pure::variants - Connector for Azure DevOps Manual

Parametric Technology GmbH

Version 7.0.0.685 for pure::variants 7.0

2025

Table of Contents

1. Introduction

pure::variants - Connector for Azure DevOps enables Azure DevOps users to manage work items variability using pure::variants. By coupling pure::variants and Azure DevOps, knowledge about variability and variants can be formalized, shared and automatically evaluated. This enables getting answers for questions about valid combinations of work items in product variants quickly; permits easy monitoring of planned and released product variants at the work item level and also permits very efficient production of variant-specific work item queries out of the work item repository.

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

1.1. About this manual

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

1.2. Software Requirements

...for pure::variants - Connector for Azure DevOps

The following software is supported by the pure::variants Connector for Team Foundation Server:

Azure DevOps Server:	Azure Devops Server 2020. Compatibility with other Azure DevOps Server releases is not guaranteed.

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

Oracle Java Runtime:	At least Java Runtime Environment 8 is required.
Web-Browser Compatibility:	Microsoft Internet Explorer 11
	Mozilla Firefox 49
	Google Chrome 53

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

Team Foundation Server and pure::variants communicate using the HTTP(S) protocol.

...for pure::variants Web Integration

The following software is required to use pure::variants Web Integration:

pure::variants Desktop Hub	The pure::variants Desktop Hub is delivered with the pure::variants Enterprise windows installer package and can be installed by selecting the **Integration Components** in installer wizard.

1.3. Installation

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

2. Using the Connector

2.1. Starting pure::variants

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

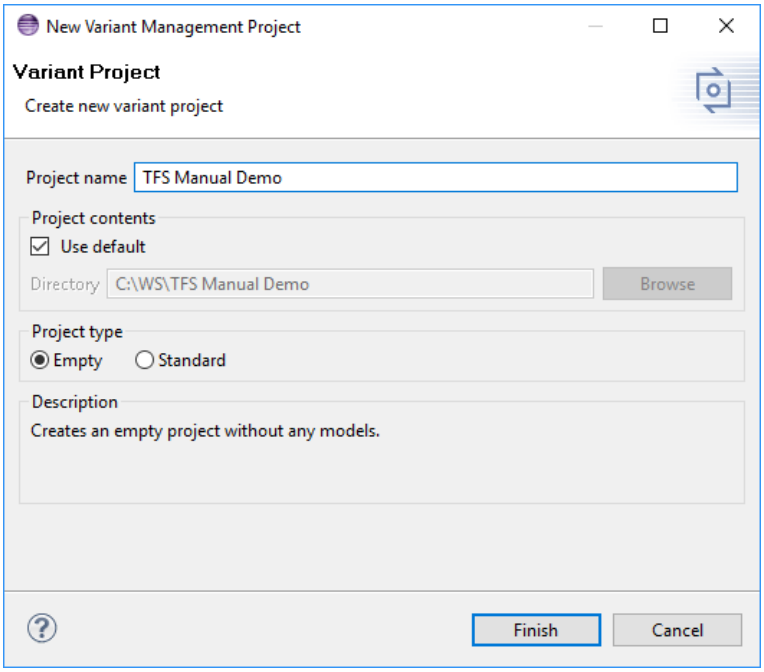
If the **Variant Management** perspective is not already activated, do so by selecting it from **Open Perspective -> Other** in the **Window** menu.

2.2. Creating the Initial Model(s)

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

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

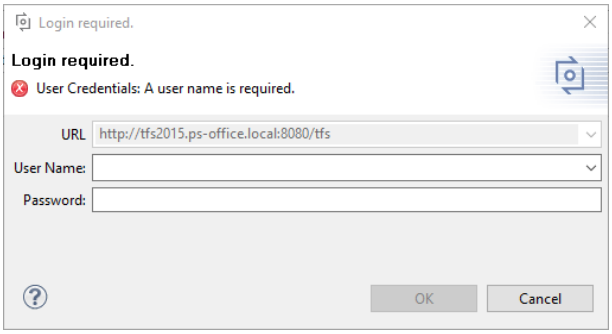
Figure 1. Creating an empty Variant Management project for Azure DevOps query import



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

The import wizard appears. A login dialog pops up if you are not authenticated to Azure DevOps server yet. The dialog needs the address of Azure DevOps server and your credentials, username and password, as input. You authenticate to server by pressing "OK" and you see the first page of wizard.

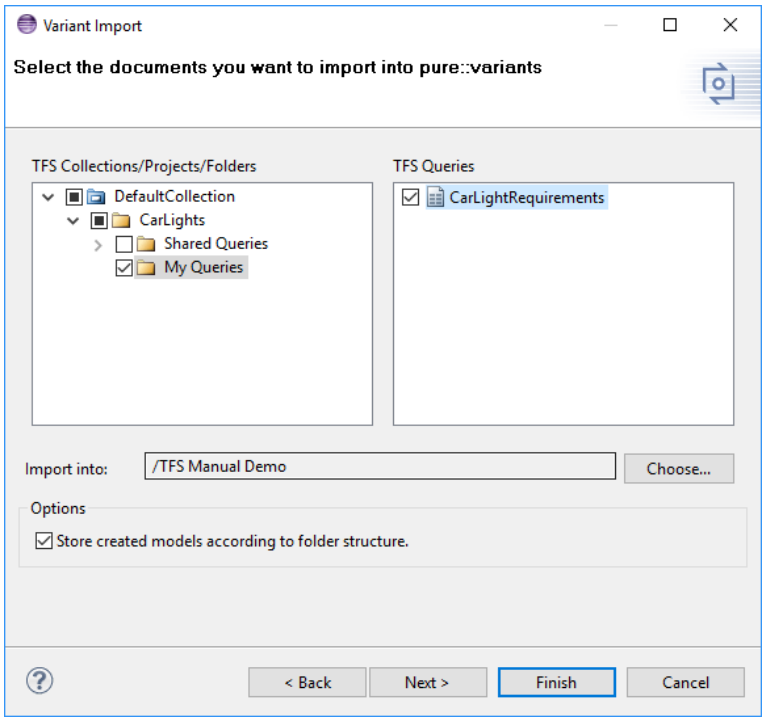
Figure 2. Login to Azure DevOps server



The complete project and folder structure of the Azure DevOps repository is shown. Navigate to the folders containing the queries of interest and select the check boxes on the right side. Selecting a check box on the left side marks all queries 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.

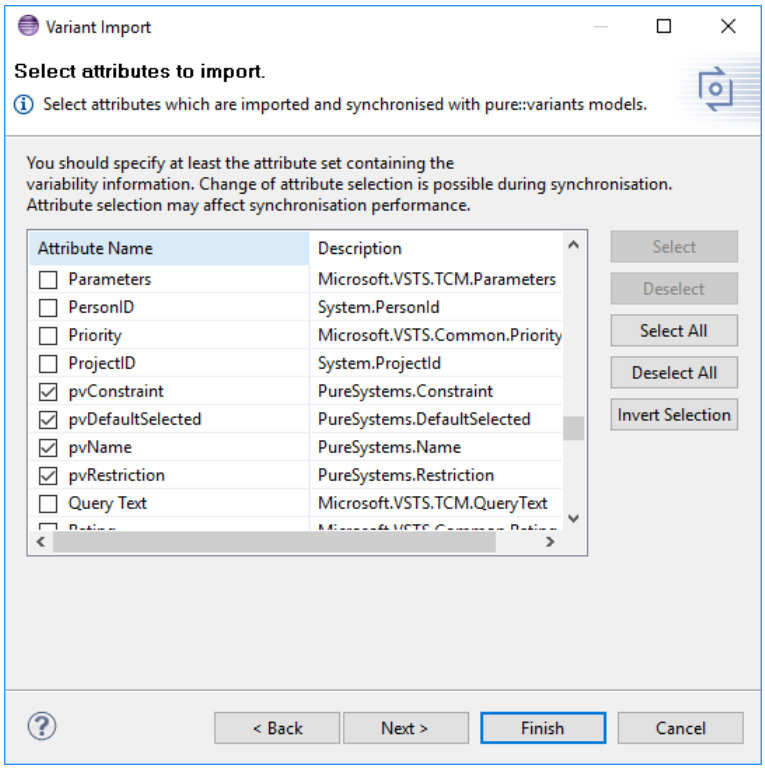
If **Store created models according to folder structure** is checked, the folder structure created in pure::variants will resemble the Azure DevOps folder structure. Project Collections and Projects are treated as normal folders in this case. If unchecked, all queries are stored directly in the selected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

Figure 3. The query selection page in the Azure DevOps import wizard



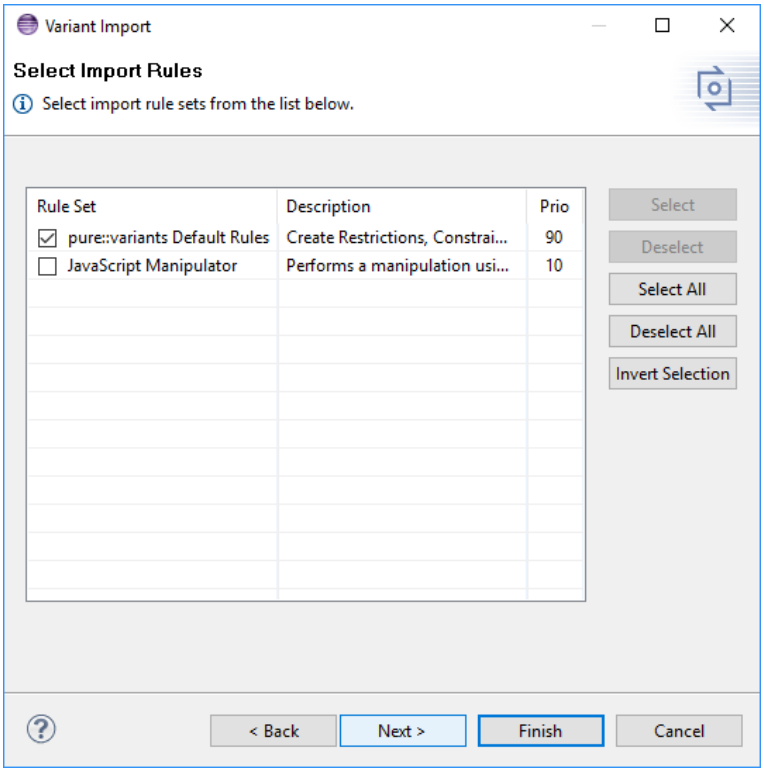
At the following page a list of Azure DevOps attributes is shown. The attributes can easily be selected. Checked attributes will be imported, unchecked ignored.

Figure 4. Set of Azure DevOps attributes to import



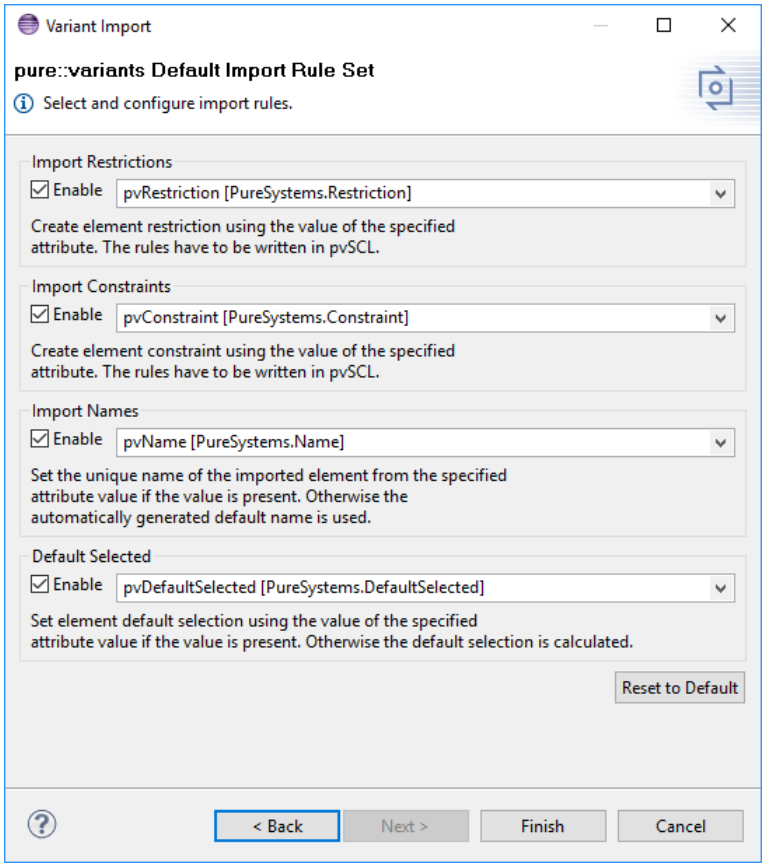
Pressing **Next** button brings the Import Rules page up. On this page you can select sets of Import Rules, which will be used to manipulate the resulting model after import. Import Rule Sets can be used to create specific pure::variants model elements like restrictions or constraints from Azure DevOps query information.

Figure 5. Select Import Rules to use during import



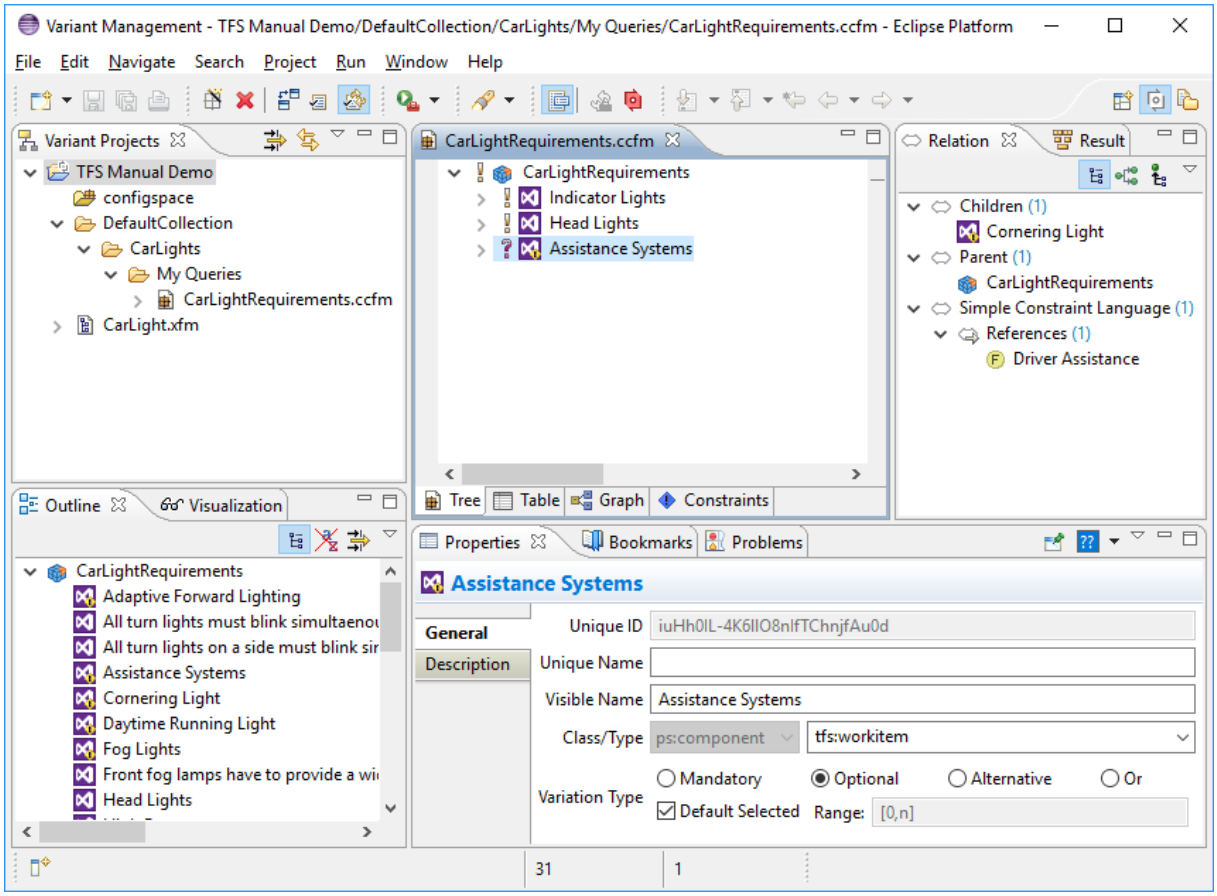
The last pages are showing the settings of the selected Import Rule Sets. For the **pure::variants Default Rules** you can choose which attribute value will be used for creating restrictions and constraints and setting the default selection on elements and which attribute value will be used as unique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

Figure 6. Settings for the pure::variants Default Import Rule Set



The import result will be visible in the Variant Project view. If nothing shows up, use the item **Refresh** in the project's context menu (right mouse click) or press **F5** after selecting the project in the view. Each query is now represented by one pure::variants model. Models can be opened by double-clicking on them or selecting **Open** in the context menu. Figure 7, “Result of initial import of a Azure DevOps query” shows the typical layout of a Azure DevOps query after import.

Figure 7. Result of initial import of a Azure DevOps query



The Azure DevOps query is imported as family model and their work items are represented as components. The elements follow the hierarchical structure as found in the original Azure DevOps query, if the query is configured as such. Otherwise the elements are presented as list.

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

All work item elements are imported as *mandatory* (if a restriction was defined in Azure DevOps) unless variability information was provided in the Azure DevOps queries. This is explained in more detail in Section 4.1, “Adding Variability Information in Azure DevOps”.

Table 1. Overview of representation of Azure DevOps entities in pure::variants

Azure DevOps Entity	**pure::variants Representation**
Project Collection	folder in project
Project	folder in project
Query Folder	folder in project
Query	family model
Work Item	component in family model
Work Item title	elements visible name (first non-empty value is used, long names are shortened)
Work item attribute `pvRestriction`	element restriction in pvSCL language
Work Item attribute `pvConstraint`	element constraint in pvSCL language

Work Item attribute `pvVariationType`	element variation type. Valid input values are either `mandatory`, `or`, `optional`, `alternative`, `ps:mandatory`, `ps:or`, `ps:optional`, or `ps:alternative`.
work item attribute `pvName`	element unique name. pure::variants will use the defined name as unique name. The name should be a valid unique name (see pure::variants User Guide for more information). If the name is not valid, pure::variants will generate a valid name from it. Uniqueness is not enforced during import, but later shown as model problem in the model editors.
Work Item attribute `pvDefaultSelected`	element default selection state. Valid input values are either `off` or `on`. Case is irrelevant, so `OFF` or `Off` are also valid input values.
Work Item id	element attribute `identifier`
Work Item attribute	element attributes with type ps:integer or ps:string

2.3. Using Variability Information from Azure DevOps queries

During the synchronization, pure::variants can optionally use some information present in the Azure DevOps queries to create the pure::variants model representation. During synchronization all information including element hierarchy, restrictions and constraints are compared and, if different from the information stored in the pure::variants model, shown as mergable difference.

Which and how variability information can be represented in Azure DevOps queries is explained in more detail in Section 4.1, “Adding Variability Information in Azure DevOps”.

2.4. Adding Variability Information

The pure::variants models created from a Azure DevOps query can be changed to represent the necessary variability information. Every aspect of the pure::variants element representing a Azure DevOps object including its position in the model can be changed. Removal or modification of attribute values imported from Azure DevOps is also possible. However, these changes will not affect the original Azure DevOps query. During update the user is given the option to restore the original attribute (value) as read from the Azure DevOps query. Removal of work item elements is also possible, but again, these changes will be shown during update from Azure DevOps. It is usually recommended to keep all the work items and attributes that have been imported from Azure DevOps.

The work items hierarchy may be changed arbitrarily. It is also possible to add new elements to the model, e.g. to group a certain set of work items as an *optional* entity, a new optional pure::variants element can be created and all work item elements which will be members of this group would be moved below this new element.

Restrictions are the most flexible way to express variability rules since they can refer to attribute values and also contain arbitrary calculations. See the pure::variants User Manual for more information on the available relations and the restrictions language. **Tip:** In most cases a good hierarchical structure reduces the need for many restrictions.

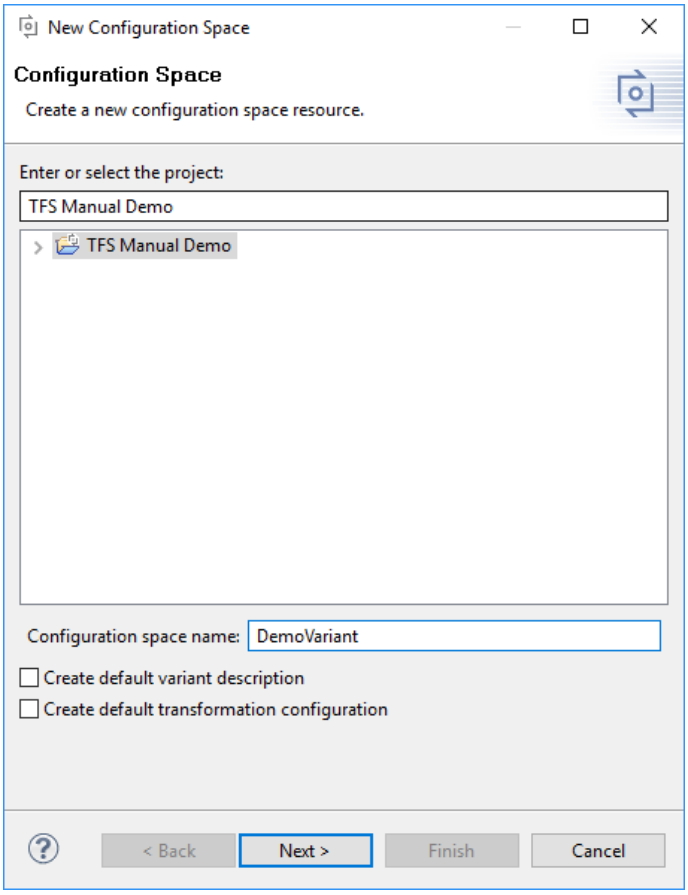
2.5. Defining a Variant

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

Variants are stored as separate entities called *Variant Description Models* (VDM). A VDM always belongs to a specific *Configuration Space*. Thus before defining variants, a configuration space has to be created. Select the project containing the imported models in the Variant Projects view and open the context menu. Below the item **New** select **Configuration Space**. A wizard is opened. On the first page (Figure 8, “The Configuration Space Wizard, page 1”), enter a name for the configuration space. The name has to follow strict rules (no spaces, no

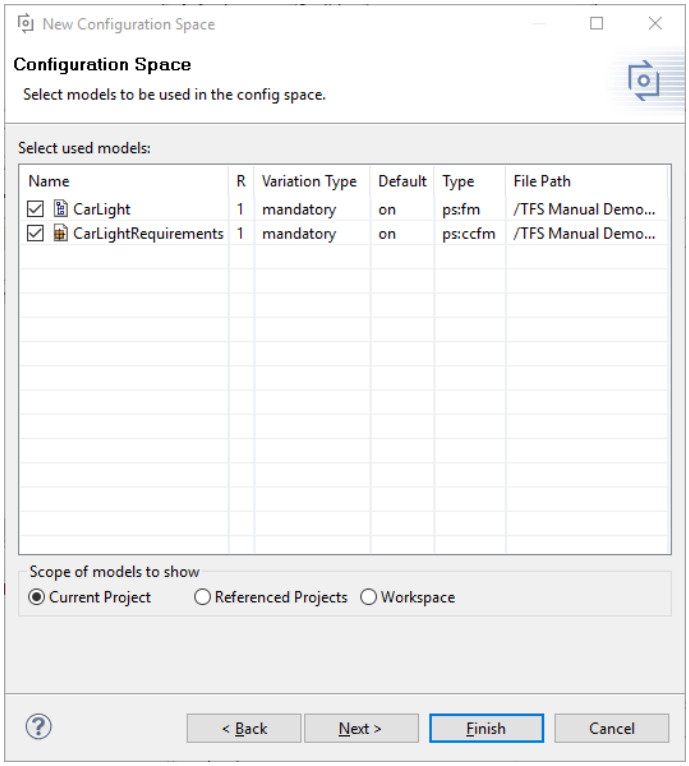
special characters). Uncheck the box before **Create standard transformation**, since for pure work items models the standard transformation does not provide any relevant functionality (See the pure::variants User Manual for more information on transformations).

Figure 8. The Configuration Space Wizard, page 1



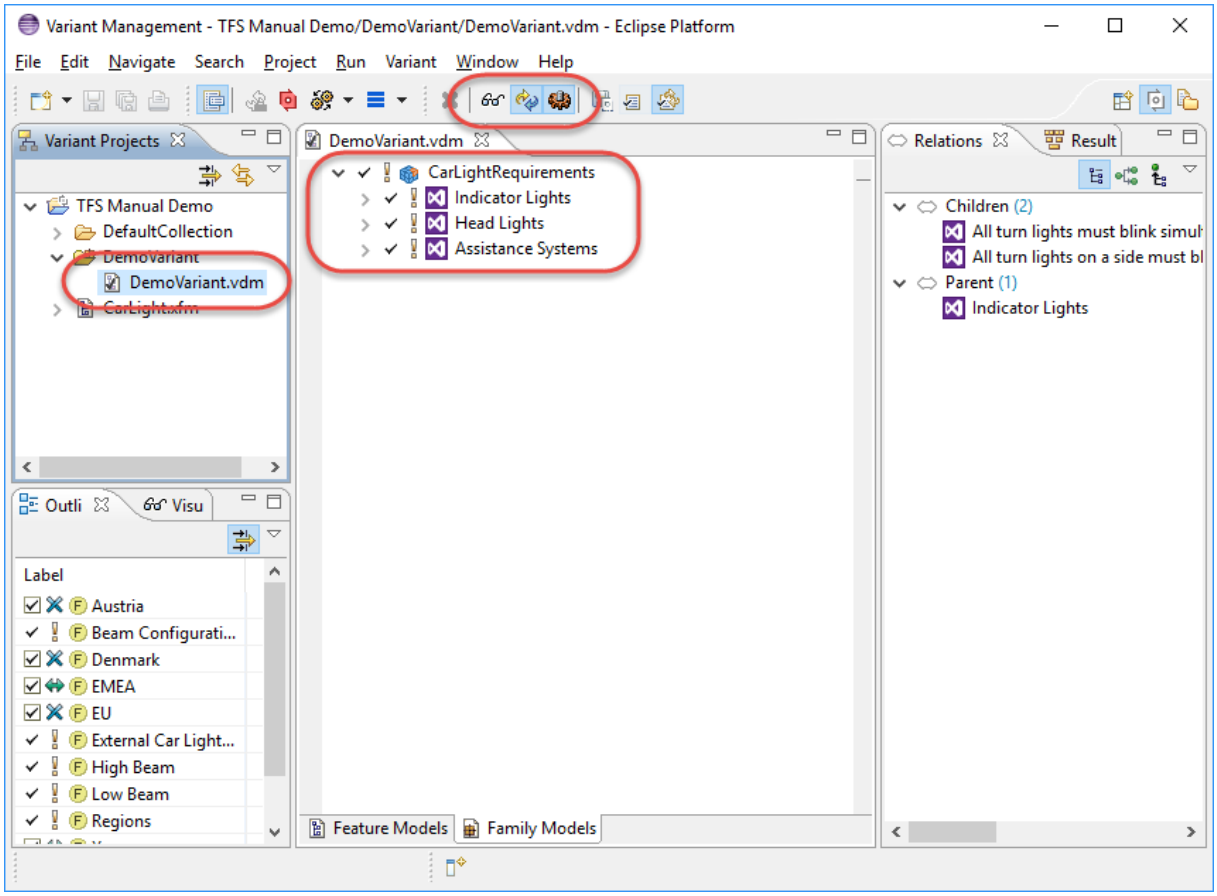
The next page is used to specify which models are to be included in this configuration space. Select here all models that represent the Azure DevOps queries of interest. In the example below just one model is selected. Now press the **Finish** button.

Figure 9. The Configuration Space Wizard Model Selection Page



The resulting project structure is shown in Figure 10, “Initial Configuration Space Structure”. The `DemoVariants.vdm` is created and immediately opened. It resembles the structure of the previously defined model(s), but has a checkbox in front of each element to permit the user to select elements for this variant by clicking on it. The buttons marked in the toolbar control the evaluation of configurations. The left-most button ( ) initiates a manual check of the variant configuration. The middle button ( ) toggles between manually checking and automatic checking after changes to the VDM. Finally the right-most button ( ) toggles the auto-resolver on or off. The auto-resolver provides automatic resolution of configuration problems where possible.

Figure 10. Initial Configuration Space Structure



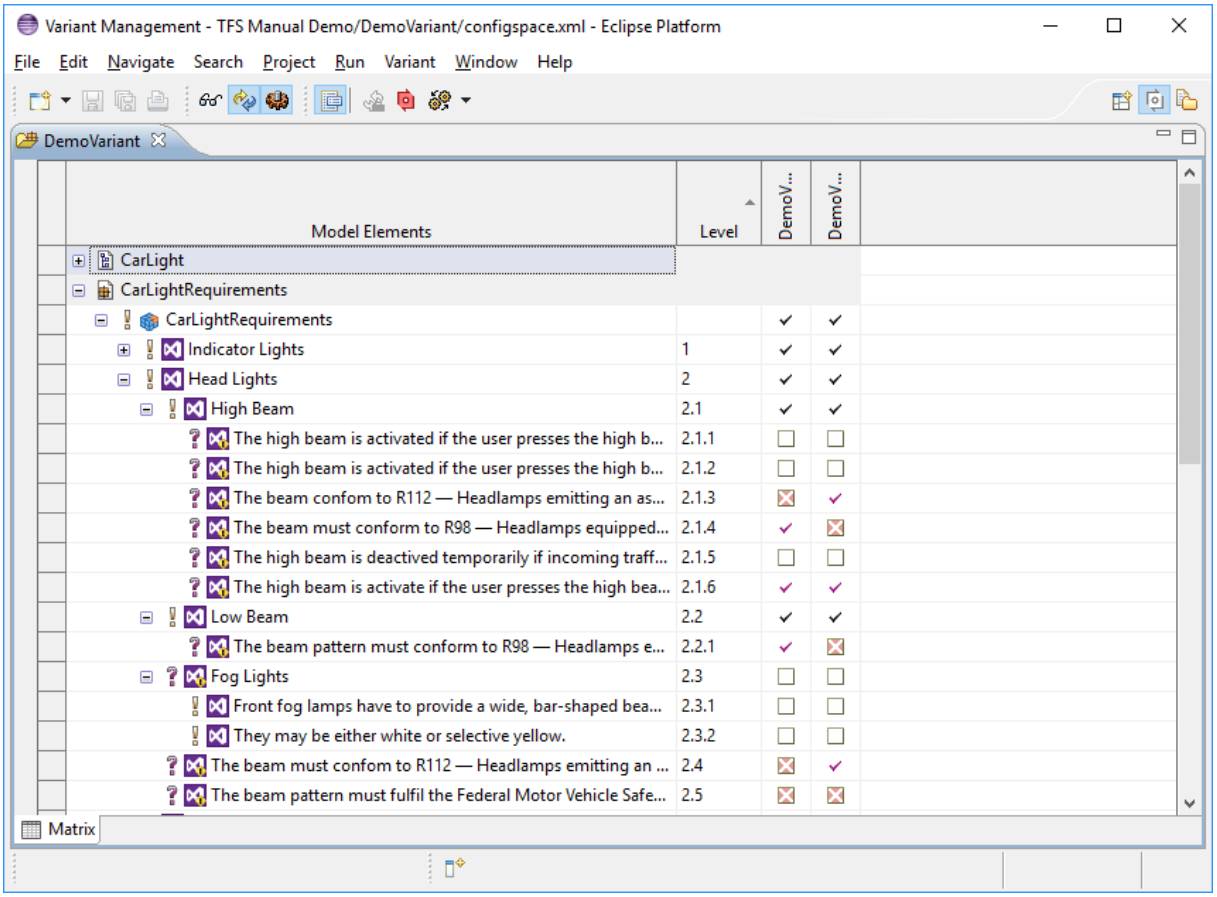
Tip: For small to medium sized models (up to several thousand elements, depending on the speed of the processor), it is convenient to turn on both auto-checking and auto-resolving by clicking on the respective toolbar buttons.

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

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

Variants may be compared at the element level by using the matrix editor (see Figure 11, “Matrix Editor for comparing variants”). This editor is activated by double-clicking on the enclosing configuration space icon. The **Table Layout** item in the context menu can be used to customize the list of variants to compare and the **Show Elements...** and **Filter** items in the same menu can be used to select elements of interest.

Figure 11. Matrix Editor for comparing variants



2.6. Transforming a Variant

Variants stored in a variant description model can be made available in Azure DevOps. The Connector supports multiple ways of representing variants: *tag-based* and *query-based*.

Tag-Based Variant Representation

In the tag-based representation we define a tag for Azure DevOps work items which are part of the variant. This transformation mode adds a tag with the name of the variant to the work item if this is part of the variant.

Query-Based Variant Representation

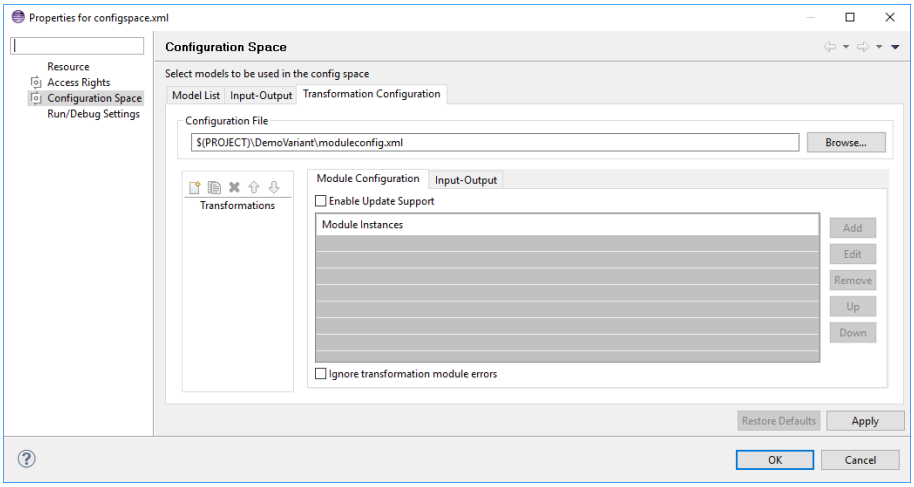
The query-based representation creates variant-specific queries of each query in a designated Azure DevOps query folder. Only those queries that are included in the configuration space as pure::variants models are considered. Work items may also be included in the variant-specific queries even if they have not been explicitly selected by the user in the variant description model. This is the case if the work item element has been selected by its parent element but has not been selected in the variant description model. These work items must be included in order to keep the query tree structure intact.

Preparing a Transformation

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

The configuration space property dialog opens and the *Transformation Configuration* tab is shown. Next step is to add a new *Module Configuration* by clicking the marked tool bar item (see Figure 12, “Transformation Configuration”). Now add a new Module to the Module Configuration, using the **Add** button.

Figure 12. Transformation Configuration



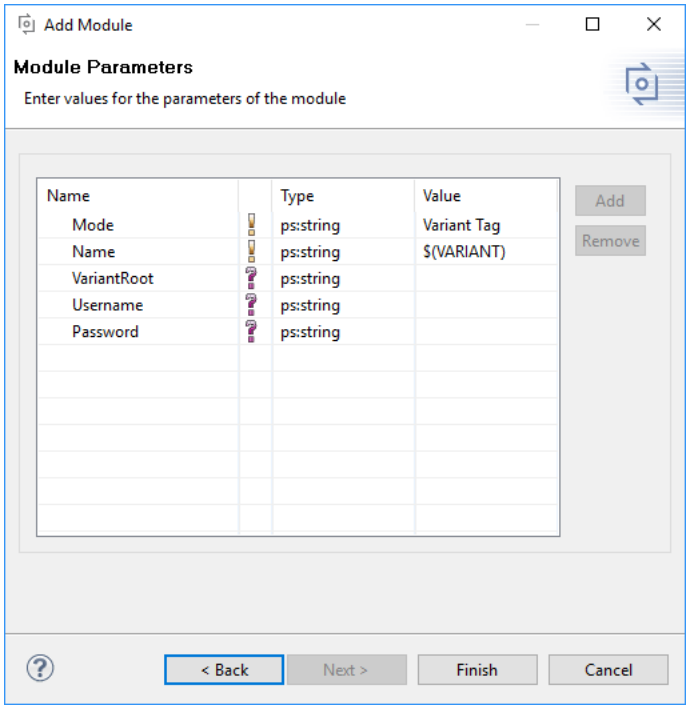
A new Dialog comes up. Choose **Microsoft Azure DevOps Module** and enter a name. The next page shows some parameters:

- Username - The username for connecting to Azure DevOps server.
- Password - The password for connecting to Azure DevOps server.
- VariantRoot - The path to query folder where the variant specific query will be stored. (Only for "Variant Query")
- Mode - Select one of the export modes.
  - *Variant Tag* adds to every work item, which is included in variant, the appropriate variant name as a Tag.
  - *Variant Query* instantiates a new query including all variant specific work items as links.
- Name - This name describes the suffix of the resulting variant name. It is calculated in time of transformation.

  "Variant Tag" concatenates the prefix 'p::v' with user defined suffix. E.g. 'p::v $(Variant)' results in 'p::v Luxury'.

  "Variant Query" concatenates the query name with the actual variant name. E.g. ' - $(Variant)' results in 'MasterQuery - Luxury'.

Figure 13. Module Parameter Page

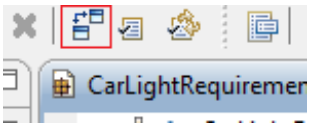


After finishing the dialogs, the transformation can simply be used by clicking on the Transformation button in the tool bar and choosing the transformation configuration in the pull down menu.

2.7. Updating Models from Azure DevOps

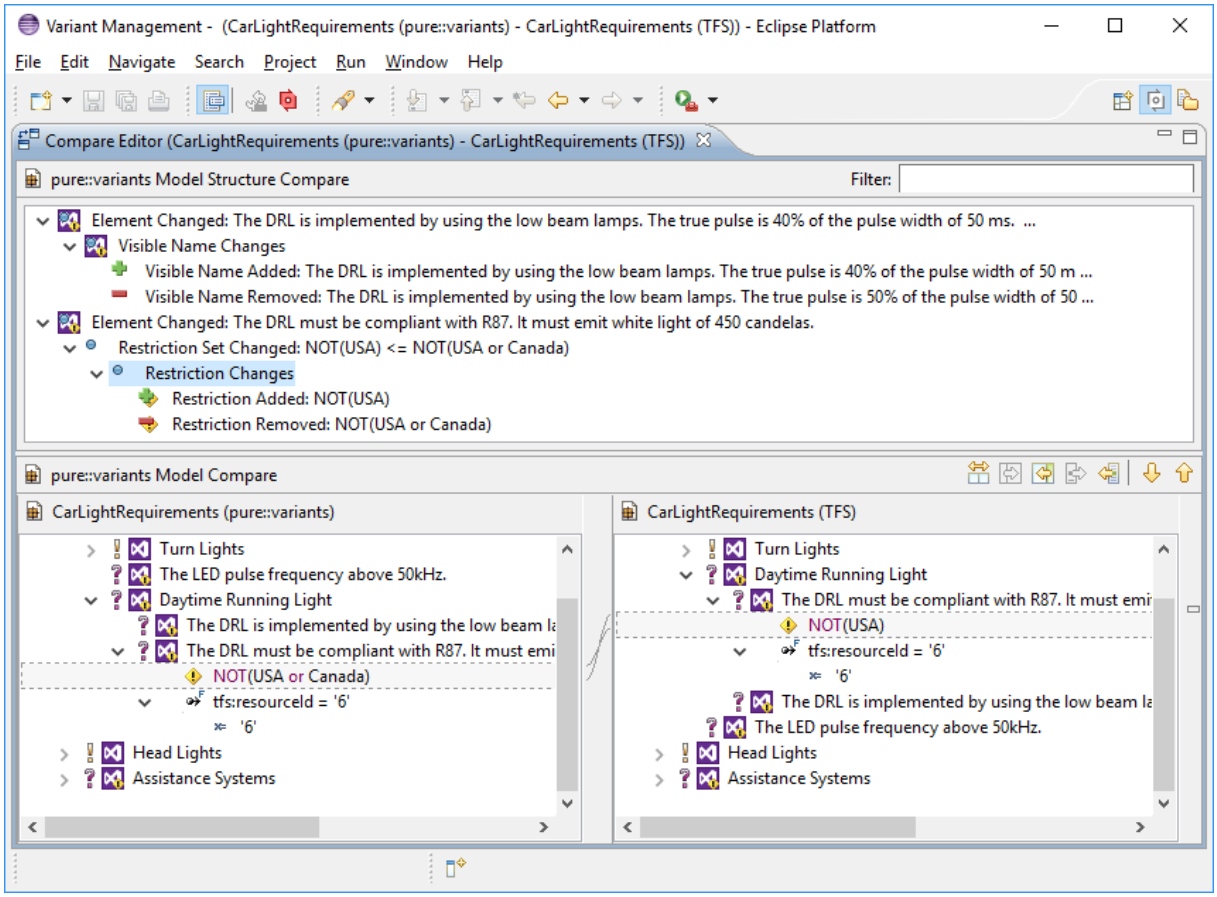
Since there is no live connection between the Azure DevOps database and pure::variants, it is necessary to update the pure::variants models with information from Azure DevOps whenever relevant changes have been made. To facilitate the synchronization, pure::variants provides a **Synchronize** action. To start the update, open the model representing the Azure DevOps query and press the **Synchronize** button in the tool bar (see Figure 14, “Synchronize model”). pure::variants will connect to Azure DevOps and present the so called Compare Editor for pure::variants models (see Figure 15, “Model update from Azure DevOps in Compare Editor”).

Figure 14. Synchronize model



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

Figure 15. Model update from Azure DevOps in Compare Editor



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

3. Using the Integration

To support users of the pure::variants - Connector for Azure DevOps in adding variability information to work items, the pure::variants web extension and pure::variants Desktop Hub is provided. It can be used to define restrictions (see the section called “Defining Element Restrictions”). To this end, it provides an editor featuring auto-completion proposals, syntax highlighting and a configuration-space representation.

The instructions for installing and activating the Integration can be found in ???.

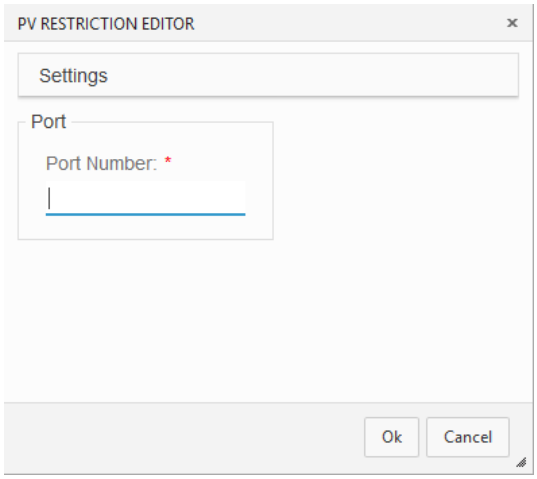
3.1. First Use

When using the pure::variants Web Integration the first time, you have to make sure to have setup the same port for Web Integration and Desktop Hub. Please see documentation of pure::variants Desktop Hub, navigating to **Windows Start->pure-variants Enterprise->Desktop Hub Manual**. For pure::variants Azure DevOps Integration, navigate to one of your work items, having the pvRestriction attribute configured, to open the restriction editor dialog by pressing the three-dotted-button (see Figure 16, “A pvRestriction field with pure::variants Azure DevOps Integration”).

Figure 16. A pvRestriction field with pure::variants Azure DevOps Integration

Navigate to **Settings** and type the same port number as configured for pure::variants Desktop Hub. Please ensure to press **OK** to persist your configured port number.

Figure 17. Settings page of Web Integration



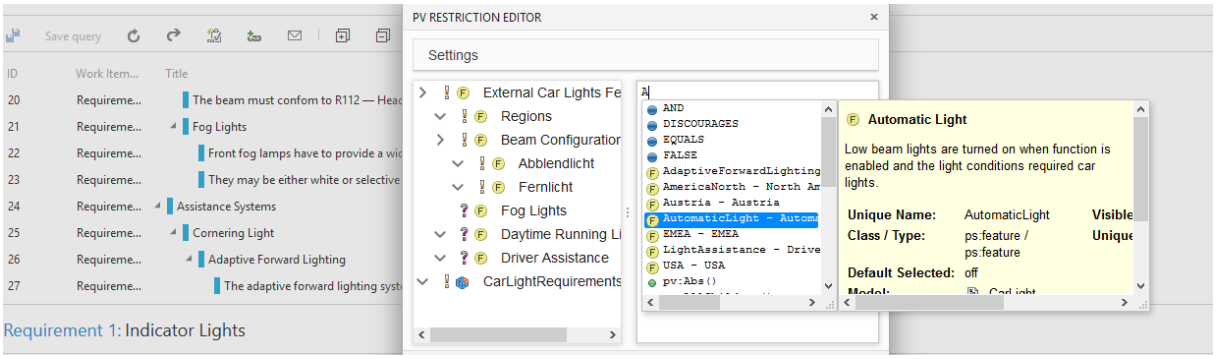
Note

The port number will be stored in your currently used web-browser.

3.2. Add variability information

For editing variability information, we provide the ability to have auto-completion proposals and syntax hightlighting while writing restrictions for Azure DevOps work items. This only requires to load your appropriate configuration-space in pure::variants Desktop Hub. For guidance, see *Desktop Hub Manual*. Navigate to a variability-aware work item and open press the three-dotted-button (see Figure 16, “A pvRestriction field with pure::variants Azure DevOps Integration”) next to pvRestriction attribute. Now you can enter an appropriate restriction in the pvSCL Restriction Editor. While entering, you can press CTRL and spacebar keys to have a proposal of relevant feature names, pvSCL functions and keywords. When you are done with defining the restriction, press **OK**, thus the entered restriction will be applied to the work item.

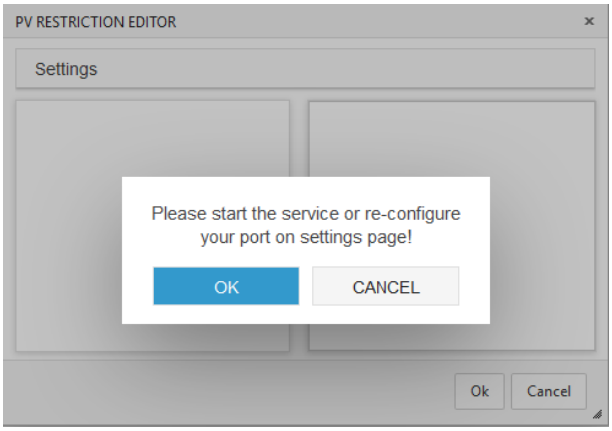
Figure 18. Add restriction with auto-completion proposal



3.3. Connection Problem

The following dialog is shown in the Web Integration when there is a problem to connect to the Desktop Hub.

Figure 19. Unable to connect to Desktop Hub



Please verify your settings according to the following actions:

- Check if the port number of the Web Integration and Desktop Hub does match. For Desktop Hub, see *Desktop Hub Manual* how to setup the port. For Web Integration navigate to **Settings** page.
- Check if a secured connection is configured for Azure DevOps server and Desktop Hub. For Azure DevOps server verify if the address bar begins with *https://...* in your web-browser. In case of Desktop Hub, see *Desktop Hub Manual* for setting up a secured connection.
- Check if web-browser accepts Desktop Hub connection if secured connection is enabled. Navigate to the following address in your web-browser and when asked, accept the self-signed certificate, once: *https://localhost:[port]/pv/version*.

4. Advanced Topics

4.1. Adding Variability Information in Azure DevOps

Defining an Element Variability Type

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

Defining an Element Name

Each imported Azure DevOps work item gets only a visible name assigned. But to use the Azure DevOps work item in restrictions, constraints and calculations of pure::variants, a unique name is required.

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

Defining Element Restrictions

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

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

Defining Element Constraints

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

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

Defining Element Default Selection

Each imported Azure DevOps work item gets an default selection state.

This is calculated using the following rules:

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

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

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

4.2. ANT transformation and synchronization

While the transformation and synchronization is supported for Azure DevOps queries a valid authentication is required. There are two possiblities for ANT transformation. Either the user credentials (username and password) are provided in the Azure DevOps transformation module or defined as environment variables. Therefore define *PV_TFS_USER* for username and *PV_TFS_PASSWORD* for password. For ANT synchronization only environment variables are suitable.