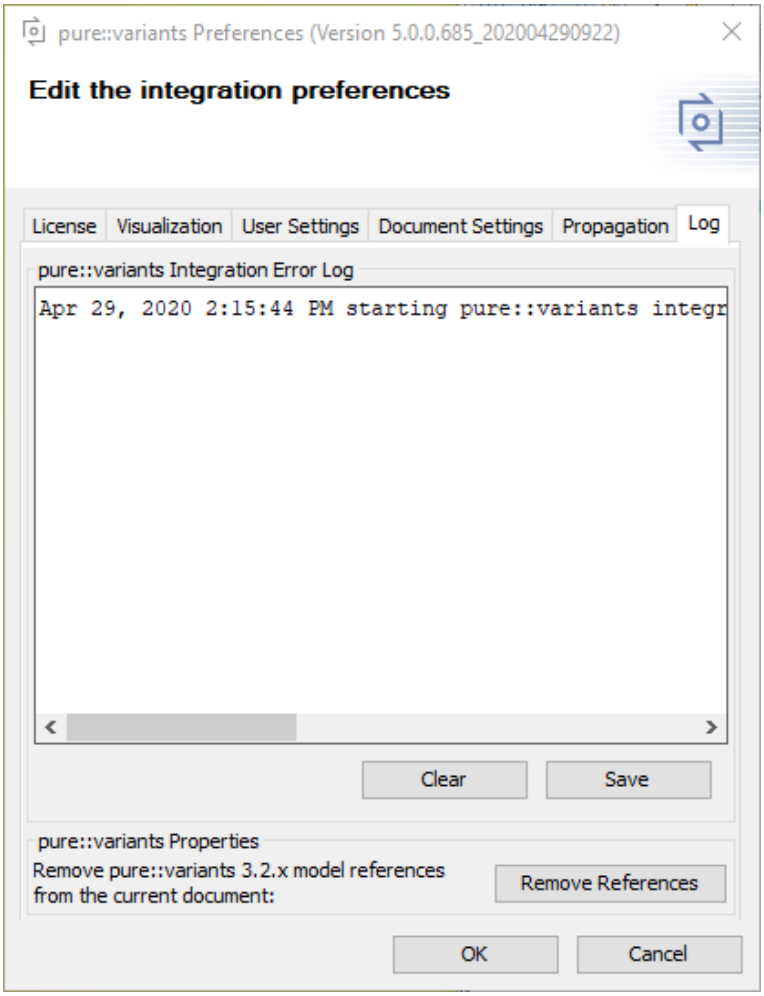
Figure 27. Propagation Preferences



Troubleshooting

If the Integration does not behave as expected, it may be useful to check its log file. You can find it in the Integration preferences ( ) on the **Log** tab (see Figure 28, “Preferences Dialog Log Tab”). It also enables you to save it to your disk or clear the contents of the log file. Furthermore, the **Log** tab provides the option to remove references to pure::variants models that can only be read by Integrations of version 3.2.x (see the section called “Removing Old Model References”).

Figure 28. Preferences Dialog Log Tab



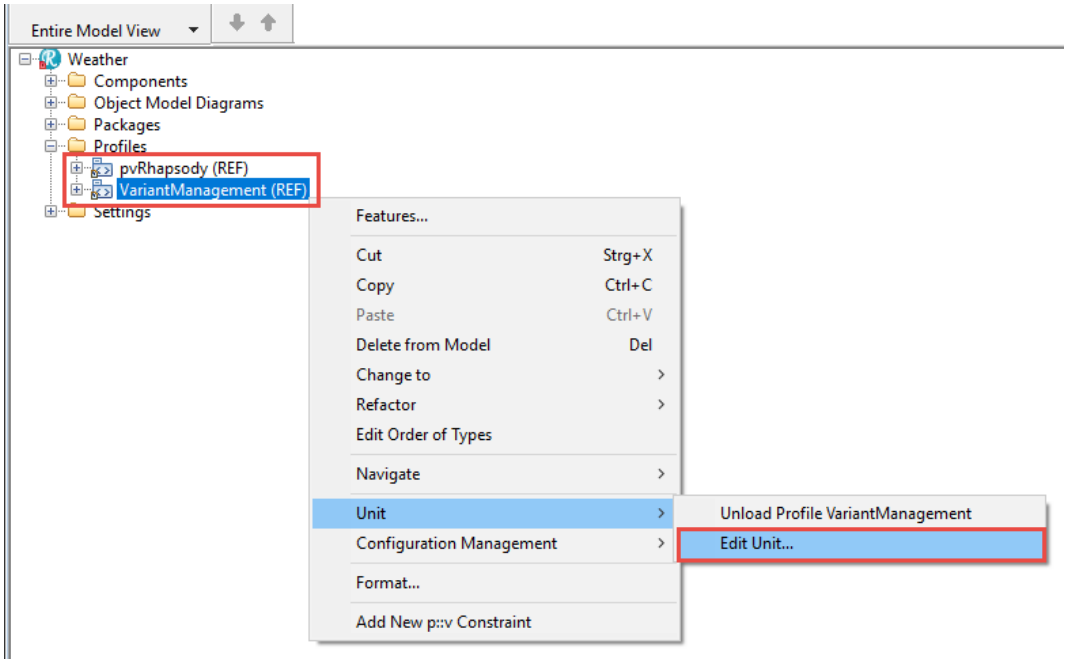
What to do when the Integration Window Does not Open?

If the Rhapsody Integration window does not show, even though you have added pvRhapsody.sbs to the model as described in Section 1.3, “Installation”, you can follow these steps to trouble shoot:

1. In Rhapsody, check whether the Integration window opens when selecting **Tools** > **pure::variants**.

2. If that does not work, find out where exactly pvRhapsody.sbs and VariantManagement.sbs are located. To do that, right-click the sbs in the Rhapsody model browser and select **Unit** > **Edit Unit**. In the Unit Information dialog, note the contents of "Directory".

Figure 29. Open Unit Information Dialog



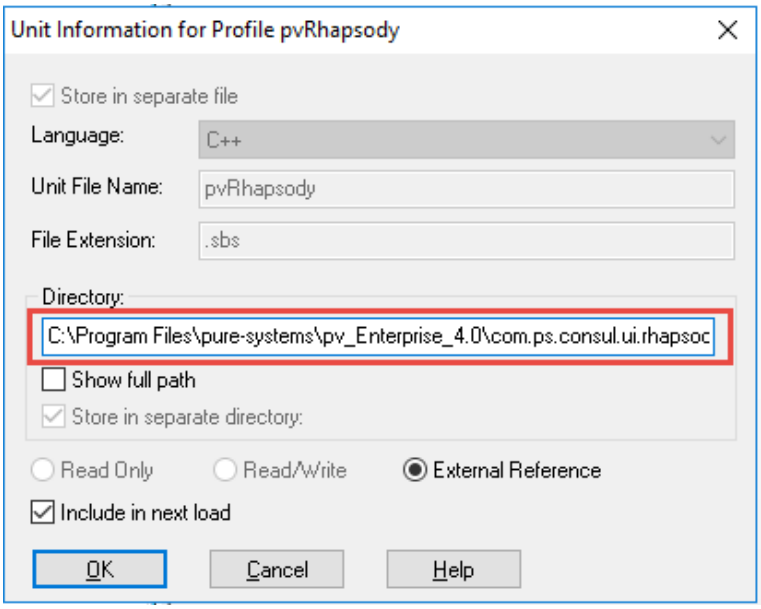
3. Make sure the "Directory" entry of both profiles points to the integration directory (default: `C:\Program Files\Parametric Technology\pv_Enterprise_6.0\com.ps.consul.eclipse.ui.rhapsody.integration`). If not, you can readd the profile via **File** > **Add Profile to Model** with the correct location. Restart Rhapsody and try again.

Note

It is possible to reference the profiles in another folder, but we recommend to use the Integration install directory, because it is updated by pure::variants. In any case, both Profiles must be referenced in the same folder, otherwise Rhapsody loads the needed jars from only one of the folders, which may lead to not understandable behavior.

4. In some cases referencing pvRhapsody.sbs and VariantManagement.sbs via a relative path caused the integration not to show. This may be the case if Rhapsody model property `General::Model::ReferenceUnitPath` is set to `relative`. To make sure an absolute path is used, open the unit information dialog for pvRhapsody.sbs again (via **Unit** > **Edit Unit**) and manually enter the absolute path in the **Directory** text box (default installation directory is: `C:\Program Files\Parametric Technology\pv_Enterprise_6.0\com.ps.consul.eclipse.ui.rhapsody.integration`). Do the same for VariantManagement.sbs, save and try to restart Rhapsody.

Figure 30. Unit Information for profile pvRhapsody



5. If it still does not work, make sure the folder that contains the referenced pvRhapsody.sbs also contains all other required files. The easiest way to do that is to reinstall the pure::variants Integration for IBM Rational Rhapsody. You can do that in pure::variants via **Help** >**pure::variants** > **Tool Integration Updates**. Just select the Rhapsody Integration, and press Finish. If it is already installed, the current installation will be overwritten. After a restart of Rhapsody, the Integration window should be shown.

   ### Note

   If you can't reinstall the integration, because you don't have administrator permissions, and you also have another Rhapsody project in which the integration window opens successfully, you can also do as follows: Open that other Rhapsody project, find out the folder of pvRhapsody.sbs and reference the same pvRhapsody.sbs as in the project that shows the integration successfully. Make sure that you reference VariantManagement.sbs in the same folder.

Known Issues

- When the pvRhapsody.sbs profile cannot be found at startup, because the Rhapsody project has moved, Rhapsody asks for the new path to the profile. However, the necessary data for preview is not loaded after specifying the new path. In this case visualizations will only work after a save and restart of the Rhapsody project.

Undo/Redo Behaviour Notes

After using any function of the pure::variants Integration for IBM Rational Rhapsody, Rhapsody's undo or redo function cannot be used any more.

2.6. Creating a pure::variants Project for Rhapsody using the New Project Wizard

Creation of the corresponding pure::variants project is supported by a "New Project" wizard. It creates a standard project with most common settings, allowing to start quickly.

To start the wizard, select **File->New->Project->Variant Management->Variant Project**.

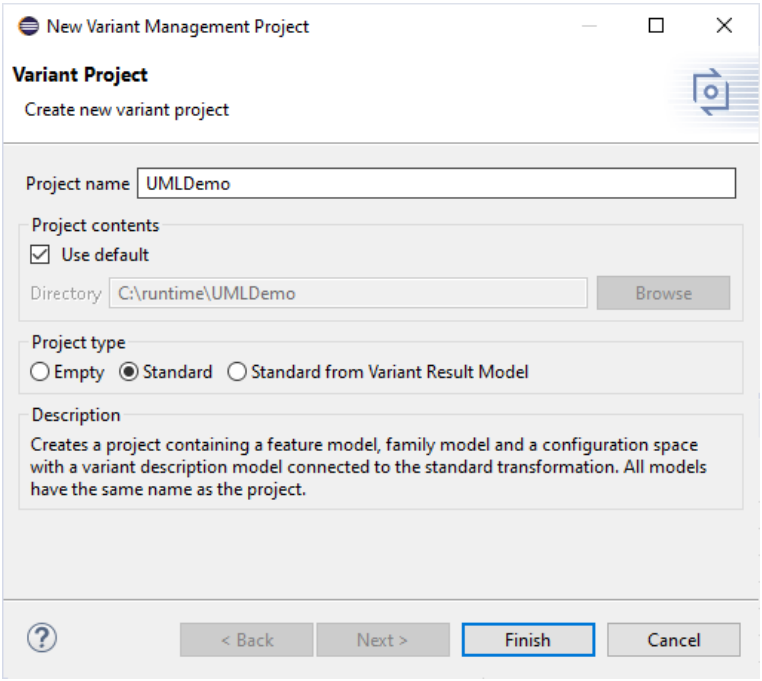
The wizard will ask for the name of the project on the first page (see Figure 31, “Running the pure::variants Variant Project wizard”). You should select the **Standard** project type, because this will create an initial pure::variants project with all needed models.

Note

If “Empty” is selected, just an empty project is created, to which all information has to be added manually.

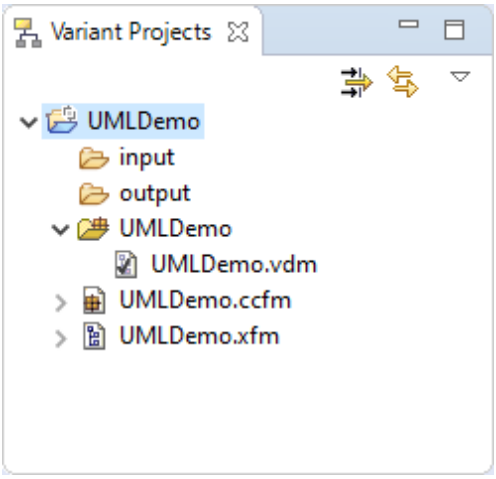
Only in special cases the next page is of relevance. Here other projects can be marked as referenced projects.

Figure 31. Running the pure::variants Variant Project wizard



The wizard creates several models. All have the name of the project (`UMLDemo` in this case). It creates a feature model with just the root feature named like the project, a configuration space, an initial variant description model in this configuration space and two folders `input` and `output`. The `input` folder should contain the Rhapsody project document(s) to transform. The `output` folder will hold generated model variants. The layout of the project is shown in Figure 32, “Layout of a Standard pure::variants project”.

Figure 32. Layout of a Standard pure::variants project

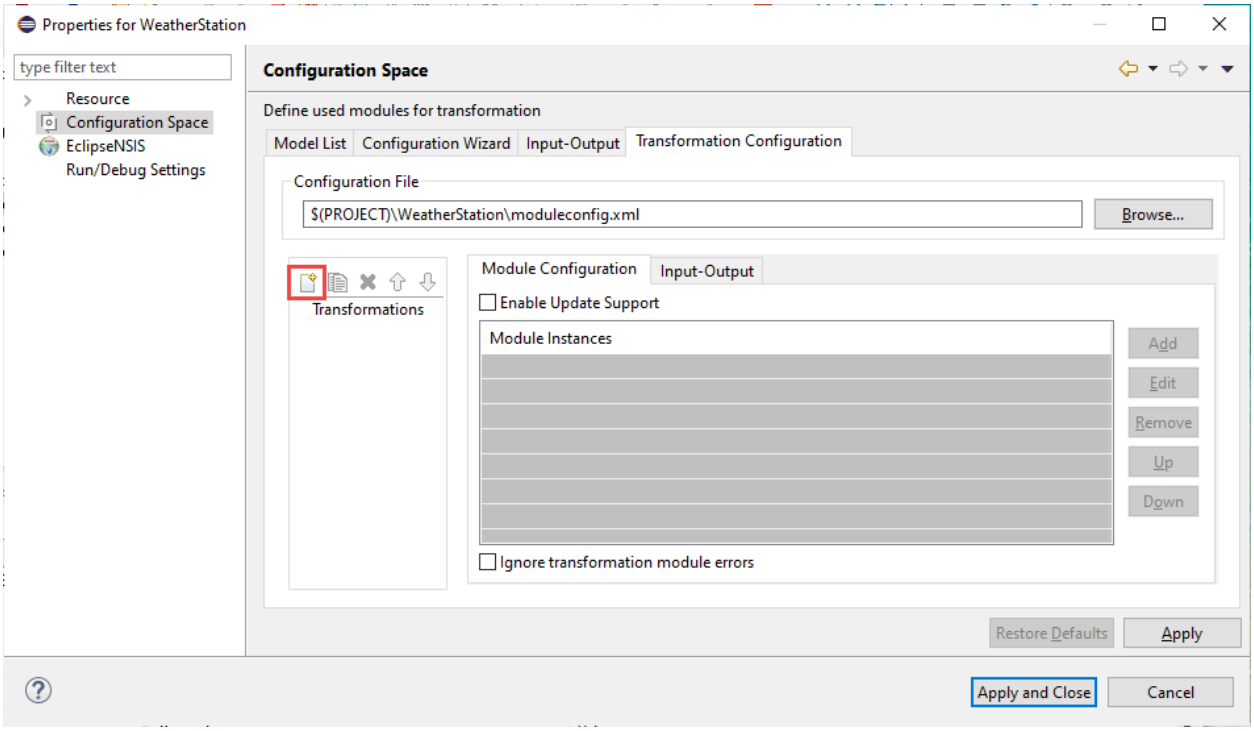


2.7. Adding a Rhapsody Transformation to pure::variants Projects for Rhapsody

For transforming Rhapsody project documents, a transformation module is needed. However, pure::variants projects created with the Variant Project Wizard do not contain a transformation module per default.

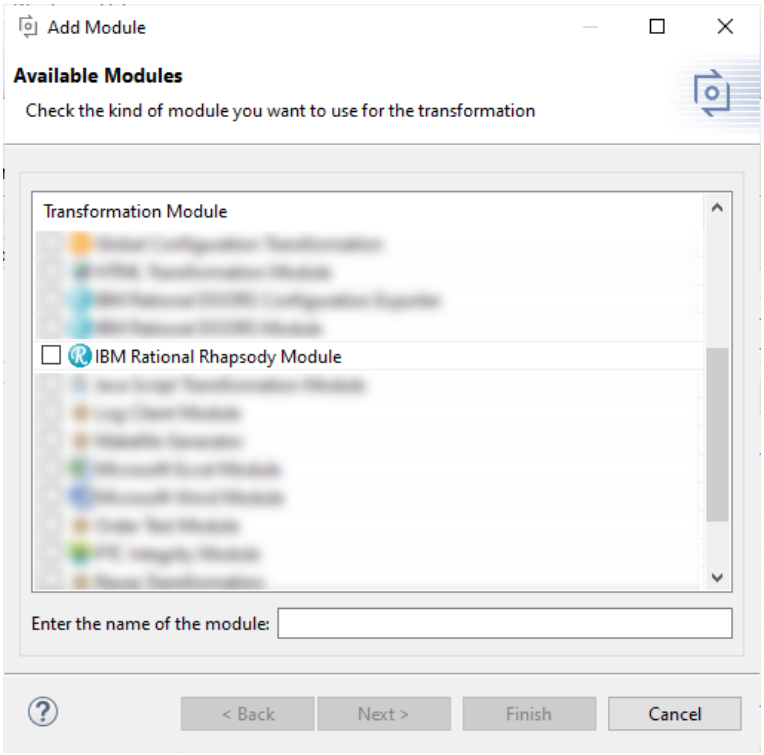
To add a Rhapsody Module, open the configuration space properties of the pure::variants project, i.e. right-click on the configuration space in the Projects View and choose **Properties** from the context menu. In the properties dialog switch to page **Configuration Space** and there to tab **Transformation Configuration**.

Figure 33. Transformation Configuration



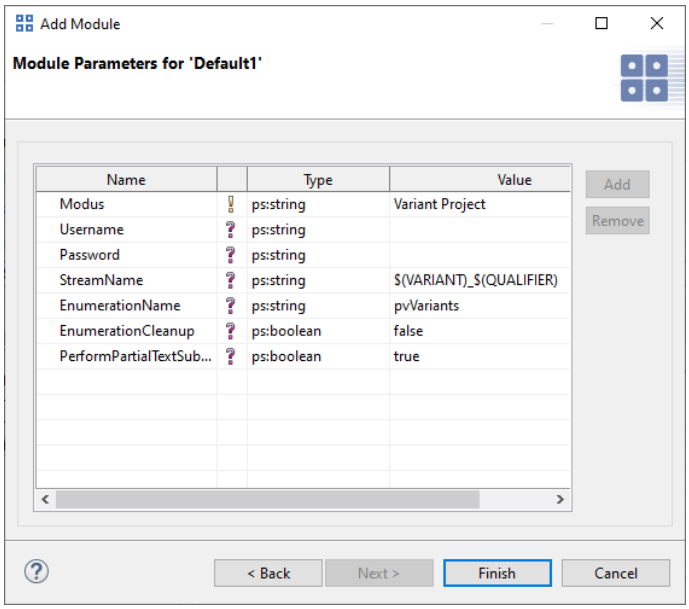
Create a new module configuration with the marked button in the image above. Click on button **Add** after creating the new module configuration. This opens the transformation module selection dialog as shown in Figure 34, “Transformation Module Selection Dialog”.

Figure 34. Transformation Module Selection Dialog



Select the IBM Rational Rhapsody module and enter a name then click on **Next**. On the following dialog page the transformation parameter of the IBM Rational Rhapsody Module can be specified (see Figure 35, “IBM Rational Rhapsody Transformation Parameters”).

Figure 35. IBM Rational Rhapsody Transformation Parameters



The meaning of the parameters is as follows:

Parameter	Meaning
Modus	Choose between the export modes. "Variant Project" copies the Rhapsody input project, including all variant-specific artefacts, to the given output file or Design Management

Parameter	Meaning
	stream. "Variant Enumeration" adds the current variant for selected Rhapsody elements to the element's enumeration tag. See Section 2.9, “Supported IBM Rational Rhapsody Transformation Modes” for details.
Username, Password	For IBM Rational Rhapsody Server projects the username and password is required for authentication. These can be defined optionally. If not defined, a login dialog pops up while transformation process.
StreamName	For IBM Rational Rhapsody Server projects a new stream is created for each variant. The new variant streams are named following this pattern. The default pattern will ensure a unique name for each transformation. Thus it should be edited with care.
EnumerationName	The name of the tag that is used for "Variant Enumeration" mode. If the value is empty the standard name "pvVariants" is used. See the section called “Variant Enumeration Mode” for details.
EnumerationCleanup	Clean up the variant enumeration tags by removing all variant names that are not part of the current transformation (Only for "Variant Enumeration"). See the section called “Variant Enumeration Mode” for details.
PerformPartialTextSubstitution	If true is selected, the partial text substitution is performed. See Section 2.9.1, “Partial Text Substitution” for details.
WorkItem	If set to valid work-item URI, the work-item is linked to the changeset, created by the transformation module.
WorkItem	If set to valid work-item URI, the work-item is linked to the changeset, created by the transformation module.

Note

You can use *User Parameters* to refer to a user defined parameter value. (Please see pure::variants User's Guide for details)

E.g. You can define a user parameter, like *WORKITEM*, in scope of your transformation module configuration, and refer to its value by writing $(PARAM:WORKITEM).

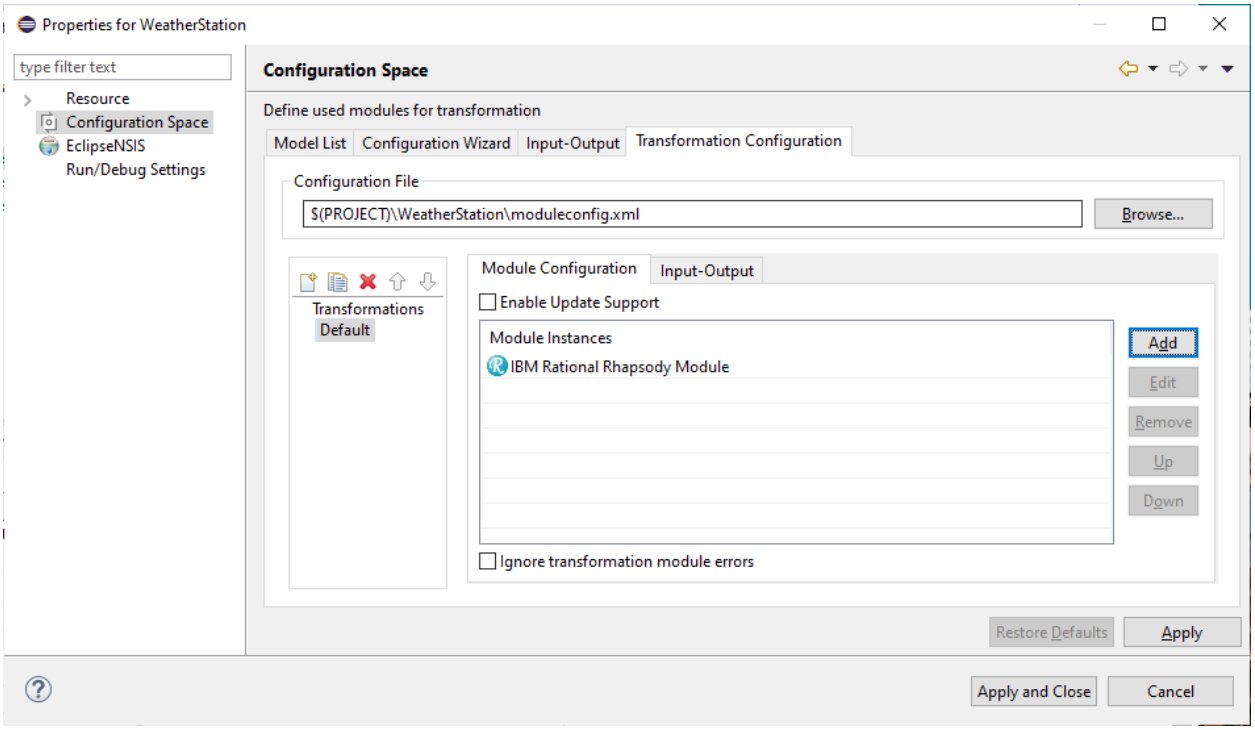
The value for *WORKITEM*parameter is queried by transformation framework in the beginning, once.

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 is used during transformation 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 transformed project is located in.
- $(STREAM)
  - Adds the name of the Stream, which is used in the project transformation.

The transformation configuration should then look as shown in Figure 36, “Transformation Configuration with Rhapsody Transformation”.

Figure 36. Transformation Configuration with Rhapsody Transformation



The new transformation is inserted at the end of the transformation module list. It is strongly recommended to leave this module on the last position in the module list.

Note

If you select the **Enable Update Support** checkbox above the list of modules, your variant output will be prepared for merging custom changes made in a variant with a newly transformed version of that variant. For details, see Section 2.12, “Using the IBM Rational Rhapsody Project Variants” and section *Variant Update* in the pure::variants User's Guide.

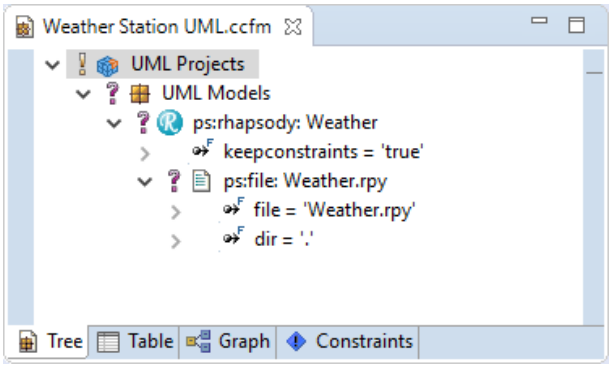
2.8. Adding IBM Rational Rhapsody Projects to pure::variants Family Models

To specify one or more Rhapsody projects that should be processed during transformation, you need to add Rhapsody project information to a family model. The pure::variants connector for IBM Rational Rhapsody is compatible with Rhapsody project files as well as Rhapsody server projects. The connector recognizes in both cases the family model part `ps:rhapsody` as trigger for a transformation.

Adding IBM Rational Rhapsody Project File

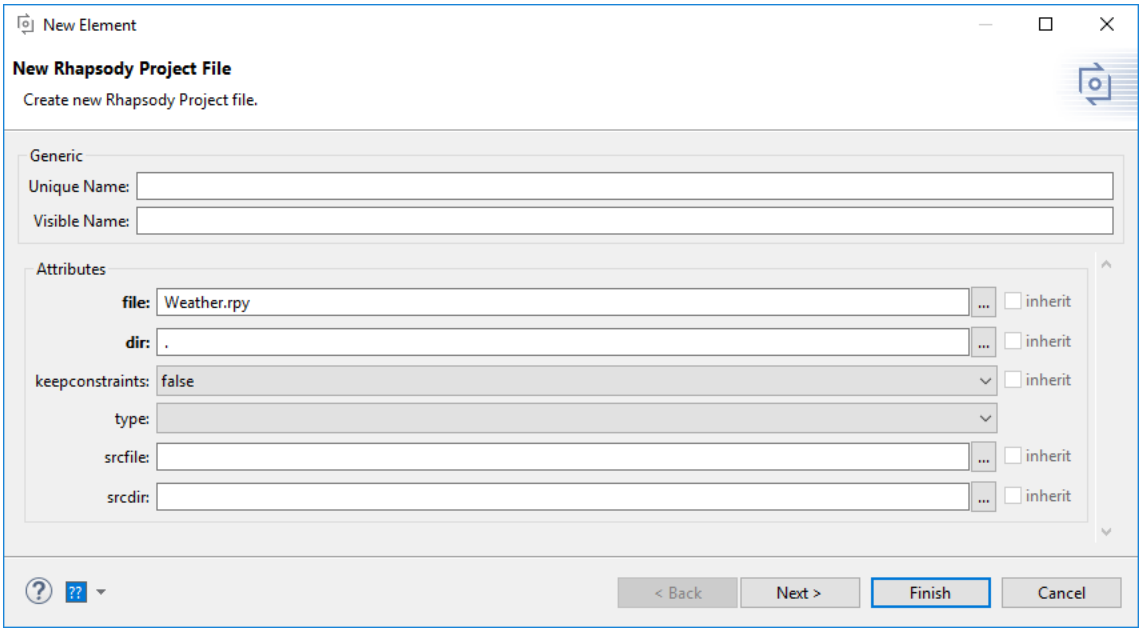
Below this family model part at least one `ps:file` element pointing to an IBM Rational Rhapsody project file (this is detected by checking the suffix `.rpy`, `.RPY`, `.rpyx` or `.RPYX`) has to be set.

Figure 37. Family Model containing Rhapsody project information



The family model part `ps:rhapsody` is added to a family model by right-clicking on a component element and then choosing **New->Rhapsody Project File** from the context menu. This opens the wizard for a new Rhapsody project file part as shown in Figure 38, “New IBM Rational Rhapsody Project File wizard”. Enter the name and path to the project file, or navigate to an existing project file by clicking on button **...** to the right of field **file**. Set **keepconstraints** to true if you want to keep the constraints during transformation. If it is set to false, all pure::variants constraints (pvRestriction, pvValueCalculation or pvValueChoice) will be removed from the transformation output. After finishing the wizard the part and the file element is created as shown in Figure 37, “Family Model containing Rhapsody project information”.

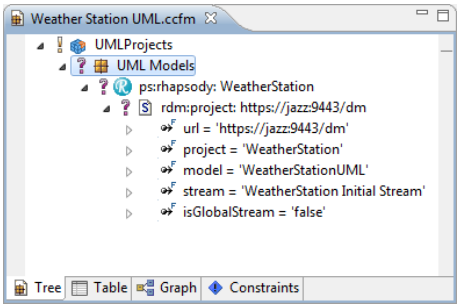
Figure 38. New IBM Rational Rhapsody Project File wizard



Adding IBM Rational Rhapsody Design Manager Server Project

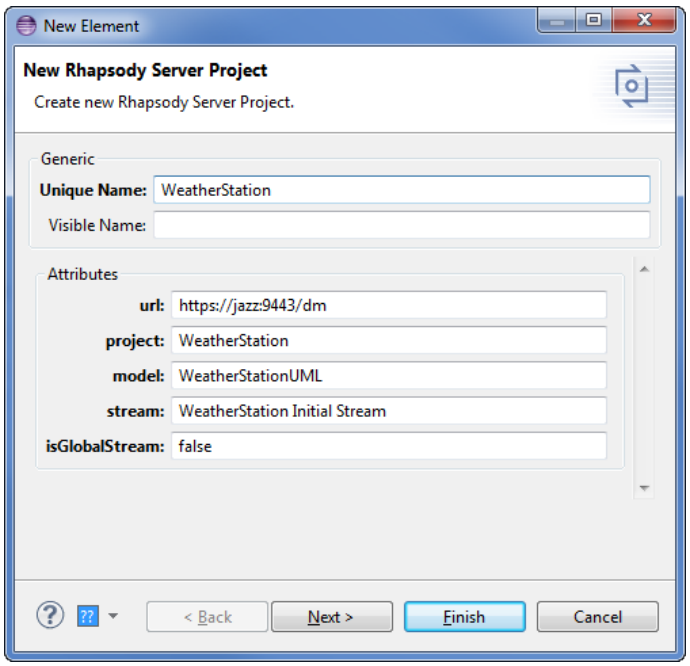
Below this family model part at least one `rdm:project` element pointing to an IBM Rational Rhapsody server project has to be set.

Figure 39. Family Model containing Rhapsody server project information



The family model part `ps:rhapsody` is added to a family model by right-clicking on a component element and then choosing **New->Rhapsody Server Project** from the context menu. This opens the wizard for a new Rhapsody server project part as shown in Figure 40, “New IBM Rational Rhapsody Server Project wizard”. Enter the server address, project name, model name, and initial stream name. Additionally define if the initial stream is a global configuration or Design Management configuration. After finishing the wizard the part and the project element is created as shown in Figure 39, “Family Model containing Rhapsody server project information”.

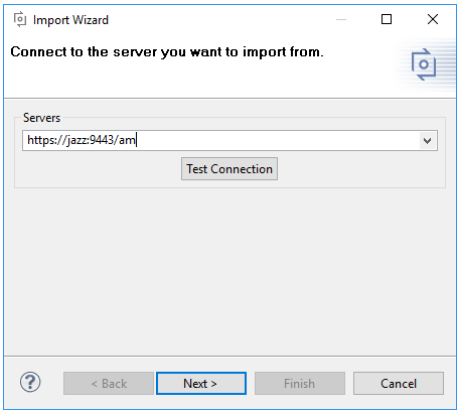
Figure 40. New IBM Rational Rhapsody Server Project wizard



Adding IBM Rhapsody Model Manager Server Project

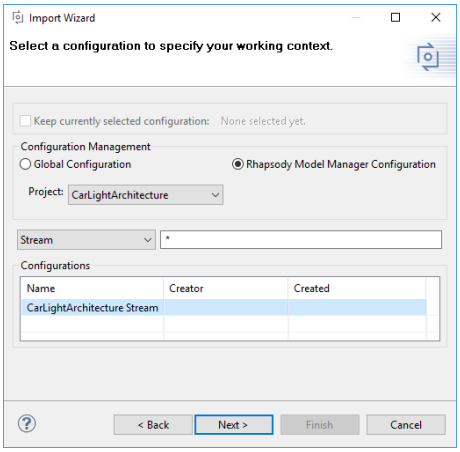
To add a new entry for a Rhapsody model, which is shared by the Rhapsody Model Manager, you have to make a context-click on a component element and then choose **New-->New Rhapsody Model Manager Project**. A import wizard opens and at first you need to define a Rhapsody Model Manager (Architecture Management) server address.

Figure 41. Import Wizard - Server Selection page



Press **Next** to navigate to the Configuration Selection page. Please choose a configuration originating from the Global Configuration Management or the Rhapsody Model Manager.

Figure 42. Import Wizard - Configuration Selection page

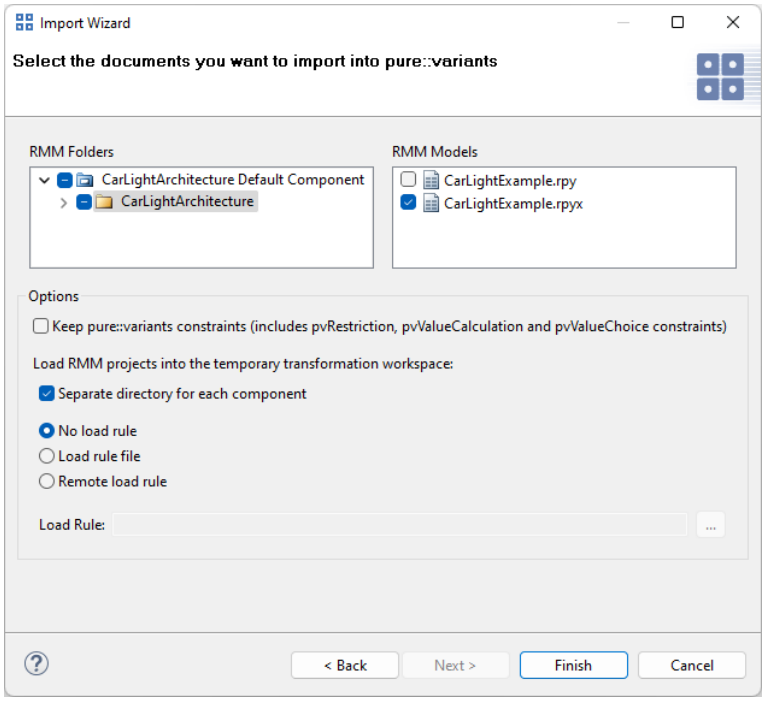


Note

If you are using Variant Enumeration mode (see the section called “Variant Enumeration Mode”), the selected configuration must be a writable stream. So it must not be a snapshot or baseline.

Press **Next** to reach to the Model Selection page. You have to navigate over the folder structure to get to your specific Rhapsody Model files and select them.

Figure 43. Import Wizard - Model Selection page



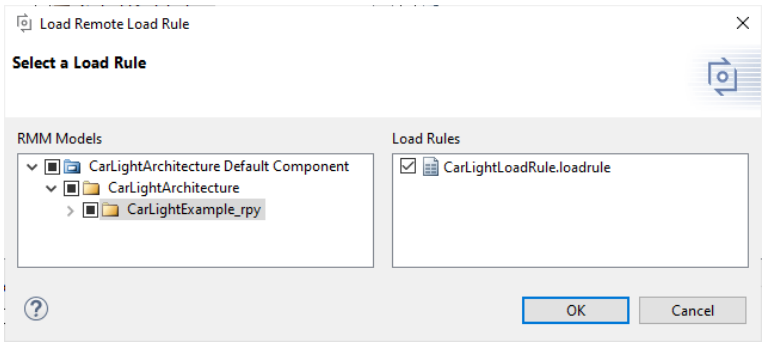
Select **Keep pure::variants constraints** if you want to keep the constraints during transformation. If it is not selected, all pure::variants constraints (pvRestriction, pvValueCalculation or pvValueChoice) will be removed from the transformation output.

The **separate directory for each stream** allows to define, if all Rhapsody projects from different streams should be loaded into a separate directory while transformation. Activating this option makes sense, if Rhapsody projects are indpendent from each other or EWM-component names may conflict (for differnt streams). If not activated, the Rhapsody projects are loaded into a common folder. This is recommended, if Rhapsody projects (from different streams) are interdependent and are assembled in a Global Configuration context.

A load rule provides flexibility in choosing which items are to be loaded from stream and where in the sandbox they should be located. If a load rule is not set, all items in the selected component are loaded, as checked-in.

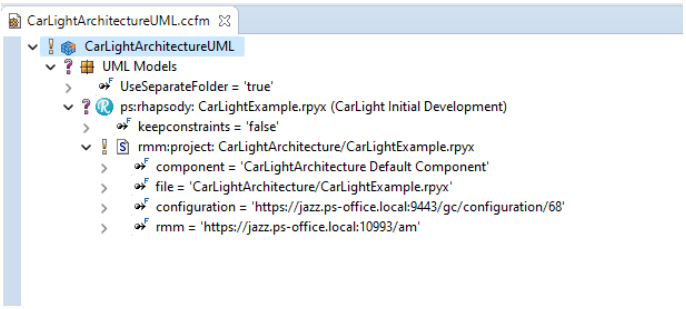
If you want to use a local load rule, select **Load rule file** and press '**...**' to browse for the file. To use a remote load rule select **Remote load rule file** and press '**...**' to reach the 'Select Remote Load Rule' page. Navigate over the folder structure to get to your specific load rule file and select it. Press **OK** to add the remote load rule. It is also possible to use p::v variables, like $(PROJECT) in the loadrulefile attribute for local load rule files. (see Figure 46, “Load rule with sandbox relative path”)

Figure 44. Select Remote Load Rule page



Press **Finish** to import the Rhapsody Model references into your current Family Model. See below an example model reference imported from Rhapsody Model Manager.

Figure 45. Rhapsody Model Manager model reference

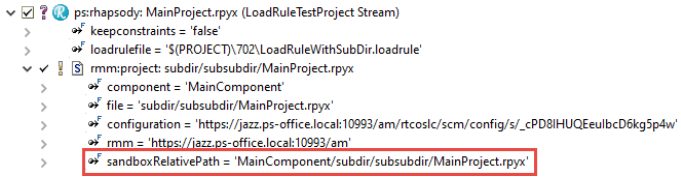


If the option **separate directory for each stream** was selected, the attribute **UseSeparateFolder=true** is added to the parent model element (e.g. *UML Models* in screenshot) of the Rhapsody project reference. This option affects all Rhapsody project references that are added below that model element.

Otherwise, if the option **separate directory for each stream** was unselected, the attribute **UseSeparateFolder=false** is added to the parent model element. With this setting, all Rhapsody projects are loaded into the same directory during the transformation.

If a **load rule** is added, it is possible that the transformation fails to find the .rpyx file in the temporary sandbox. In such cases, please add an attribute with name**sandboxRelativePath** to the rmm:project element. This attribute forces the transformation to use a specific path to the .rpyx file. The given path should be relative to your sandbox and should point to the .rpyx file. For example, if your sandbox was located in C:/ewm/sandbox and the .rpyx file would be in C:/ewm/sandbox/componentname/subfolder/rhapsodyproject.rpyx, you would need to add an attribute "sandboxRelativePath" with value "componentname/subfolder/rhapsodyproject.rpyx".

Figure 46. Load rule with sandbox relative path



Authentication

To use the the connector it is always required to be logged-in to the Rhapsody Model 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.

Transforming with a specific Rhapsody version

Per default, the Rhapsody transformation is done with the Rhapsody version that was last installed on your computer. However, you can also force to use a specific Rhapsody version by specifying the location of the rhapsody.ini file in the preferences (see **Window** > **Preferences**> **Variant Management** > **Connector Preferences** > **Connector for IBM Rational Rhapsody**).

When a rhapsody.ini file is set, the transformation uses the given Rhapsody version and the Java version specified in rhapsody.ini. This may also speed up the transformation if Rhapsody takes a long time to start up on your computer. Furthermore, it supports scenarios where a 64bit Rhapsody version and only a 64bit Java is installed

on the computer. When no rhapsody.ini is set, it is necessary to also install a 32bit Java, even when you are using a 64bit Rhapsody.

2.9. Supported IBM Rational Rhapsody Transformation Modes

The transformation of Rhapsody projects supports two different modes: The *Variant Project* mode and the *Variant Enumeration* mode. The mode can be set through the Rhapsody transformation module parameter *Modus* (see Section 2.7, “Adding a Rhapsody Transformation to pure::variants Projects for Rhapsody” for a list of all parameters).

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.

Variant Project Mode

Variant Project mode is the standard transformation mode. When it is selected in the Rhapsody transformation module, the transformation first copies the Rhapsody input project to the given output project location. This can be either an input and output project located on the file system, or a Rhapsody Model Manager project, which is copied to another stream. Then, this copy is adapted so that

- all elements for which all connected pvRestrictions evaluate to false are removed
- for all tags attributes or value properties with a pure::variants value calculation, the value is set to the evaluation result of the pvValueCalculation's specification
- for all tags with a pure::variants value choice, the value is set to the value of the pvValueChoice's first child tag

Variant Enumeration Mode

When *Variant Enumeration* mode is selected, the transformation does not copy the input project. Instead the tag *pvVariants* is added to all variable elements and their children. If an element is part of the currently transformed variant, the variant name is added as a value to the *pvVariants* tag. If it is not, the name is removed from the tag's values.

The tag can also have a custom name specified via transformation module parameter *EnumerationName*. Furthermore, it is possible to force the transformation to clean up the *pvVariants* tags before each transformation. This can be done via the transformation module parameter*EnumerationCleanup*. If it is set to true, the existing *pvVariants* tags are removed before starting the new transformation. If multiple variants are transformed at once, this cleanup is only done before the first variant is transformed.

Note

You can use these *pvVariants* tags to filter your Rhapsody project. To this end, define a Rhapsody query based on the created*pvVariants* tags. For each query, Rhapsody automatically adds a new view, which can be selected from the drop down box in the Rhapsody project browser. When a view is selected, the project browser is filtered based on the respective query and optionally also the diagrams are filtered (can be enabled by selecting **Custom View...** from the same drop down box and checking **Apply on diagrams**).

Note

If you are using an RMM transformation and run a transformation in Variant Enumeration mode, the RMM configuration must be a writeable stream. So it must not be a snapshot or baseline.

Change Scenarios for Repeated Transformation

Regarding repeated transformation in variant enumeration mode, there are several change scenarios:

- Item X is still part of the transformed variant

- the existing pvVariants tag is not changed

- Item X is not part of the transformed variant anymore, because the connected pvRestriction evaluates to false
  - the variant name is removed from the pvVariants tag
  - if item X does not belong to any other variant, an empty pvVariants tag remains
- Item X is now a (new) part of transformed variant, because the connected pvRestriction evaluates to true
  - if no pvVariants tag existed, it is now created
  - the variant name is added to the values of the pvVariants tag
- Item X is not variable any more, because a previously connected pvRestriction was removed
  - the pvVariants tag is always removed (also if the pvVariants tag still lists other variant names)

2.9.1. Partial Text Substitution

'PerformPartialTextSubstitution' attribute in the transformation module parameter can be used to decide if partial text substitution has to be performed during the transformation or not. If **true** is selected, the partial text substitution is performed.

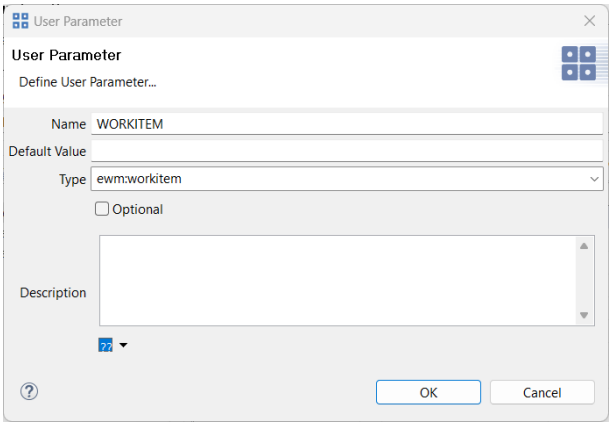
2.10. Linking Work-Item to a Change Set in RMM Transformation

A work-item can be linked to a change set during a RMM transformation by creating a user parameter, adding it to the transformation module and finally selecting or creating the work-item during the transformation.

Creating a User Parameter

To link a work-item with a change set, create a new user parameter of type **ewm:workitem**. (See Figure 47, “Adding Work-Item as User Parameter”.) Please refer to pure::variants User's Guide for details on User Parameters.

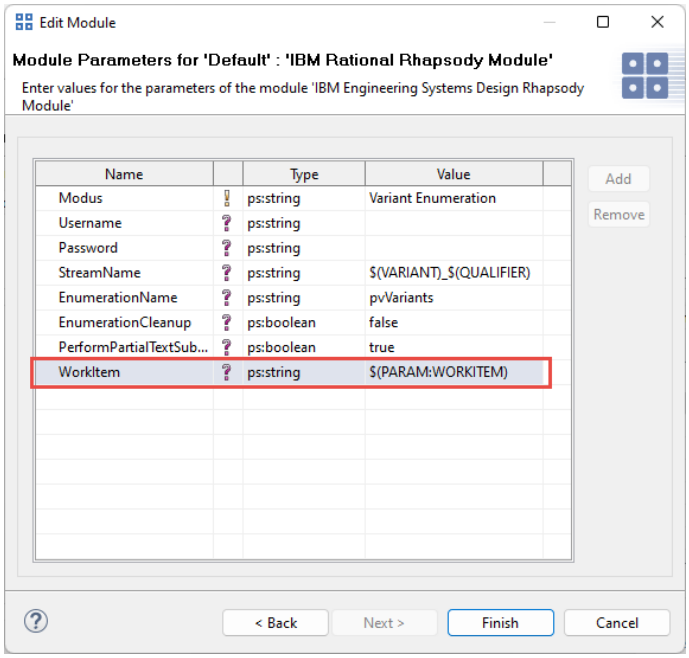
Figure 47. Adding Work-Item as User Parameter



Adding a Transformation Module Parameter

The user paramter defined in the previous step has to be linked to the transformation module configuration, by adding a value to the transformation module parameter with name **WorkItem**. The value should match the name defined in the user parameter page. For example: **$(PARAM:WORKITEM)**. This parameter is queried during the transformation. (See Figure 48, “Linking Work-Item in Module Configuration”.)

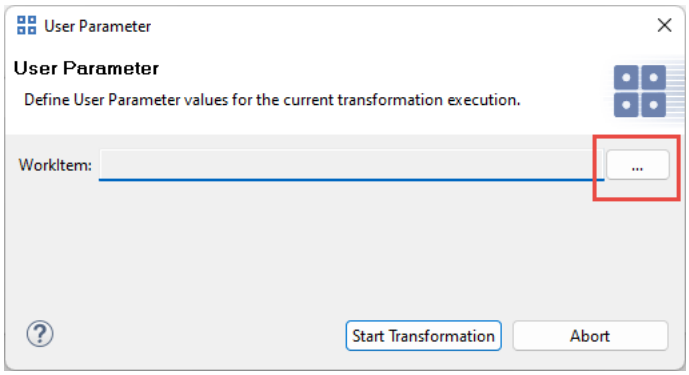
Figure 48. Linking Work-Item in Module Configuration



Work-Item Selection Dialog

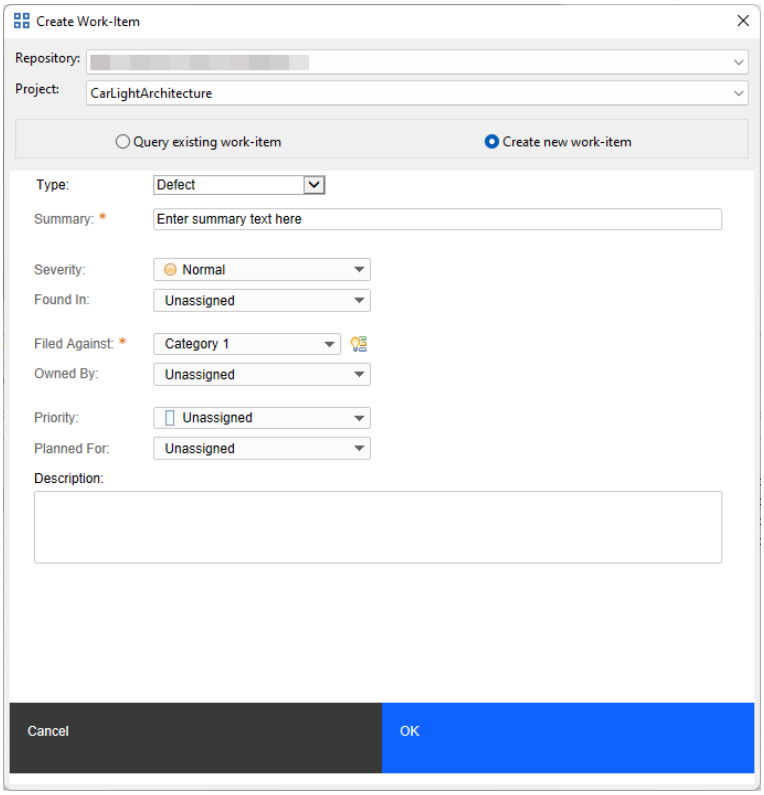
During transformation, you can select the work-item from the User Parameter dialog. (See Figure 49, “User Parameter Selection Dialog”.)

Figure 49. User Parameter Selection Dialog



In the Create Work-Item dialog (See Figure 50, “Create or Query Work-Item”), the servers listed under the **"Known Servers"** (Window->Preferences->Variant Management), in the **"Configuration & Change Management"** category are listed in the **'Repositories'** dropdown box. All the projects in the selected repository are listed in the **'Projects'** dropdown box. Based on the selected project, the user can choose to either create a new work-item or to query an existing work-item.

Figure 50. Create or Query Work-Item



Once the user presses **OK**, the transformation will proceed and the work-item is linked to the changeset.

2.11. Rhapsody Elements Affected by the Transformation

In both transformation modes, pure::variants Restrictions in the Rhapsody model affect the same elements. The only difference is that the *Variant Project mode* removes elements, while the *Variant Enumeration mode* adds a tag to variable elements. Also please see the section called “How Rhapsody Referenced Packages or Units are Handled” for how elements in external Rhapsody packages are treated during transformation.

A Rhapsody element can be annotated with a *pvRestriction* in three different ways (see also Section 2.4, “Adding Variability to IBM Rational Rhapsody Projects”):

1. by a Rhapsody constraint with stereotype *pvRestriction*, which is connected to the element via an anchor
2. by a child Rhapsody constraint with stereotype *pvRestriction*, which is not anchored to any other element (!)
3. by a tag with name *pvRestriction*

If an element is annotated with a pvRestriction in one of these ways, the following elements will be removed or tagged during transformation:

- the annotated element itself
- all elements that are located hierarchically below the annotated element (the hierarchy is represented in the Rhapsody model browser)
- all instances of the annotated element (e.g., instances of a class or type)
- all flows connected to the annotated element
- all transitions connected to the annotated element

- if the element is a state, all connected pins and the pins' connected transitions

- if the annotated element is a requirement, all elements that are dependent on the requirement are removed/tagged. This only applies if propagation from requirements is enabled and the stereotype of the dependency is selected in the preferences (see the section called “Using Requirements”)

In contrast to pure::variants Restrictions, pure::variants Value Calculations and pure::variants Value Choices are only evaluated in *Variant Project mode*. If an element is annotated with a *pvValueCalculation* or *pvValueChoice*, only the value of the element itself is changed. See the section called “Adding Variability to Rhapsody Element Values” for more information.

How Rhapsody Referenced Packages or Units are Handled

External Rhapsody packages can be included into a Rhapsody project in different ways. Depending on how a package is included, it is handled differently during transformation. There are 3 basic ways to include a Rhapsody package:

1. as a *read-only reference* (**File**->**Add to Model...** and select radio button **As Reference**)

2. as a *read-writeable unit*, which is *copied into the model* (**File**->**Add to Model...**, select radio button **As Unit** and checkbox **Copy Into Model**)

3. as a *read-writeable unit*, which is *not copied into the model* (**File**->**Add to Model...**, select radio button **As Unit** and deselect checkbox **Copy Into Model**)

When using way (1), the referenced package is read-only. Thus, the transformation cannot make any changes to it. Any pvRestrictions or calculations affecting elements in the referenced package will be ignored. If you do want the transformation to make changes to the referenced package, consider adding a Rhapsody project element to your family model which uses an .rpyx file in which that package is part of the Rhapsody project and read-writeable.

When using way (2), the unit is part of the Rhapsody project and therefore treated just like any other package of your model. All pvRestrictions affecting elements in that package will be processed as usual.

When using way (3), the unit is read-writeable, but not part of the Rhapsody project. This leads to a more complicated behavior during transformation. Changing or removing elements from these external read-writeable units may lead to loss of input data, because the unit would be edited directly without copying it to the output folder first. To prevent this, the transformation ignores any pvRestrictions that affect elements in such units.

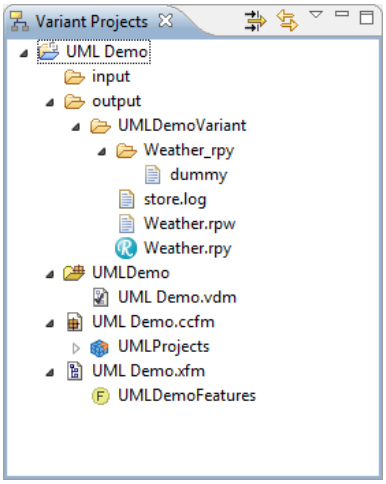
Only in cases where we can be sure that no input data is lost, pvRestrictions and calculations are processed also in external read-writeable units. These cases include transformations in variant enumeration mode (since nothing is removed in this mode) and Rhapsody Model Manager transformations (since changes are done in a new stream and thus no input data can be lost). Please note that during RMM transformations only external read-write units are processed, which are checked out in the temporary workspace/sandbox that is used during transformation.

2.12. Using the IBM Rational Rhapsody Project Variants

... With Rhapsody Project File

In the output folder (as defined in the transformation configuration, for standard projects the name is `output`) a sub folder with name of the VDM is created. Inside this folder the Rhapsody project is stored (see highlighted area in Figure 51, “Layout of a Standard pure::variants project”). A double-click will open the generated Rhapsody project file in IBM Rational Rhapsody.

Figure 51. Layout of a Standard pure::variants project



The generated project file can be used like the original master project file. Since this model is technically a copy of the original model, all element ids etc. are the same as in the master project, permitting easy compare options of generated variants with each other or with the master IBM Rational Rhapsody project.

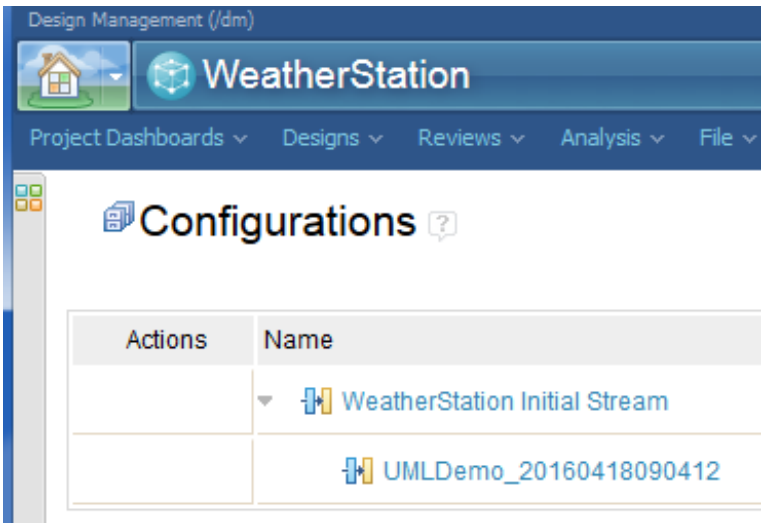
If you selected the **Enable Update Support** checkbox in the transformation configuration of your configuration space (see Figure 36, “Transformation Configuration with Rhapsody Transformation”), multiple sub folders will be created below the variant output folder. You can use the Rhapsody project files in these folders to merge custom changes made in a variant with a newly transformed version of that variant. For details, see section *Variant Update* in the pure::variants User's Guide.

Since per default a three-way merge of Rhapsody project files is not supported in Eclipse, you may need to execute some merge steps manually or by using an external merge tool, like *IBM Rational Rhapsody DiffMerge*.

... With Rhapsody Server Project

The output folder remains empty. Instead the project in the Design Management server has added a new variant stream. This can be opened with the Rhapsody client installation or via web-browser.

Figure 52. Output of a new variant stream



Regardless of having **Enable Update Support** activated (see Section 2.7, “Adding a Rhapsody Transformation to pure::variants Projects for Rhapsody”[32]) the output is always updatable. The produced output is different to the described updatable output in section *Variant Update* in *pure::variants User's Guide*. In Rhapsody Design

Manager a new variant stream version is created for transforming the same variant. Thus the changes made in the previous variant stream can be merged to the newly transformed version of that variant.

In case of transforming a Rhapsody project referenced in a Global Stream, the stream is created as described previously. Having other transformation modules, which also support Global Streams, included in the same transformation configuration, only one global variant stream is generated. This requires that all variability-aware models, participating in the transformation process, originiate from the same global stream.

Since per default a three-way merge of Rhapsody server projects is not supported in Eclipse, you may need to execute some merge steps manually or by using an external merge tool, like *IBM Rational Rhapsody DiffMerge*.

3. Troubleshoot

3.1. 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**.

4. Known Issues

- IBM Rational Rhapsody cannot open project files with an uppercase file extension. So the transformation cannot handle them either.
- RMM transformation opens a RTC application instance. This must be closed manually by the user after the transformation process has finished. When closing the RTC application an error dialog opens. This dialog can be ignored and closed by pressing OK.
- If Rhapsody packages are shown unloaded in project, please ensure to add your Rhapsody packages and profiles with proper path. If Rhapsody profile is managed via EWM's versioning system, then ensure to define the load rule in your Family Model, as well, if you use load rules in general. If Rhapsody profile is provided by Rhapsody installation (or located elsewhere), please ensure to define a proper relative path (via Edit Unit), using $OMROOT variable, e.g. for SysML profile: $OMROOT\Profiles\SysML\SysMLProfile_rpy

- The pure::variants menu entries may be grayed out. This happens when the maximum helper count is reached in Rhapsody. This means that only a limited number of helpers (specified in .hep files or in rhapsody.ini) can be loaded. Please remove some helpers from your current project and try again.
- RMM transformation fails when using rhapsody 9.0 or above, if the VariantManagement profile is added "As Unit" into the model with "Copy into model" option checked.
- After transformation, the diagrams checked-in into the variant stream in the RMM server are updated only in online mode.
- Adding constraints on orthogonal states is not allowed to remain consistent with the behavior of Rhapsody. It is currently not possible to delete or perform visualizations on orthogonal states using the Rhapsody API.