pure::variants - Connector for Jama Manual

Parametric Technology GmbH

Version 7.0.0.685 for pure::variants 7.0

2025

Table of Contents

1. Introduction

1.1. Software Requirements

The following software has to be present on the user's machine in order to support the pure::variants Connector for Jama:

Jama connect: Jama Connect versions 8.25 to 9.18.0 are supported. Compatibility with other releases is not guaranteed.

The pure::variants Connector for Jama is an extension for pure::variants and is available on all supported platforms.

1.2. 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**).

1.3. 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 the Connector

2.1. Starting pure::variants

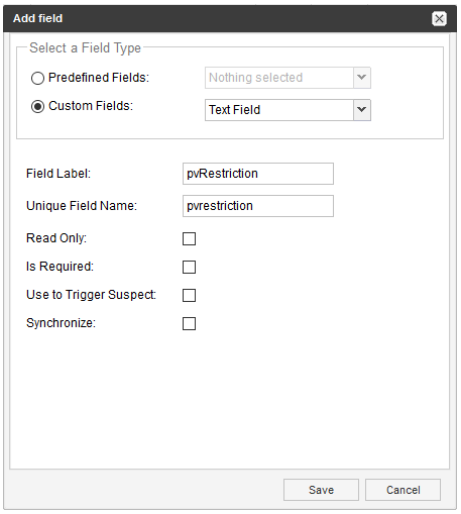
Depending on the installation method used either start the pure::variants-enabled Eclipse or under Windows select the **pure::variants** item from the **program** menu.

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

2.2. Preparing the Jama project

In order to get the variability information from Jama items as well as to create variants in Jama, the projects need to be prepared initially. To make the variability information available to pure::variants, an attribute has to be set for every Jama item that shall be processed with respect to its variability. To set this attribute in Jama, go to the administration page (ADMIN tab at the top of the page), select **Item Types** and select the **Fields** link for each item type, that shall be processed by pure::variants. Press **Add field** in the upper right corner and enter the information as shown below.

Figure 1. Add a new field to an item type



After this was done for every item, pure::variants will use the information stored in this field to create Restrictions on the imported items automatically.

For the creation of items included in a certain variant, pure::variants needs some preconditions to be met, depending on the mode used in the transformation (see also Section 2.5, “Transforming a Variant” for details on the transformations). To add tags to an item, there is nothing to be done. However if a pruned copy of the master project shall be created using the copy mode, a project with the same name has to exist on the server prior to pure::variants transformation start. The location of the project needs to be given in a transformation parameter. See Section 2.5, “Transforming a Variant” for information on how to give this location as well.

With linking mode, pure::variants will create an item at a given location, that has a link of a given type. For this to work, pure::variants assumes, the target project and the set that holds the created items to exist. The location of said set, as well as the name of the link type needs to be given in a transformation parameter. See Section 2.5, “Transforming a Variant” for information on how to give those.

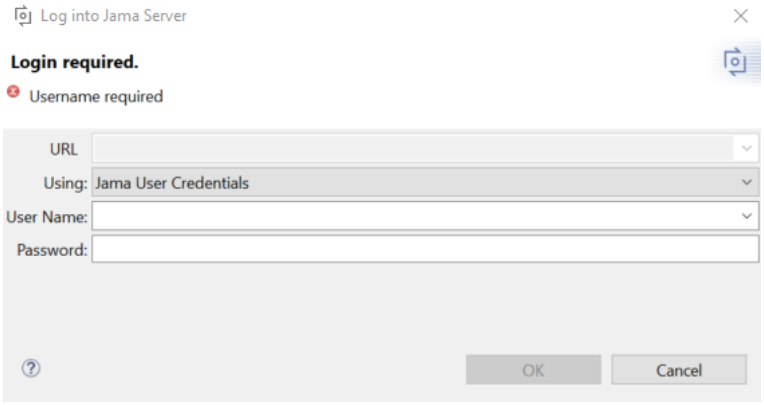
2.3. Creating the Initial Model(s)

The first step is always to create corresponding family models for each relevant Jama project. These initial family models serve as starting points for using existing variability information. The import procedure has to be executed **only once** for each Jama project. Each project is represented by one pure::variants model.

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

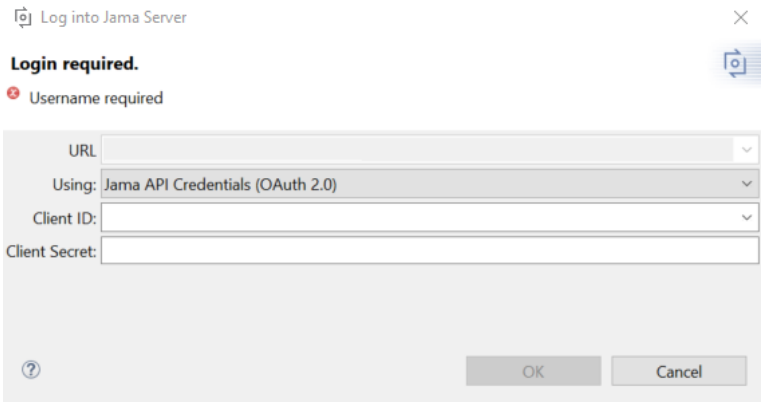
The import wizard opens. On the first page the Jama server connection is specified. To log into the Jama server, multiple possibilities are provided. Firstly the Jama user name and password can be provided with the Jama User Credentials login.

Figure 2. Connecting to the Jama server using user credentials



For Jama servers that provide Jama API Credentials (OAuth2 authentication), the ClientID and Client Secret have to be provided in a similar matter. To get more information about Jama's configuration of OAuth2, please visit: https://dev.jamasoftware.com/api/#auth.

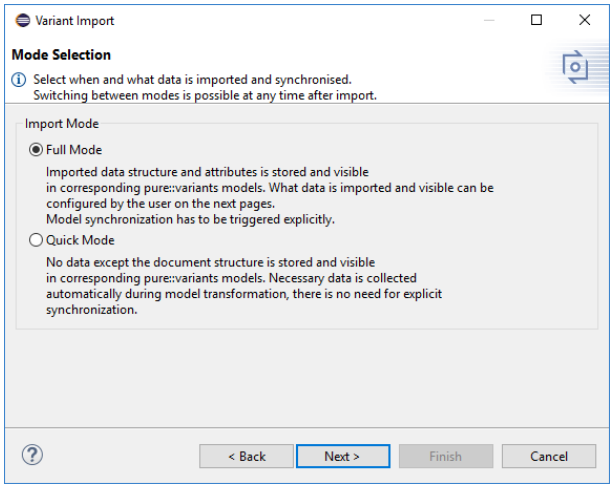
Figure 3. Connecting to the Jama server using Jama API Credentials (OAuth2)



Independent of the authentication method used, the environment variables *"PV_JAMA_PASSWORD"* and *"PV_JAMA_USER"* can be set with user credentials or oauth id and secret for automation purposes or ease of use. Furthermore *"PV_JAMA_AUTHMETHOD"* will specify if the given credentials are used for "BASIC" (default, if not provided), or "OAUTH" authentication.

On the second page you can decide whether you want to perform a full import of your Jama Projects (**Full Mode**), or just to import the project's connection information (**Quick Mode**). In the latter case the data is automatically synchronized before a transformation, whereas in full mode, the user is responsible to keep the data synchronized, as the existing data is used to transform the variants.

Figure 4. The Import Mode Selection Page in the Jama Import Wizard



By using **Full Mode** all relevant data of the selected Jama projects can be imported. The imported data is stored and visible in corresponding pure::variants models. If using this mode, the models need to be explicitly synchronized before transforming a variant. Usually the **Quick Mode** should be used. The Quick Mode is used to just import information pure::variants needs to access the Jama projects during synchronization and transformation. The result of this mode are 'empty' family models. Each module is represented by a pure::variants model containing the link information to the related Jama projects. In **Quick Mode** there is no need to explicitly synchronize the models imported from Jama, this is done automatically during transformation.

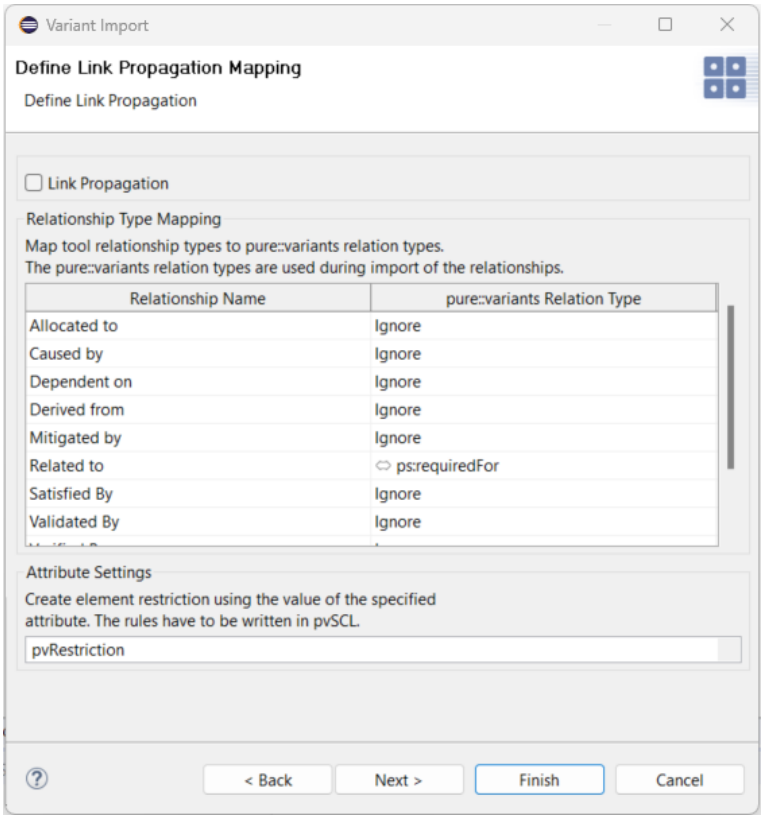
It is possible to switch between modes by performing a synchronization and changing the mode in the occurring wizard.

On the next page the complete project and folder structure of the Jama server is shown. Navigate to the folders containing the projects of interest and select the check boxes on the right side representing the Jama modules. Selecting a check box on the left side marks all modules inside this folder and its sub-folders for import. Make sure that the import target location given next to **Import into:** is correct. The location can be changed using the **Choose** button.

If **Store created models according to folder structure** is checked, the folder structure created in pure::variants will resemble the Jama folder structure. Projects are treated as normal folders in this case. If unchecked, all projects are stored directly under the selected target location. *Use this only when all selected modules have unique names.*

On the next page you can adjust several options for your project import.

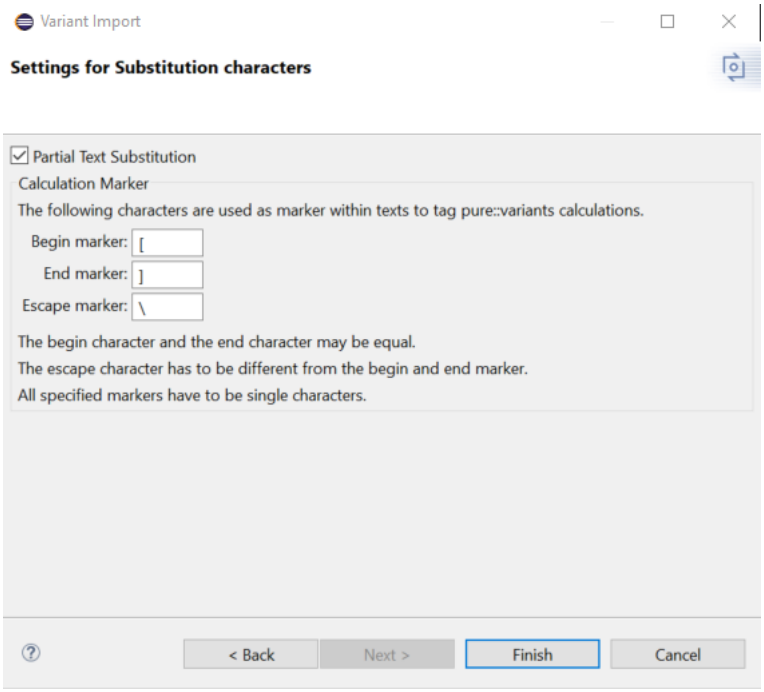
Figure 5. The Module Selection Page in the Jama Import Wizard



The link propagation allows to evaluate the dependency relations between Jama items with the usage of pure::variants relation types. If link propagation is disabled, the relationships in the Jama project will not be fetched. The table allows you to specify which Jama relationship type is treated as an pure::variant relation and which pure::variants relation type will be used. The combo box below the table allows to define the attribute, which will be used for creating restrictions in the pure::variants models. The default attribute is **pvRestriction** .

On the next page you can decide whether or not to import calculations and partial text substitutions. If selected, the text of an item can be adapted in the variants depending on parameters in e.g. the feature model.

Figure 6. The Substitution Page in the Import Wizard

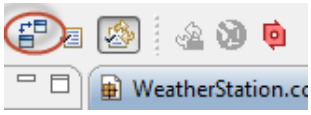


If **Partial Text Substitution** is checked, all marked calculations in Text typed properties of an item (e.g. titles and descriptions), will be checked for occurrences of pvSCL references. Those pvSCL expressions in the text need to be marked, using the characters specified on the page and can be escaped by the set escape character where needed.

2.4. Updating Models from Jama

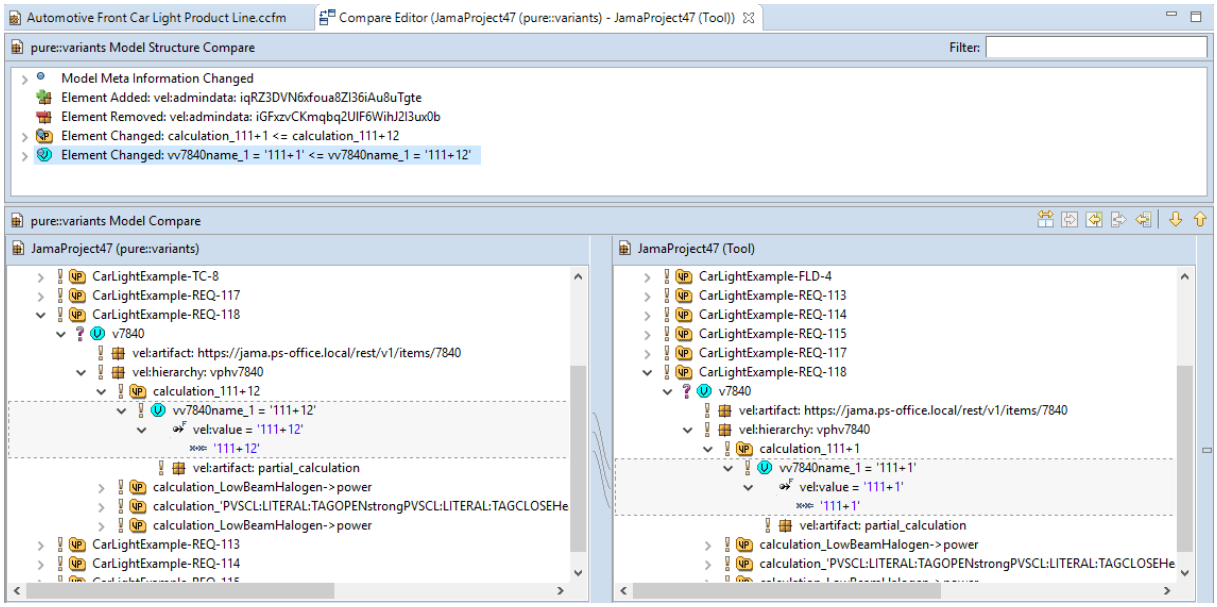
If the **Full Mode** is used, it is necessary to update the pure::variants models with information from Jama whenever relevant changes have been made. To facilitate the synchronization, pure::variants provides a **Synchronize** action. To start the update, open the model representing the Jama project and press the **Synchronize** button in the tool bar.

Figure 7. Synchronize model



pure::variants will connect to Jama and present the so called Compare Editor for pure::variants models.

Figure 8. Synchronize model compare



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

The Merge toolbar provides tools to copy single or even all (non-conflicting) changes as a whole from the current model to the out-dated model.

2.5. Transforming a Variant

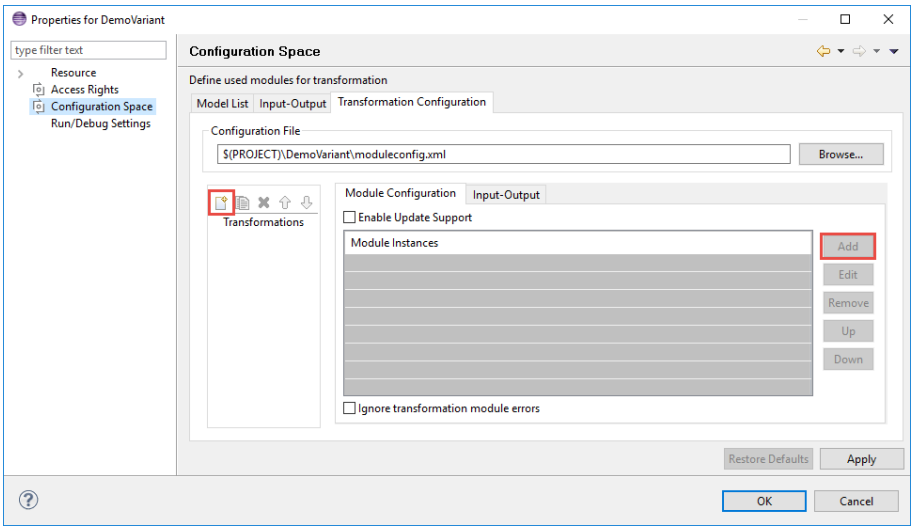
Variants stored in a variant description model can be made available in Jama. There are three ways to created Jama projects from pure::variants, based on the variant information. If a Jama item is part of a variant, the item can be tagged with the name of the variant, a copy of the item can be stored in a dedicated area on the Jama server, or a link can be created on a created item within a provided set of items.

To transform a variant, first a *Transformation Configuration* has to be created. To create a Transformation Configuration click on the arrow next to the **Transformation** button in the tool bar and choose *Open Transformation Config Dialog...*

Figure 9. Transform Model

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

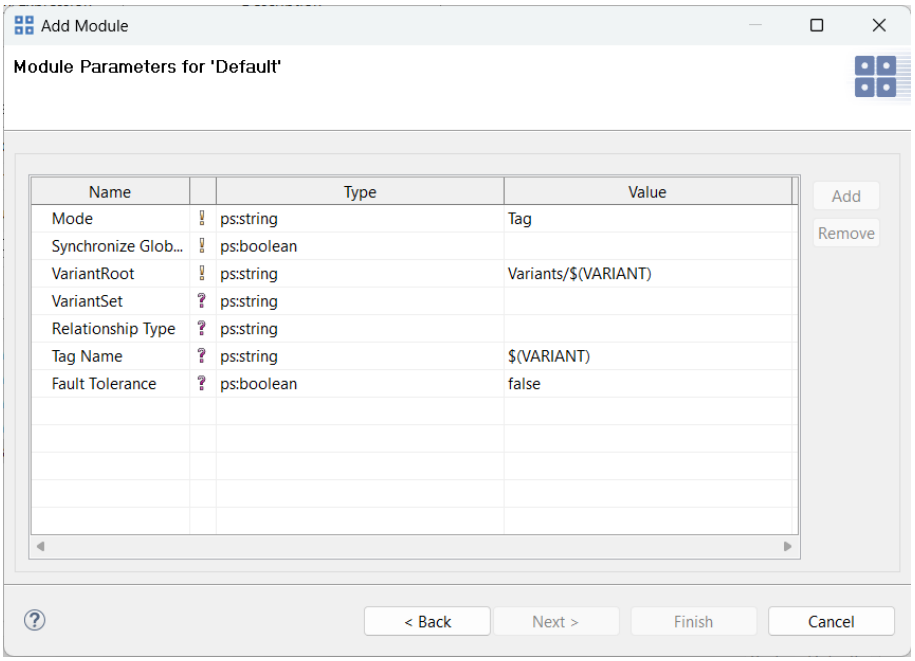
Figure 10. Transformation Configuration



A new Dialog comes up, choose **Jama Transformation Module** and enter a name if needed.

The next page shows some parameters:

Figure 11. Transformation Configuration



- Mode - Define the transformation mode. You can choose between the three modes: 'Tag', 'Copy', and 'Link'
- Synchronize Items - if the mode is set to 'Copy', setting this parameter to true results in the copied items sharing the same global ID as the source they were copied from. This is used for traceability in Jama. This will be ignored if "Enable Update Support" is enabled.
- VariantRoot - If you choose the 'Copy' mode, the path to the pre-existing project has to be defined, where a copy of all modules and items will be created. The default is 'Variants/[name-of-the-variant-model]'
- VariantSet - If you choose the 'Link' mode, the path to the pre-existing set of items has to be defined. Here the Object ID of the target set (ProjectKey-ItemKey-ID, e.g. 'VLP-VARS-1') has to be given.

- Relationship Type - If you choose the 'Link' mode, the type to the link that will be created has to be defined. Here the name of the Jama relationship type (e.g. 'Allocated to') has to be given.
- Tag Name - Specify the name of the generated tag. The default value is $(VARIANT). This setting is only valid for 'Tag' mode.
- Fault Tolerance - If set to true, this will skip, but log, read/sync failures of items. The transformation will succeed with warnings, if these failures were encountered.

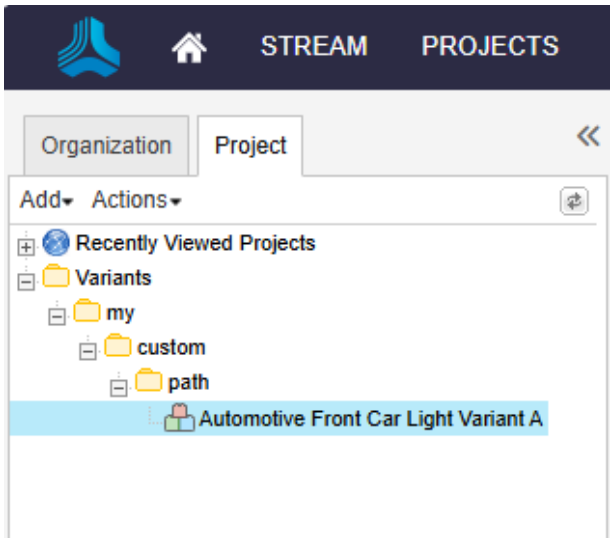
If it is not sufficient to transform all modules to the same location defined in the "VariantRoot" parameter, customized module paths can be used. To specify the path, where each module will be copied to (note: applicable for copy transformation only) an attribute can be manually added to the root element of the corresponding family model. The attributes name has to be *jama:outputPath* . The entered path will be appended to the folder defined in the "VariantRoot" parameter, must not start with a "/", and needs to contain the module name as last segment of the path. If the module shall be renamed during transformation, just specify the new module name instead of the old one. Note that the renamed target project needs to exist before the transformation is started and that the path will not be checked for syntactical errors.

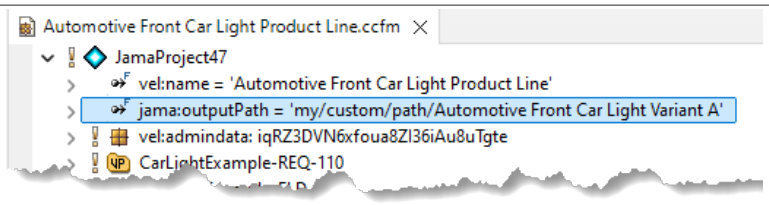
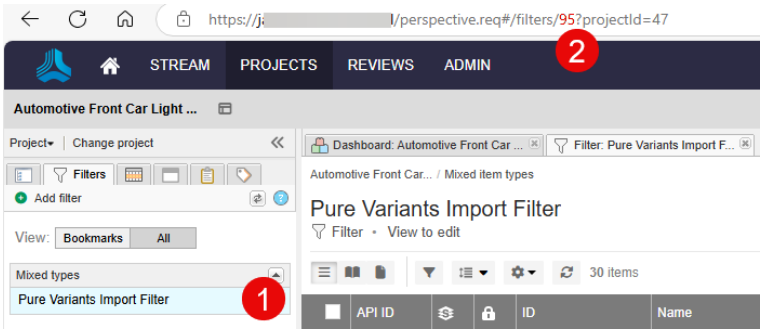
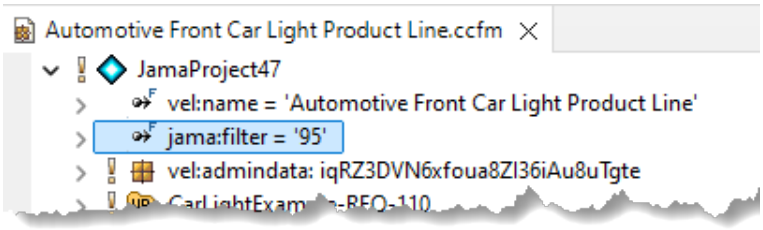
After completing the dialogs, the transformation can simply be used by clicking on the Transformation button (see Figure 9, “Transform Model”) in the tool bar and choosing the transformation in the drop-down-menu.

2.6. Family model attributes to customize transformation behaviour

The following attributes can be added to the root element of a family model, which represents a Jama project. Adding those attributes changes the transformation behavior and thus can be used to customize the Variant Representation. All attributes are optional and can be freely combined.

Table 1. Transformation customization attributes

Attribute Name	Behaviour
jama:outputPath	Defines the target path of the variant project. The path is relative to the *VariantRoot* defined in the transformation configuration. The entered path needs to contain the final variant project name. So the resulting path is: <VariantRoot>/<content of the attribute> Example Jama Output Path:  Example "jama:outputPath" in family model:

Attribute Name	Behaviour
	 Assuming *VariantRoot* transform module parameter is defined with "Variants" value, the project "Automotive Front Car Light Product Line" will be copied to the variant project "*Variants/my/custom/path/Automotive Front Car Light Variant A*".
jama:filter	This attribute is used to refer to a Jama filter by its identifier. The Jama filter has to exist in the project of the transformed Jama project. If the filter is defined, all items not visible in the filter are ignored in the processing of the Jama project. Example Jama Filter:  Example "jama:filter" in family model:  In the example, the Jama project filter with identifier "95" is used while importing/transforming the Jama project "Automotive Front Car Light Product Line".

3. Advanced Topics

3.1. Variability in HTML tables

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

Figure 12. Example HTML table

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

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

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

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

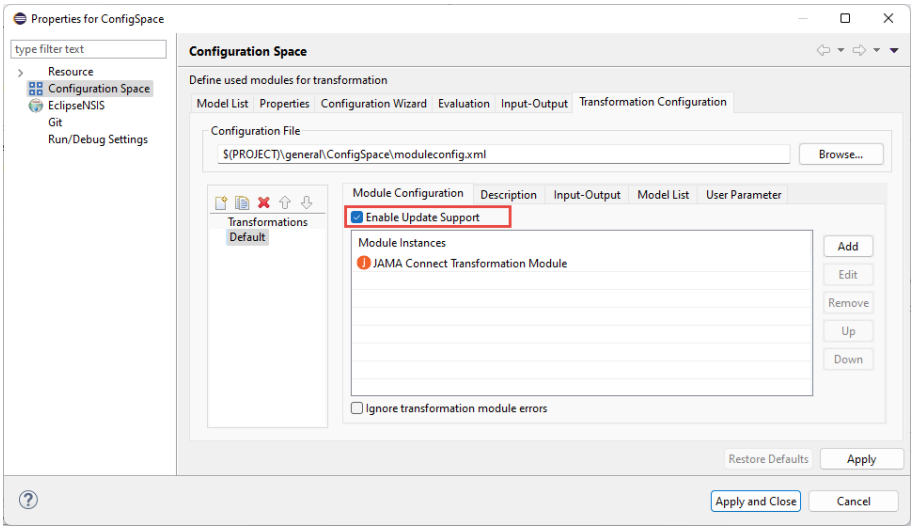
3.2. Jama Transformation with Update Support

With update support, the update of previously transformed variant is supported, i.e., when a copy transformation is triggered, instead of items being deleted and recreated, they are updated. This makes sure that the Global IDs and item IDs remain the same. If the "Enable Update Support" is activated, the "Synchronize GlobalIDs" option is activated implicitly.

Please note that the first transformation will always create new artifacts even if "Enable Update Support" is activated. Each subsequent transformation, will relate existing assets, by using the asset's Global-ID, and only update the fields.

The "Enable Update Support" option can be enabled as shown in the figure Figure 13, “Enable Update Support”.

Figure 13. Enable Update Support



See 'pure::variants User's Guide', Chapter 'Setting up a Transformation' for more information on how to enable update support.

4. Troubleshoot

4.1. Connection Issues - Timeouts, Interrupts, etc.

Since pure::variants Connector are heavily relying on integration to Jama 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:

- 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:

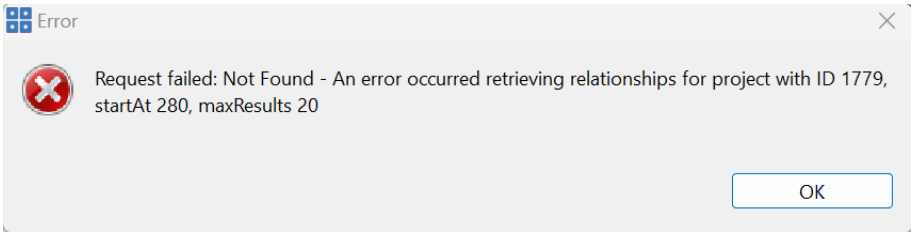
```
...
-vmargs
-DPV_HTTP_CLIENT_CONNECTION_TIMEOUT=240000
-DPV_HTTP_CLIENT_READ_TIMEOUT=240000
...
```

Note

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

4.2. Import/Transformation might fail if too many relationships exists

Since pure::variants Connector is importing all relationships, which exists between Jama's items, the fetching of these relationships may raise a strange error.



If too many relationships exists and Jama fails to return thos, you may overcome this issue, by defining the following parameter:

- Add retry strategy timeout (in milliseconds, default: 0): *`JAMA_RETRY_ON_404=120000`* (equal to 2 minutes)

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

```
...
-vmargs
-DJAMA_RETRY_ON_404=120000
...
```

Note

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

Note

This issue is reported to Jama: https://support.jamasoftware.com/hc/requests/74250.