pure-systems pure variants Connector for codebeamer User Manual

pure-systems pure variants Connector for codebeamer

Contents [hide

1 Introduction
  1.1 About this manual
  1.2 Software Requirements
  1.3 Installation
2 Using the Connector
  2.1 Starting pure::variants
  2.2 Preparing the codebeamer Project
  2.3 Authentication
  2.4 Creating the Initial Model(s)
  2.5 Updating Models from codebeamer
  2.6 Defining a Variant
  2.7 Transforming a Variant
3 Using the Integration
  3.1 Adding Variability Information Using the Desktop Hub
  3.2 Adding Variability Information Using the pure::variants Widget
  3.3 . Introduction to Integration GUI
  3.4 Working with Restriction Editor
  3.5 Working with Calculations Editor
  3.6 Working with Test Steps
  3.7 Error Check
  3.8 Variability in WIKI-Tables
4 Restrictions
  4.1 Limitations Relating Text Substitution in WikiText Fields
  4.2 Known Limitations of the Supported codebeamer Versions
5 Documents / Resources
6 Related Posts

Introduction

pure::variants The connector for codebeamer enables codebeamer users to manage requirements as well as test artefacts variability using pure::variants. By coupling pure::variants and codebeamers, knowledge about variability and variants can be formalized, shared, and automatically evaluated. This enables getting answers to questions about valid combinations of requirements and test artefacts in product variants quickly; permits easy monitoring of planned and released product variants at the requirements and test artefact level, and also permits very efficient production of variant-specific requirements documents, test sets and test cases out of the requirements and test artefact repository respectively.

About this manual

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

Software Requirements

The following software is required by the pure::variants Connector for codebeamer:

- PTC codebeamer 22.10-LTS or 22.10-SP3. Compatibility with other codebeamer releases is not guaranteed.
- pure::variants server component for codebeamer in the same version as the connector. The connector requires the deployment of a pure::variants specific components on the codebeamer server
- pure::variants Desktop Hub or Web Hub also in the same version as the connector itself:
  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 the installer wizard, while the installation of the Web Hub is described in the pure::variants Setup Guide.

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

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).Installation steps specific to the codebeamer connector, deployment of the components, as well as description of how to configure single sign-on are described in the pure::variants Setup Guide

Installation steps specific to the codebeamer connector, deployment of the components, as well as description of how to configure single sign-on are described in the pure::variants Setup Guide

Using the Connector

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.

Preparing the codebeamer Project

In order to get the variability information from codebeamer items as well as to assign items to variants in codebeamer, the codebeamer trackers need to be prepared initially. To make the variability information available to pure::variants, an attribute has to be set for every codebeamer tracker that shall be processed with respect to its variability.
To set this attribute for each selected tracker, go to CodeBeamer and use the Configure option of the tracker. Here, select the Fields page and add a new custom field named ‘pvRestriction’ of the type ‘Text’. Furthermore, to prepare the Enum transformation to store variability information in codebeamer, a further custom field is needed, named ‘pvVariants’ of the type ‘Text’.
For Test Steps within Test Cases, in the Table Definition, the custom fields ‘pvRestrictionTestSteps’ and ‘pvVariantsTestSteps’ of the type ‘Text’ need to be added, respectively.

Authentication

To use the connector it is always required to be authenticated to the codebeamer application.
There are two authentication mechanisms supported

1. Using codebeamer credentials
2. OpenID Connect (for Single-Sign-On)

During the use of the connector, for both mechanisms, the user will be prompted with a login dialog, which expects the user’s credentials. In the case of Single-Sign-On a browser-based login dialog will be shown that is provided by the Authentication Server configured.

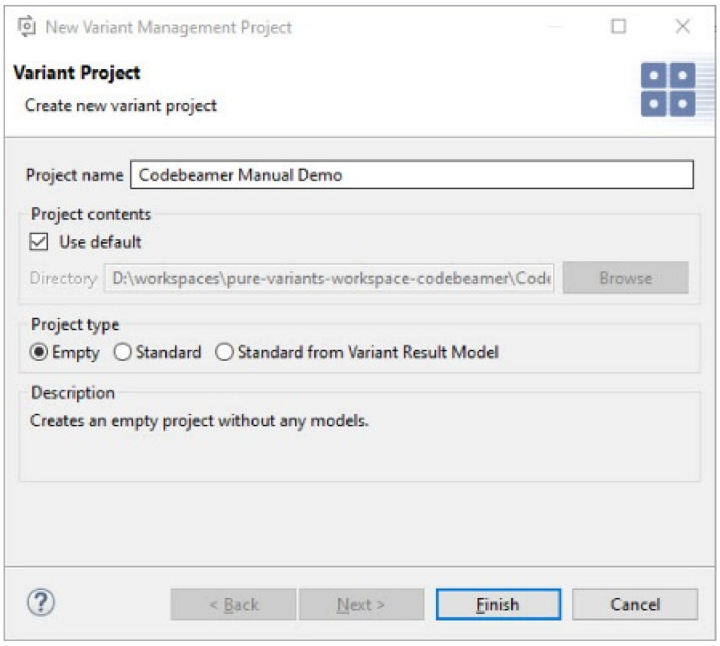
Creating the Initial Model(s)

The first step is always to create the corresponding family model for each relevant working set containing selected code beamer trackers. These initial family models serve as starting points for using existing variability information.

The import procedure has to be executed only once for each code beamer working set but can be updated afterwards Each tracker is represented by one model node element in the pure::variants family model that is created during the import.

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 the project type (see Figure 1, “Creating an empty Variant Management project for code beamer tracker import”)

Figure 1. Creating an empty Variant Management project for codebeamer tracker import



New Variant Management Project

Variant Project

Create new variant project

Project name: Codebeamer Manual Demo

Project contents

[✓] Use default

Directory: D:\workspaces\pure-variants-workspace-codebeamer\Code

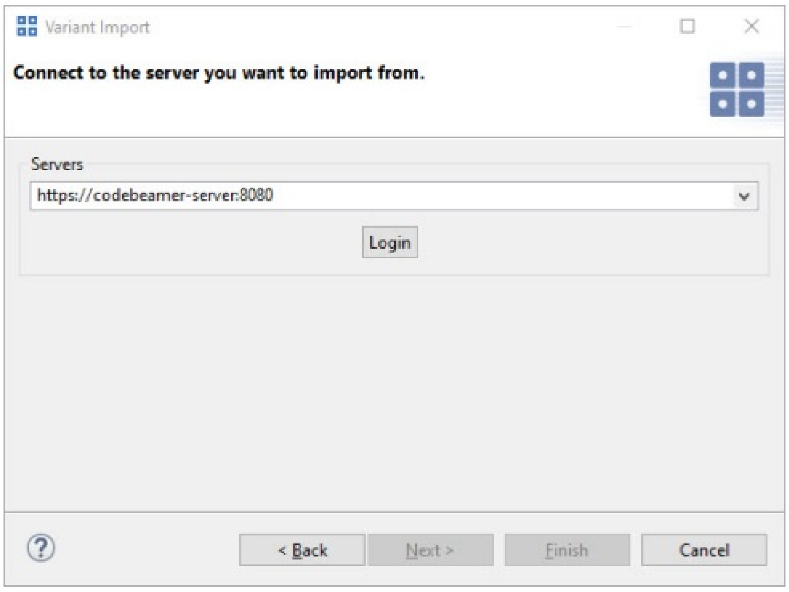
Project type: [● Empty] ○ Standard ○ Standard from Variant Result Model

Description

Creates an empty project without any models.

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 codebeamer trackers or working-sets’.
The import wizard appears. In the first page, you have to define or select the codebeamer server address you want to import the trackers from.

Figure 2. The server selection page in the codebeamer import wizard



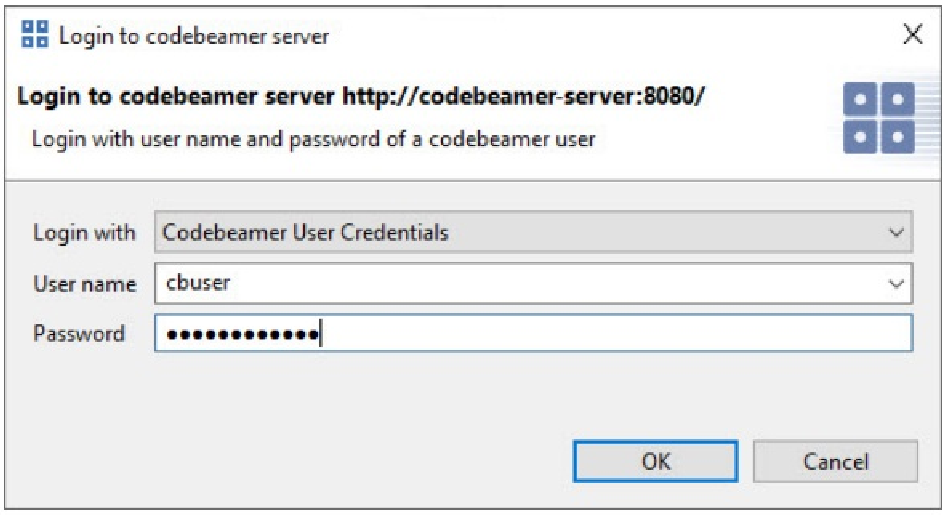
Variant Import

Connect to the server you want to import from.

Servers: https://codebeamer-server:8080

If you are not already authenticated, you can use Test Connection.This will open the login dialog that provides multiple possibilities for authentication.

Figure 3. Connecting to the codebeamer server using user credentials



Login to codebeamer server

Login to codebeamer server http://codebeamer-server:8080/

Login with user name and password of a codebeamer user

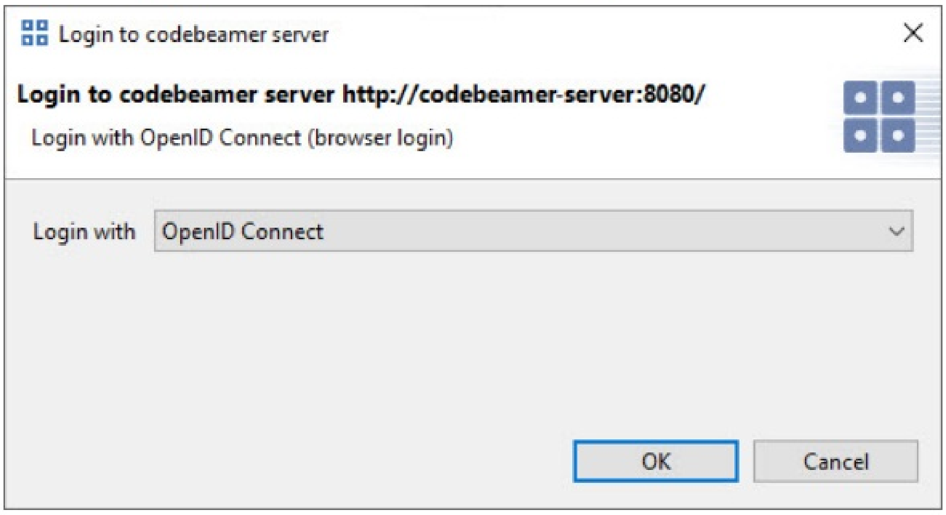
Login with: Codebeamer User Credentials

User name: cbuser

Password: ●●●●●●●●●●●●

1. The codebeamer user name and password can be provided with the ‘codebeamer User Credentials’ option.
2. Authentication server can be used selecting the ‘Authentication Server (OAuth2)’ option.

Figure 4. Connecting to the codebeamer server using Authentication Server (OAuth2)



Login to codebeamer server

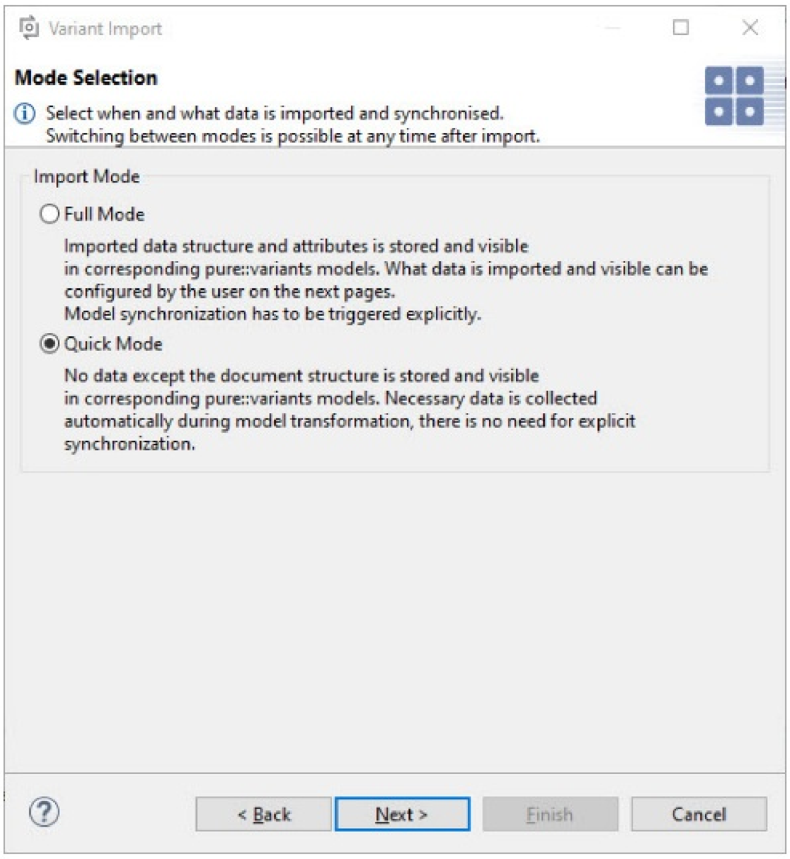
Login to codebeamer server http://codebeamer-server:8080/

Login with OpenID Connect (browser login)

Login with: OpenID Connect

Which login method is working for you depends on the configuration of the codebeamer server.
In the next page, you can decide whether you want to perform a full import of your codebeamer tracker’s variability information (Full Mode) or if you just want to import the module header (Quick Mode). In the latter case, the data is automatically synchronized before a transformation, whereas in full mode, the user is responsible for keeping the data synchronized, as the existing data is used to transform the variants.
Using Full Mode, variation points found in trackers are represented in the Family Model being created.

Figure 5. The mode selection page in the codebeamer import wiza



Mode Selection

Select when and what data is imported and synchronised.
Switching between modes is possible at any time after import.

Import Mode

( ) Full Mode

Imported data structure and attributes is stored and visible in corresponding pure::variants models. What data is imported and visible can be configured by the user on the next pages.
Model synchronization has to be triggered explicitly.

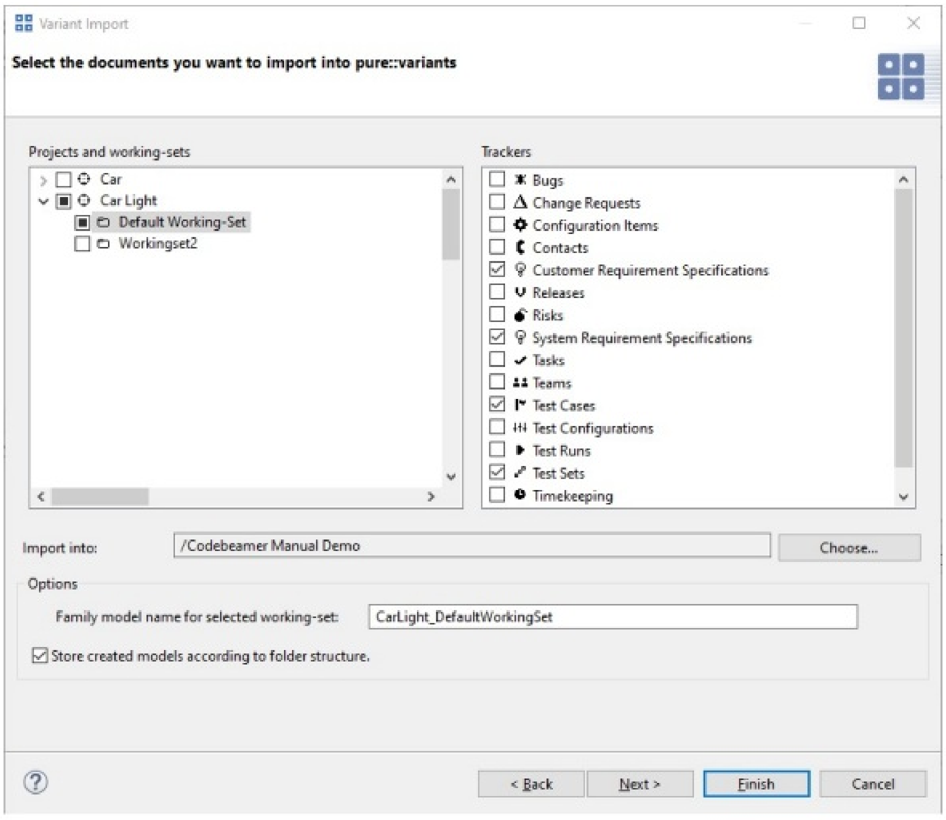
(●) Quick Mode

No data except the document structure is stored and visible in corresponding pure::variants models. Necessary data is collected automatically during model transformation, there is no need for explicit synchronization.

Which data will be imported can be configured by the user on the next page.
The complete list of projects of the codebeamer repository is shown, as well as the working-sets are listed below each project that are available. Navigate to the Working-Set containing the trackers of interest and select the check

boxes on the left side. Multiple Working-Sets from different projects can be selected at once for import. Selecting a check box on the left side for the Working-Set marks all trackers for import. Selection for individual trackers within one Working-Set is also possible using the right pane.
Note: Please note, only information is presented to which the user has appropriate access rights

Figure 6. The tracker selection page in the codebeamer import wizard


Variant Import

Select the documents you want to import into pure::variants

Projects and working-sets

- ☐ Car
- ■ Car Light
  - ■ Default Working-Set
  - ☐ Workingset2

Trackers

- ☐ Bugs
- ☐ Change Requests
- ☐ Configuration Items
- ☐ Contacts
- ✓ Customer Requirement Specifications
- ☐ Releases
- ☐ Risks
- ✓ System Requirement Specifications
- ☐ Tasks
- ☐ Teams
- ✓ Test Cases
- ☐ Test Configurations
- ☐ Test Runs
- ✓ Test Sets
- ☐ Timekeeping

Import into: /Codebeamer Manual Demo

Options

Family model name for selected working-set: CarLight_DefaultWorkingSet

✓ Store created models according to folder structure.

Make sure that the import target location given next to “Import into” is correct. The location can be changed by using the Choose button. Selecting the option “Store created models according to folder structure,” the import process creates folders for the project and the Working-sets respectively, in pure::variants for the family models.

The family models created are named by default according to the _ scheme, but this can be modified using the edit box.

Note: Although trackers of all types can be imported, only trackers of certain types will be considered during transformation (for a list of types, see chapter Transforming a Variant of this manual).

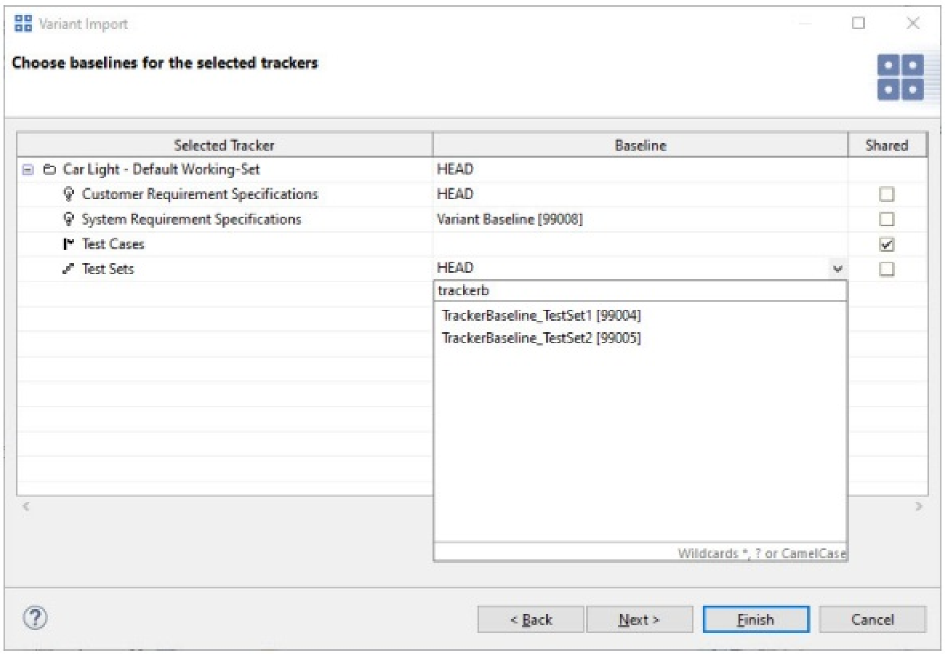
Also, variation points are only considered during import in these trackers.

Using the next page, you can select the baseline for each selected tracker to be used as the source version for Working Set Transformation

- The selection can be performed on Working Set level for relevant baselines or separately for each of the trackers. On Working Set level those baselines are listed that are common for each of the trackers. The selection is assisted by a search function that filters the baselines to be selected
- Alternatively, it can be defined for a tracker to be included as shared in the Working Set that is created by the Working Set Transformation

Note: The shared state of trackers that are by definition shared cannot be changed and is displayed as read-only in the dialog.

Figure 7. Select Tracker Baselines



Variant Import

Choose baselines for the selected trackers

Selected Tracker	Baseline	Shared
Car Light - Default Working-Set	HEAD	
Customer Requirement Specifications	HEAD	☐
System Requirement Specifications	Variant Baseline [99008]	☐
Test Cases		☑
Test Sets	HEAD	☐

trackerb

TrackerBaseline_TestSet1 [99004]
TrackerBaseline_TestSet2 [99005]

Wildcards *, ? or CamelCase

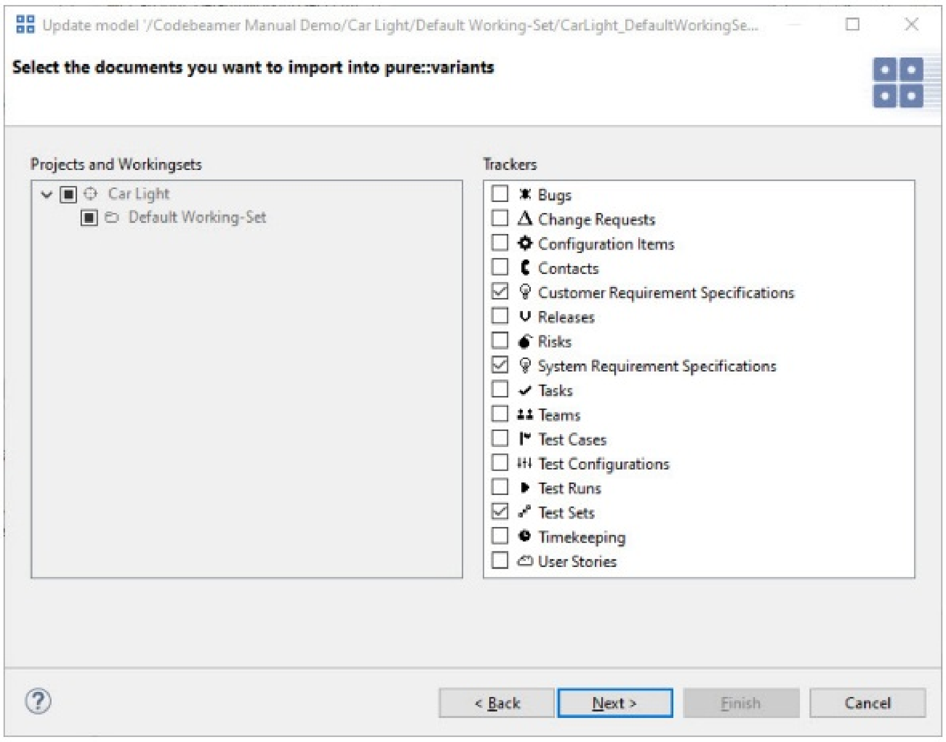
< Back | Next > | Finish | Cancel

On the following page, the Import Rules are shown. On this page, you can select sets of Import Rules, that 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 codebeamer artefacts information.

Updating Models from codebeamer

Using the Synchronize action, the set of trackers to be imported as part of a Working-Set can be modified. Additionally, when using Full Mode, it is necessary to update the pure::variants models with information from codebeamer whenever relevant changes have been made. To start the update, open the model representing the Working-Set and press the Synchronize button in the tool bar.

Figure 8. Synchronize model



Update model '/Codebeamer Manual Demo/Car Light/Default Working-Set/CarLight_DefaultWorkingSe...

Select the documents you want to import into pure::variants

Projects and Workingsets

- ■ Car Light
  - ■ Default Working-Set

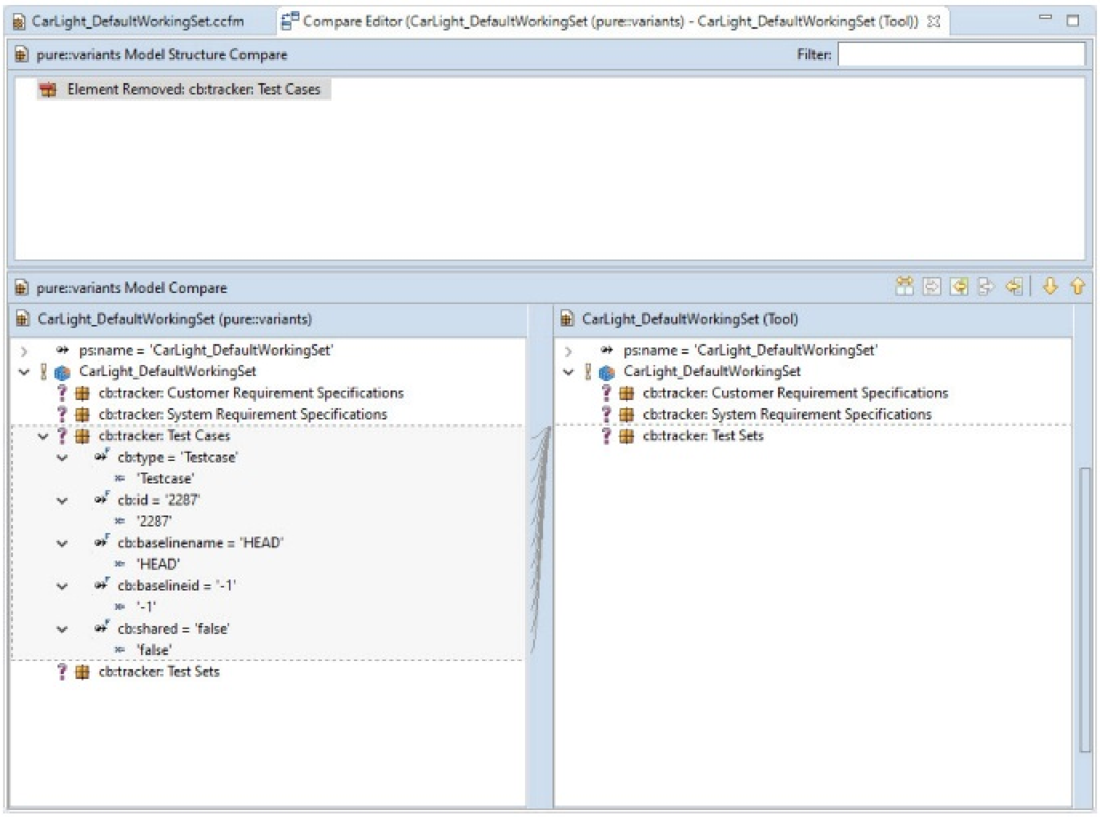
Trackers

- ☐ Bugs
- ☐ Change Requests
- ☐ Configuration Items
- ☐ Contacts
- ☑ Customer Requirement Specifications
- ☐ Releases
- ☐ Risks
- ☑ System Requirement Specifications
- ☐ Tasks
- ☐ Teams
- ☐ Test Cases
- ☐ Test Configurations
- ☐ Test Runs
- ☑ Test Sets
- ☐ Timekeeping
- ☐ User Stories

< Back | Next > | Finish | Cancel

pure::variants will connect to codebeamer to present the tracker selection page, the baseline selection page, and subsequently the Compare Editor for pure::variants models.

Figure 9. Synchronize model compare


CarLight_DefaultWorkingSet.ccfm | Compare Editor (CarLight_DefaultWorkingSet (pure::variants) - CarLight_DefaultWorkingSet (Tool))

pure::variants Model Structure Compare — Filter:

Element Removed: cb:tracker: Test Cases

pure::variants Model Compare

CarLight_DefaultWorkingSet (pure::variants)

```
ps:name = 'CarLight_DefaultWorkingSet'
CarLight_DefaultWorkingSet
    cb:tracker: Customer Requirement Specifications
    cb:tracker: System Requirement Specifications
    cb:tracker: Test Cases
        cb:type = 'Testcase'
            'Testcase'
        cb:id = '2287'
            '2287'
        cb:baselinename = 'HEAD'
            'HEAD'
        cb:baselineid = '-1'
            '-1'
        cb:shared = 'false'
            'false'
    cb:tracker: Test Sets
```

CarLight_DefaultWorkingSet (Tool)

```
ps:name = 'CarLight_DefaultWorkingSet'
CarLight_DefaultWorkingSet
    cb:tracker: Customer Requirement Specifications
    cb:tracker: System Requirement Specifications
    cb:tracker: Test Sets
```

The compare editor is used throughout pure::variants to compare model versions, but in this case it is used to compare the codebeamer 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.

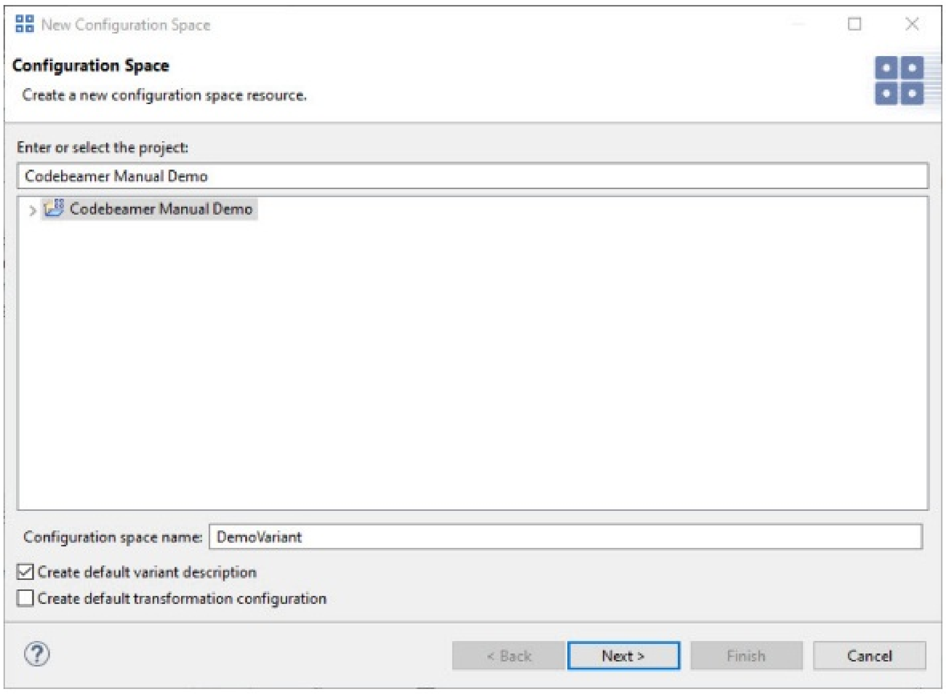
Selecting an item in this list highlights the respective change in both models. In this example, the tracker “Test Cases” was removed from the scope of the import.
The Merge toolbar provides tools to copy single or even all (non-conflicting) changes as a whole from the current model to the out-dated model.

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 that are of interest to 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 10, “The Configuration Space Wizard”)**, 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 requirements models the standard transformation does not provide any relevant functionality (See the pure::variants User Manual for more information on transformations

Figure 10. The Configuration Space Wizard



New Configuration Space

Configuration Space

Create a new configuration space resource.

Enter or select the project:

Codebeamer Manual Demo

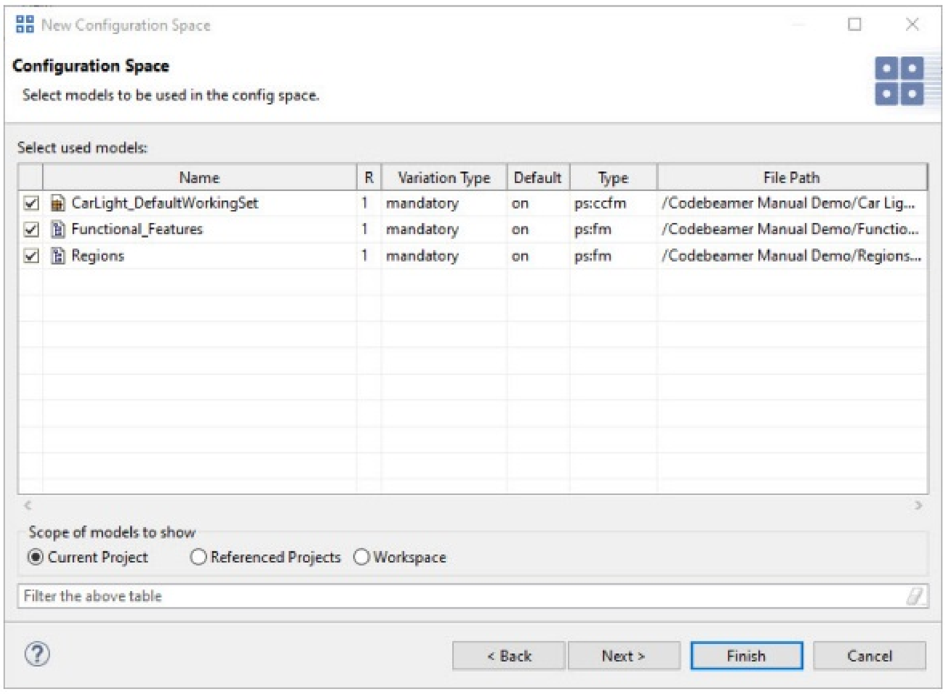
Configuration space name: DemoVariant

[✓] Create default variant description

[ ] Create default transformation configuration

The next page is used to specify which models are to be included in this configuration space. Select here all models that represent the Working-Sets and so the trackers of interest. In the example below, just one Family Model is selected. Now press the Finish button.

Figure 11. The Configuration Space Wizard Model Selection Page



New Configuration Space

Configuration Space

Select models to be used in the config space.

Select used models:

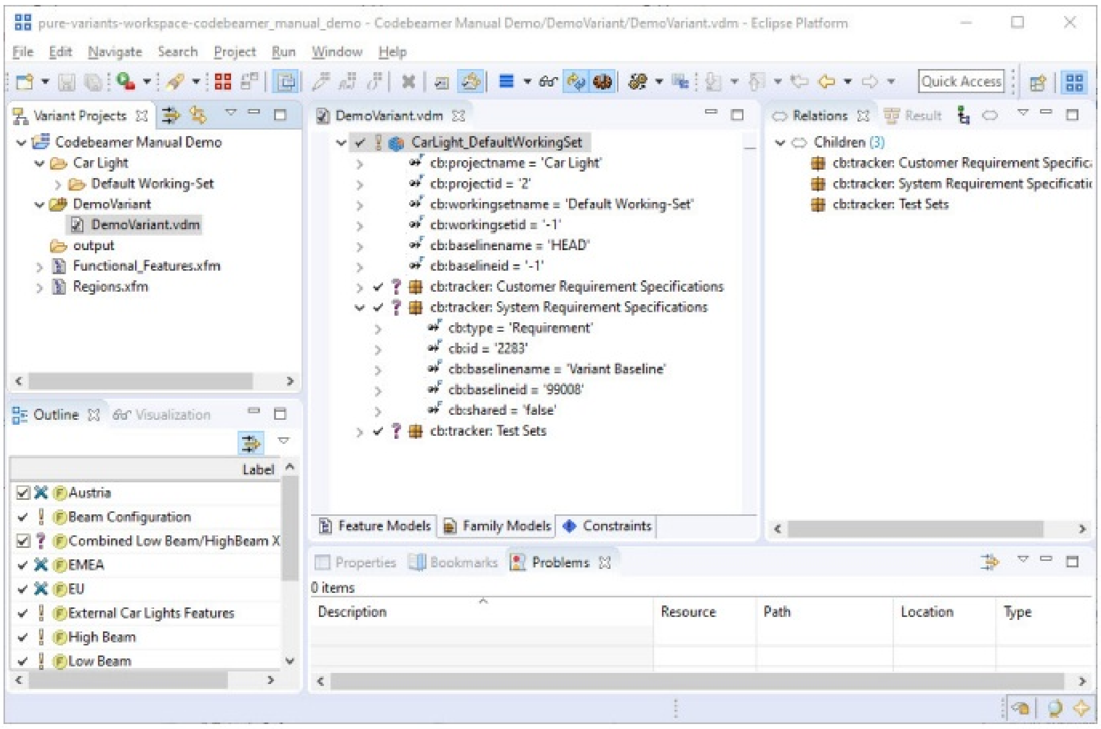
	Name	R	Variation Type	Default	Type	File Path
✓	CarLight_DefaultWorkingSet	1	mandatory	on	ps:ccfm	/Codebeamer Manual Demo/Car Lig...
✓	Functional_Features	1	mandatory	on	ps:fm	/Codebeamer Manual Demo/Functio...
✓	Regions	1	mandatory	on	ps:fm	/Codebeamer Manual Demo/Regions...

Scope of models to show

[● Current Project] ○ Referenced Projects ○ Workspace

The resulting project structure is shown in (Figure 12, “Initial Configuration Space Structure”). The DemoVariants.vdm is created and immediately opened in case the Create default variant description was selected on the first page of the wizard.

Figure 12. Initial Configuration Space Structure



pure-variants-workspace-codebeamer_manual_demo - Codebeamer Manual Demo/DemoVariant/DemoVariant.vdm - Eclipse Platform

```
Codebeamer Manual Demo
  Car Light
    Default Working-Set
  DemoVariant
    DemoVariant.vdm
  output
  Functional_Features.xfm
  Regions.xfm
```

```
CarLight_DefaultWorkingSet
  cb:projectname = 'Car Light'
  cb:projectid = '2'
  cb:workingsetname = 'Default Working-Set'
  cb:workingsetid = '-1'
  cb:baselinename = 'HEAD'
  cb:baselineid = '-1'
  cb:tracker: Customer Requirement Specifications
  cb:tracker: System Requirement Specifications
    cb:type = 'Requirement'
    cb:id = '2283'
    cb:baselinename = 'Variant Baseline'
    cb:baselineid = '99008'
    cb:shared = 'false'
  cb:tracker: Test Sets
```

```
Children (3)
  cb:tracker: Customer Requirement Specific
  cb:tracker: System Requirement Specificatio
  cb:tracker: Test Sets
```

```
Label
Austria
Beam Configuration
Combined Low Beam/HighBeam X
EMEA
EU
External Car Lights Features
High Beam
Low Beam
```

0 items

Description	Resource	Path	Location	Type

Transforming a Variant

Variants stored in a variant description model can be made available in codebeamer. The Connector supports the following ways of representing variants: attribute-based

Attribute-Based Variant Representation

In the attribute-based representation we define a custom field for each selected codebeamer tracker to be added to each tracker item. This transformation modus adds the name of the variants (as a list, separated by new lines) if the tracker item is part of the variant. The name of this attribute can be user defined for a transformation, default is pvVariants.

This type of transformation is applicable to the following tracker types: Requirement, Test Case (incl. Test Steps), Test Set, Test Configuration, Configuration Items

Note: In case a tracker is excluded by the user for a variant in the Family Model, the variant name is going to be removed from this list of variants in codebeamer.

Note: This transformation cannot be run on trackers with baselines other than HEAD and therefore pure::variants will report an error.

Working Set Transformation

Product Line assets (150%) i.e. trackers (requirements, test cases, etc.) are assigned to a dedicated Working Set or the Default Working Set in codebeamer. The Working Set

Transformation can be used with or without update support:

- Without update support, the transformation creates a variant specific Working Set for each transformation run and for each variant (vdm) with trackers containing the variant specific subsets of tracker items (100%). By default, the naming convention for the variant specific Working Set created equals to the , e.g. ‚DemoVariant'. The default name can be changed by setting the transformation parameter 'WorkingSetName' (see chapter 'Preparing a Transformation '). In case a Working Set with the same name already exists, an error is reported and no transformation is performed.

A practical way to create differently named working sets for each run is to append the timestamp of the transformation to the name. You can do it by setting the value of ‘WorkingSetName’ to ‘$(VARIANT)_$(QUALIFIER)’.
- With update support, the update of previously transformed variant specific trackers is supported. There are two modes available, the Manual Merge Mode, and the Full Overwrite Mode

Manual Merge Mode – in this mode, the variant is represented by two working sets, a reference- and a working-copy working set. The reference working set is created new in the first transformation run and is overwritten in each further transformation. The working-copy is also created by the first transformation and the content can be modified by the user. This working set is not updated automatically, but the changes made in the reference working set need to be merged manually into the working copy.

Full Overwrite Mode: in this mode, the variant is represented by one working set that is created by the first transformation and is overwritten in each subsequent transformation.

In needs to be decided upfront which update mode to use, switching between the modes after transformations were already performed is not possible.

Note:

- The user shall have the tracker level permission to replace the content of a branch (permission setting in codebeamer: ‘Branch – Replace content’)
- In some tracker combinations, the working set creation in codebeamer changes the tracker configuration however is not supported by the update. This restricts the usability of the update of certain tracker combinations, e.g. the trackers ‘System Requirement Specification’ and ‘Customer Requirement Specification’ always need to be included in the working set at the same time.
- See ‘pure::variants User’s Guide’, Chapter ‘Setting up a Transformation’ for information how to enable update support.

For all modes of the Working Set Transformation, following needs to be noted:

- The user shall have permission to create working sets. (permission setting in codebeamer: ‘Working-Set – Admin’).
- Included trackers (i.e. not shared trackers) are branched off from the trackers in the originating Working Set at the baseline defined by the user during import and are reduced to the variant specific subset. Shared trackers are only added to the Working Set but without any change as variability information in not considered there.
- The HEAD version of a variant specific tracker branch (if appropriate) can be only modified in codebeamer to include variant exclusive content.

This type of transformation is applicable to following tracker types: Requirement, Test Case (incl. Test Steps), Test Set, Configuration Items.

Text Substitution

Following tracker types and fields are subject to text substitution:

- Requirement (Name, Description) Test Case (Name, Pre-Action, Post-Action, Test Parameters (both test parameter
- names and their values), Description) Test Steps (All fields of type Text and Wikitext)
- Test Set (Name, Test Parameters (both test parameter names and their values), Description)
- Configuration Items (Name, Description)

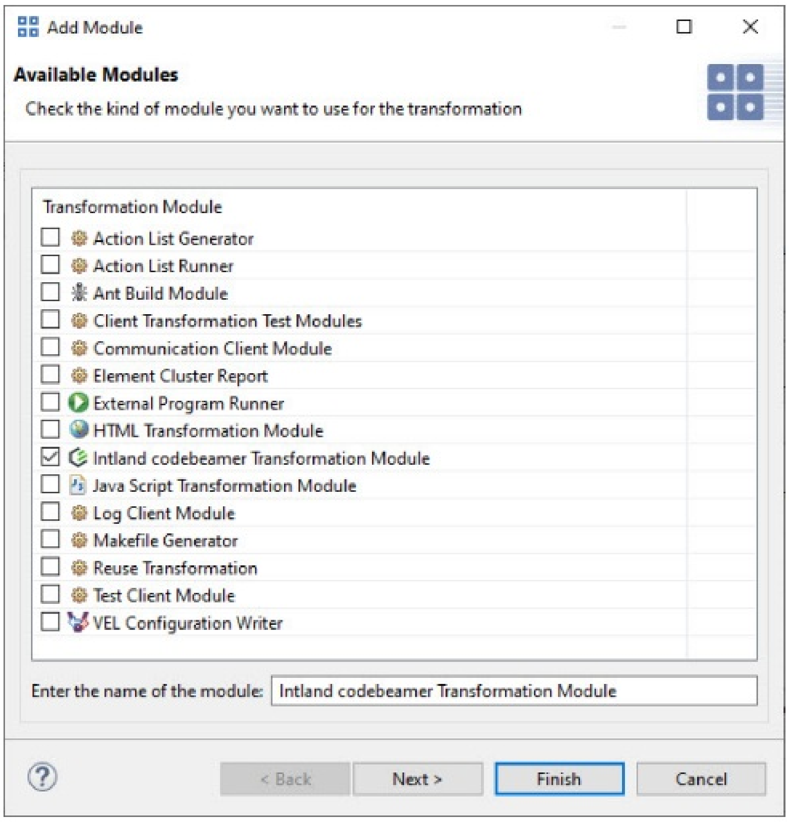
Note: Text substitution is performed only during ‘Working Set Transformation’ and only for trackers that are not included in a working set as shared.

Preparing a Transformation

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…

The configuration space property dialog opens, and the Transformation Configuration tab is shown. The 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 13. Transformation Configuration


Add Module

Available Modules

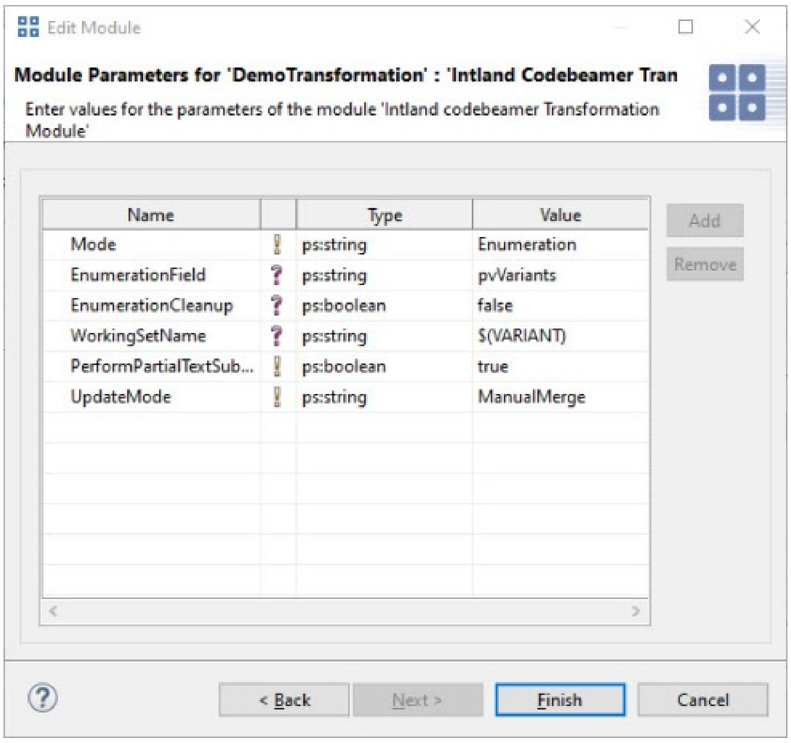
Check the kind of module you want to use for the transformation

Transformation Module
- [ ] Action List Generator
- [ ] Action List Runner
- [ ] Ant Build Module
- [ ] Client Transformation Test Modules
- [ ] Communication Client Module
- [ ] Element Cluster Report
- [ ] External Program Runner
- [ ] HTML Transformation Module
- [x] Intland codebeamer Transformation Module
- [ ] Java Script Transformation Module
- [ ] Log Client Module
- [ ] Makefile Generator
- [ ] Reuse Transformation
- [ ] Test Client Module
- [ ] VEL Configuration Writer

Enter the name of the module: Intland codebeamer Transformation Module

From the opened dialog, choose Intland codebeamer Transformation Module and enter a name. The next pageshows all parameters. The Modus parameter specifies one of the variant result representations, as described abov

Figure 14. Module Parameter Page


Edit Module

Module Parameters for 'DemoTransformation' : 'Intland Codebeamer Tran

Enter values for the parameters of the module 'Intland codebeamer Transformation Module'

Name		Type	Value
Mode	!	ps:string	Enumeration
EnumerationField	?	ps:string	pvVariants
EnumerationCleanup	?	ps:boolean	false
WorkingSetName	?	ps:string	$(VARIANT)
PerformPartialTextSub...	!	ps:boolean	true
UpdateMode	!	ps:string	ManualMerge

Following parameters need to be defined:

- Mode: Define the transformation mode. The available modes are: Enumeration – this option stands for the Attribute-Based Variant Representation. Working-Set – this option stands for the Working Set Transformation.
- EnumerationField: Specifies the name of the tracker item field to be filled with variant names in Enumeration transformation mode. If not set standard name ('pvVariants') is used.
- EnumerationCleanup: If true is selected, all existing variant attributes are removed before exporting the current variant.If false, only the names of the transformed variant will be updated (either removed or added).
- WorkingSetName: Specifies the name of the Working Set created by the transformation.
- PerformPartialTextSubstitution: If true is selected, the partial text substitution is performed.
- UpdateMode: Defines the update mode for Working Set Transformation in case update support is enabled. The available update modes are:

ManualMerge -this option stands for the Manual Merge Mode
FullOverwrite – this option stands for the Full Overwrite Mode After finishing the dialogs, the transformation can simply be used by clicking on the **Transformation** button in the tool bar and choosing the transformation from the pull down menu.

Web Client Integration for transformation

Please consult section Transformatio Help Contents in the pure::variants Web Client Manual for detailed information on how to perform Transformation using Web Client Integration

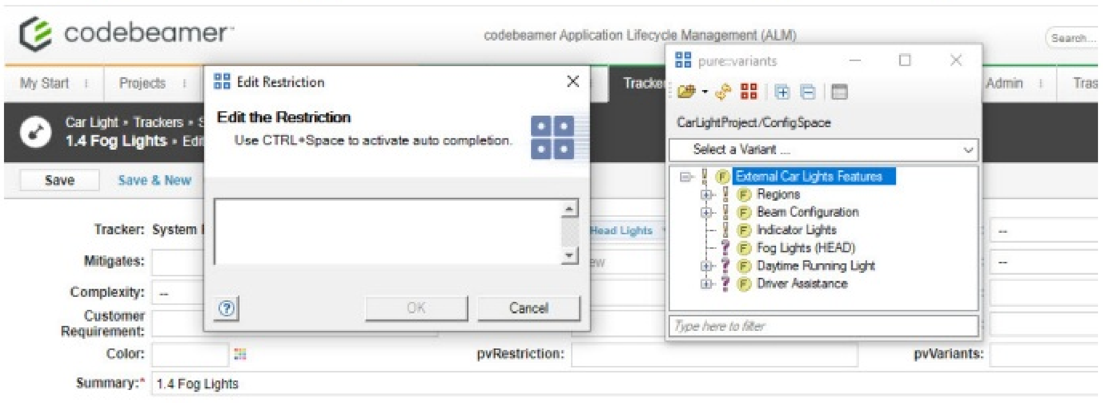
Using the Integration

In order to facilitate the pure::variants connector for codebeamer variability, information needs to be added to the tracker items. This is performed by adding restrictions to tracker items and is assisted by the Desktop Hub application that is provided by the pure::variants client installation or the in-tool integration called pure::variants Integration for CodeBeamer.

Adding Variability Information Using the Desktop Hub

The Desktop Hub uses the clipboard to insert information from pure::variants into other applications by pasting it into active fields being edited by the user. In codebeamer, the tracker item needs to be opened in edit mode first, then select the “pvRestriction” field before activating Desktop Hub using the hotkey combination set.

Figure 15. Adding restrictions in codebeamer using the Desktop Hub



Edit the Restriction
Use CTRL+Space to activate auto completion.

Tracker: System
Mitigates:
Complexity: --
Customer Requirement:
Color:
pvRestriction:
pvVariants:
Summary:* 1.4 Fog Lights

Note: More information on the Desktop Hub can be found in the dedicated pure::variants Desktop Hub Manual’

Adding Variability Information Using the pure::variants Widget

Once the Integration has been added to codebeamer (see respective chapter ‘pure::variants Connectors’ in the pure::variants Setup Guide) for the very first time, the General tab view under the Settings page will be shown which basically takes the input from the end-user to select between one of the two available modes, Integration should run into i.e. Desktop Hub mode or Web Hub mode. By default, Desktop Hub mode is being set as the default mode

Prerequisites for the Desktop Hub Mode

In order to run the integration in Desktop Hub mode, a running instance of the Desktop Hub is required in the background. While the Desktop Hub instance is running, inside the Integration, go to the General tab view under the Settings page. Notice, that the Desktop Hub is already selected in Connect via the drop-down (that’s because Desktop Hub is the default mode setting of the integration); the only thing required is the port number on which the Desktop Hub instance is running, hence, enter the port number inside the given Desktop Hub input type. Afterward, press the OK button in order to save the mode settings. Integration will then redirect to its main page and start running in Desktop Hub mode.

For loading Configuration Space in Desktop Hub Mode: In order to select a Configuration Space please press the Open Configuration Space button from the Integration’s menu bar. The Desktop Hub’s file selection dialog will be shown to select the desired Configuration Space. Once the Configuration Space is selected, the Integration will immediately show the selected Configuration Space.

Prerequisites for the Web Hub Mode

In order to run the integration in Web Hub mode, a running instance of the pure::variants Web Components is required (see chapter “pure::variants Web Components’ in the ‘pure::variants Setup Guide”). While the pure::variants Web Components is running, inside the Integration on the General tab view under the Settings page, select the Web Hub value from the Connect via drop-down and then enter the URI to the running instance of the pure::variant Web Components in the given Web Hub input type. Afterwards, press the OK button to save the mode settings. Integration will then redirect to its main page and start running in Web Hub mode

Define Transformation Related Settings

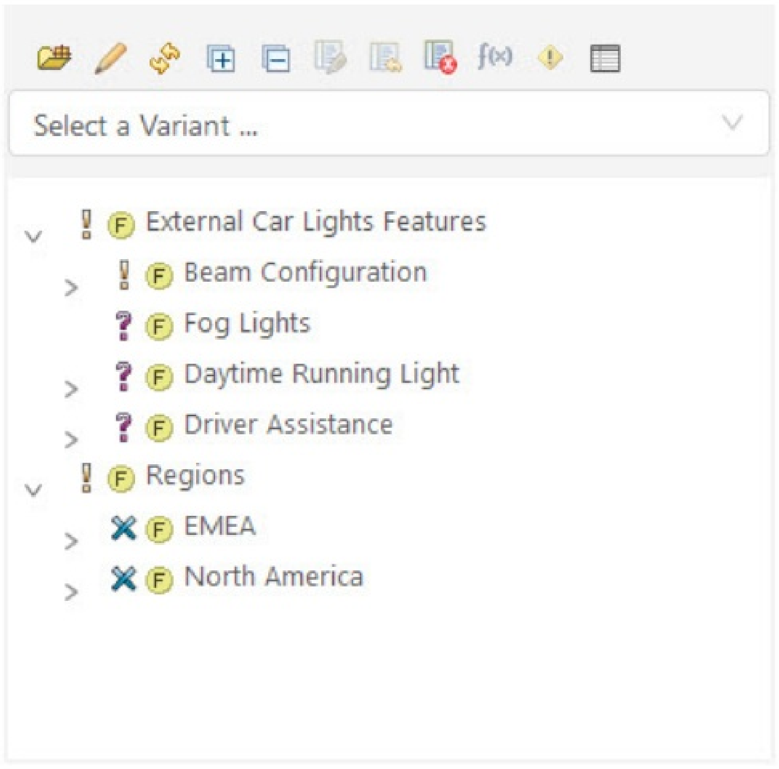
On the Settings page, further transformation relevant settings can be defined for the active tracker instance:

- Using the General tab, the attribute used for storing the restrictions can be defined. The default value is ‘pvRestriction’.
- For tracker items inside Table-Fields, the name of the field is appended without whitespace and special characters to the value defined above, e.g. for ”Test Steps’ it results in ‘pvRestrictionTestSteps’
- Using the Calculation tab, the text substitutions markers can be defined. The default values are: Opening character is [Closing character is] Escape character is

. Introduction to Integration GUI

The Main page view of the Integration is shown in Figure 16, “Integration Main page view”

Figure 16. Integration *Main* page view



The integration is displayed in the side panel of the Document View and is available for supported tracker types where the DocumentView is available. Function of the buttons of the menu bar, from left to right:

1. Refers to Open Config Space button – click to select the configuration space as explained in the Desktop Hub and the Web Hub sections.
2. Refers to Model Viewer button – click to open currently selected Configspace/VDM in the Model Viewer web application. (Only visible in the Web Hub mode)
3. Refers to Refresh button – click to refresh the Feature/Variant model tree inside the Tree-view.
4. Refers to Expand button – click to expand the entire tree inside the Tree-view.

5. Refers to Collapse button – click to collapse the tree rendered inside the Tree-view.
6. Refers to Show Preview button – click to enable the preview for visualizing variability Information; available in the Document View and supports Wiki format only for fields with the type WikiText.
7. Refers to Reset Preview button – click to disable the Preview.
8. Refers to Error Check button – click to open the Error Check view, to see the errors in PVSCL rules.
9. Refers to Calculations button – click to open the Calculations page, so to edit calculations present inside the fields of a tracker item.
10. Refers to Restriction button – click to open the Restriction page, so to edit the restriction inside the pvRestriction fields of a tracker item.
11. Refers to Settings button – click to navigate to the Settings page so to configure the General settings, Calculations specific settings, and, also to see the Integration specific information.

Below the menu bar, there is the VDM Selector dropdown that lists all the variant models attached to the selected configuration space. On selecting any of the variant model from the dropdown, the model will be rendered inside the Tree-view. The Tree-view lists the selected Feature/Variant model(s).

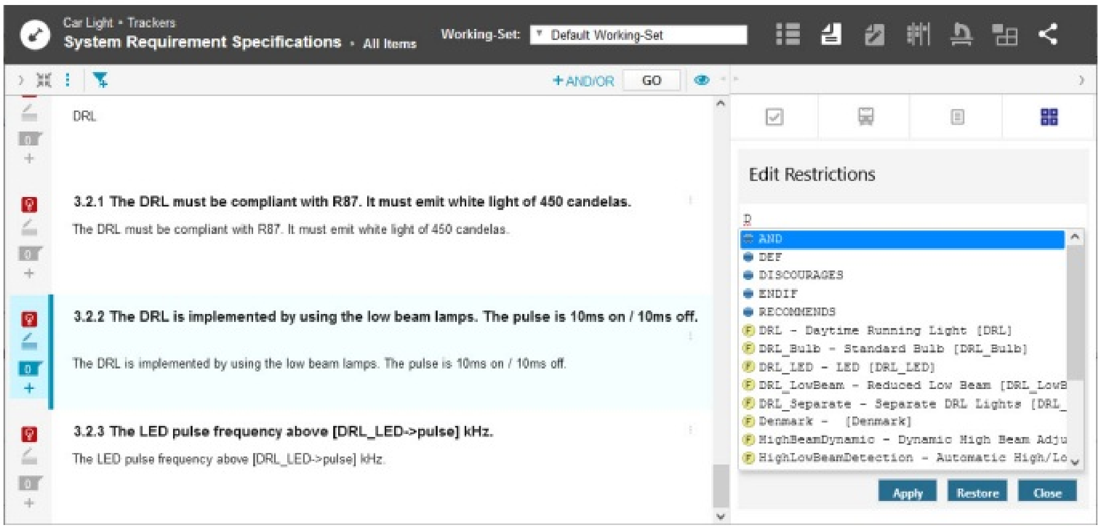
Note:

- The button for Error Check is disabled in case the incorrect DesktopHub or WebHub version is used (see Chapter ‘Software Requirements’).
- During preview vertical scrolling of the main document section is supported to enable review of the document.

Working with Restriction Editor

The Restriction Editor can be opened by clicking the Restrictions icon. Edit a restrictionin Restriction Editor by selecting an item in a tracker. The Restriction Editor provides the ability of auto completion proposals and syntax highlighting while editing restrictions for tracker items

Figure 17. Working with the Restriction Editor



DRL

3.2.1 The DRL must be compliant with R87. It must emit white light of 450 candelas.
The DRL must be compliant with R87. It must emit white light of 450 candelas.

3.2.2 The DRL is implemented by using the low beam lamps. The pulse is 10ms on / 10ms off.
The DRL is implemented by using the low beam lamps. The pulse is 10ms on / 10ms off.

3.2.3 The LED pulse frequency above [DRL_LED->pulse] kHz.
The LED pulse frequency above [DRL_LED->pulse] kHz.

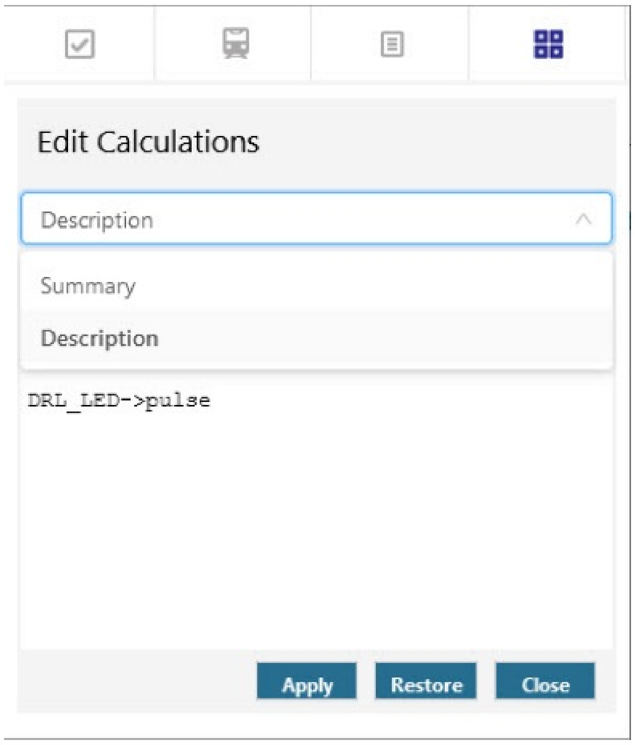
```
D
AND
DEF
DISCOURAGES
ENDIF
RECOMMENDS
DRL - Daytime Running Light [DRL]
DRL_Bulb - Standard Bulb [DRL_Bulb]
DRL_LED - LED [DRL_LED]
DRL_LowBeam - Reduced Low Beam [DRL_LowB
DRL_Separate - Separate DRL Lights [DRL_
Denmark -  [Denmark]
HighBeamDynamic - Dynamic High Beam Adju
HighLowBeamDetection - Automatic High/Lo
```

Working with Calculations Editor

The Calculations Editor can be used to edit the calculations present in the fields of a tracker item. You can open it by clicking the Calculations icon. Calculations can be edited by selecting an item in a tracker and then, in the

Calculations Editor, elect the field of an item that contains the calculation markers. After selecting a field, all the calculations in that field appear in the list below. Select a calculation from the list and edit it in the editor below. Calculations Editor supports auto-completion of proposals and syntax highlighting while editing calculations.

Figure 18. Calculations Editor of the Integration

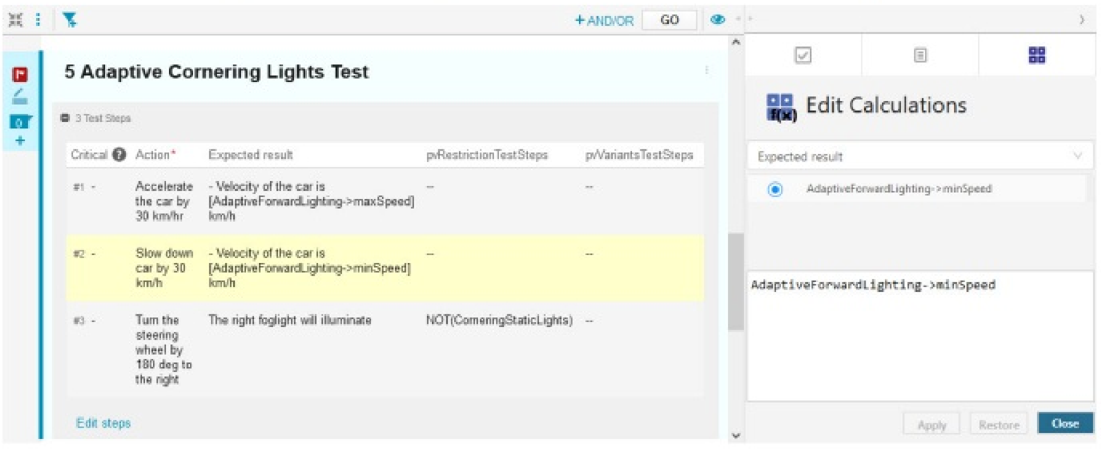


```
DRL_LED->pulse
```

Working with Test Steps

Both the restriction and the Calculation Editor support Test Steps as listed inside Test Cases in the Document View of codebeamer. Here, a concrete row representing a Test Step needs to be selected in the expanded Tests Steps table, and so the data of this Test Step will appear in the editors. Restrictions will be added to the “pvRestrictionsTestSteps” field, while calculations will be added at the locations indicated by substitution markers inside supported fields, as shown in the example below (Figure 19, “Editing Calculations Inside Test Steps”).

Figure 19. Editing Calculations Inside Test Steps



5 Adaptive Cornering Lights Test

3 Test Steps

Critical	Action*	Expected result	pvRestrictionTestSteps	pvVariantsTestSteps
#1 -	Accelerate the car by 30 km/hr	- Velocity of the car is [AdaptiveForwardLighting->maxSpeed] km/h	--	--
#2 -	Slow down car by 30 km/h	- Velocity of the car is [AdaptiveForwardLighting->minSpeed] km/h	--	--
#3 -	Turn the steering wheel by 180 deg to the right	The right foglight will illuminate	NOT(CorneringStaticLights)	--

Edit steps

● AdaptiveForwardLighting->minSpeed

```
AdaptiveForwardLighting->minSpeed
```

Error Check

Added variability information can be checked for errors using the Error Check function. Errors in pvSCL scripts are

reported if the script’s syntax is not pvSCL compliant, or if an element is unknown based on the loaded pure::variants models.
The problems are displayed in a list containing following information:

- Message: Description of the error.
- pvSCL: The pvSCL expression containing the error.
- Field: The name of the field that contains the error.
- Item Link: URI of the affected item

Severity of the problem is indicated by an icon (error or warning).
Note: To check with hierarchical feature model structures containing variant instances, a variant model also needs to be selected for proper evaluation

Variability in WIKI-Tables

To add variability to WIKI-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 they need to contain the specified keyword, which is also used to indicate a restriction, on e.g. a requirement. By default, this keyword is pv Restriction.

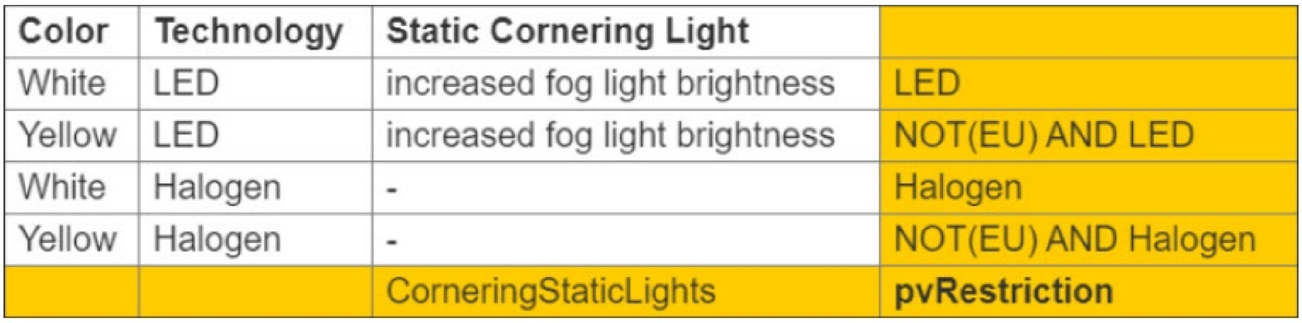


Figure 20. Example WIKI-Table

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 values 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) are removed from the variant by default.

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

Restrictions

Limitations Relating Text Substitution in WikiText Fields

Since the set of pure::variants substitution marker characters can conflict with WikiText special characters (e.g. ‘[…]’ defines a WikiText hyperlink), the pure::variants text substitution processing looks for the WikiText-escaped form of those special characters (so e.g. ‘~[‘ and ‘~]’ for the default markers ‘[‘ and ‘]’). This form is created in most cases by the codebeamer WikiText Rich Text editor when adding these characters there. Also the integration widget functionality will use this escaped form.

The usage of the codebeamer WikiText RichText editor and the WikiText syntax restrictions in general will result in limitations of the usage of text substitutions within WikiText content:

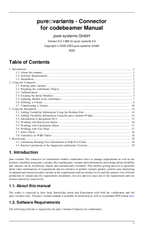
- Text substitution sections are only supported where formatted text can be used within WikiText content. So it is for example not supported to add substitutions in WikiText control sequences or in the target URL part of a WikiText hyperlink.
- Since the text sequence '${…}' has special meaning in codebeamer, it is not recommended to use '{' and '}' as substitution open and close markers and '$' as substitution escape marker.
- It is not supported to use text formatting within or across text substitution section boundaries. This can result in invalid pvSCL expressions or in the creation of invalid WikiText content during transformation. The only exception is the usage of formatting within a pvSCL text literal.
- In codebeamer, the item description field format can be switched from WikiText to plain text for each single item. The usage of text substitutions in such plain text descriptions is not supported.

Known Limitations of the Supported codebeamer Versions

In this section known issues of codebeamer are listed, which cause limitations of the functionality of the pure::variants Connector for codebeamer:

- During update of Test Set trackers in variant working-sets, for referenced test cases, which are are not part of the variant, test case references to the 150% Test Case tracker are falsely added.
- Updating of variant working-sets based on certain source baseline combinations on tracker level are falsely declined. So, if at least for one tracker the HEAD baseline is selected and for at least two trackers the same baseline is selected, code beamer will cancel the update with an error message 'Duplicate key …'.
- Tracker baselines, i.e., baselines created for a single tracker, are always created on the master branch of the default working-set. So the usage of tracker baselines in context of a non-default working-set will not work.
  So it is recommended to use instead project baselines to define the state of the source working-set to be used for transformation

Documents / Resources

	pure-systems pure variants Connector for codebeamer [pdf] User Manual pure variants, Connector for codebeamer, pure variants Connector for codebeamer, pure varian ts Connector, Connector

pure::variants - Connector for codebeamer Manual

pure-systems GmbH

Version 6.0.1.685 for pure::variants 6.0

Copyright © 2003-2023 pure-systems GmbH

2023

Table of Contents

1. Introduction 1
1.1. About this manual 1
1.2. Software Requirements 1
1.3. Installation 2
2. Using the Connector 2
2.1. Starting pure::variants 2
2.2. Preparing the codebeamer Project 2
2.3. Authentication 2
2.4. Creating the Initial Model(s) 3
2.5. Updating Models from codebeamer 7
2.6. Defining a Variant 8
2.7. Transforming a Variant 10
3. Using the Integration 14
3.1. Adding Variability Information Using the Desktop Hub 14
3.2. Adding Variability Information Using the pure::variants Widget 14
3.3. Introduction to Integration GUI 15
3.4. Working with Restriction Editor 16
3.5. Working with Calculations Editor 17
3.6. Working with Test Steps 17
3.7. Error Check 18
3.8. Variability in WIKI-Tables 18
4. Restrictions 18
4.1. Limitations Relating Text Substitution in WikiText Fields 18
4.2. Known Limitations of the Supported codebeamer Versions 19

1. Introduction

pure::variants The connector for codebeamer enables codebeamer users to manage requirements as well as test artefacts variability using pure::variants. By coupling pure::variants and codebeamers, knowledge about variability and variants can be formalized, shared, and automatically evaluated. This enables getting answers to questions about valid combinations of requirements and test artefacts in product variants quickly; permits easy monitoring of planned and released product variants at the requirements and test artefact level, and also permits very efficient production of variant-specific requirements documents, test sets and test cases out of the requirements and test artefact repository respectively.

1.1. About this manual

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

1.2. Software Requirements

The following software is required by the pure::variants Connector for codebeamer:

1