
pure::variants Setup Guide

pure-systems GmbH

Version 5.0.11.369 for pure::variants 5.0

Copyright © 2003-2022 pure-systems GmbH

2022

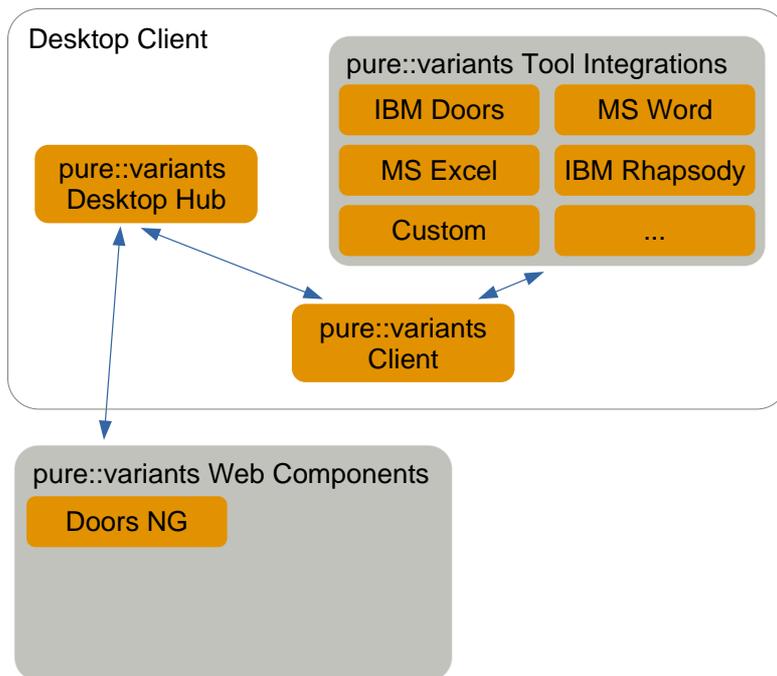
Table of Contents

1. Introduction	2
2. System Requirements	3
2.1. Eclipse based Client	3
3. pure::variants Client	4
3.1. Install pure::variants Client	4
3.1.1. Install with pure::variants Installer	4
3.1.2. Install into an existing Eclipse	9
3.2. Update pure::variants Client	13
3.2.1. Update with pure::variants Installer	13
3.2.2. Update with Update Action	16
3.2.3. Update with Eclipse package manager	18
3.3. Uninstall pure::variants Client	21
3.3.1. Uninstall using pure::variants Uninstaller	21
3.3.2. Uninstall pure::variants from existing Eclipse instance	22
3.4. Basic Setup of the pure::variants Client	25
3.4.1. Setup a pure::variants Client License	25
3.4.2. Update a pure::variants Client License	26
3.4.3. Add pure::variants Client License using environment variable or Java property	26
3.5. Trouble Shooting	27
3.5.1. pure::variants is low on memory	27
4. pure::variants Connectors	27
4.1. Installation of pure::variants Connectors	27
4.2. pure::variants Connector for Capella	27
4.3. pure::variants Connector for Team Foundation Server	28
4.4. pure::variants Connector for PTC Integrity	28
4.4.1. Add additional Fields for pure::variants	28
4.4.2. Change Connector and In-Tool Integration Settings	30
4.4.3. Change Fields Copied for Variant Creation	32
4.4.4. Enable PTC Integrity Client Access	32
4.5. Connector for IBM Rational Rhapsody	33
4.5.1. Preparing IBM Rational Team Concert	33
4.5.2. Preparing pure::variants	33
4.6. Connector for Codebeamer	34
4.6.1. Installation of pure::variants Client	34
4.6.2. Installation of Server Component and pure::variants Widget to Codebeamer	34
4.6.3. Installation without running in a docker container	34
4.6.4. Installation in a docker image	35
4.6.5. Permissions	35
4.6.6. Getting Version Information of the Server Component	35
4.6.7. Configuration To Enable Open ID Connect (OIDC) Authentication	35
4.6.8. docker-compose.yml for the NGNIX Proxy	36
4.6.9. oidc-auth-proxy.dockerfile for the NGNIX Proxy	37
4.6.10. oidc-auth-proxy-nginx.conf for the NGNIX Proxy	37
4.6.11. Steps to Setup a Docker-container the NGNIX Proxy	38
5. pure::variants Tool Integrations	38

5.1. Install pure::variants Tool Integrations	38
5.1.1. pure::variants Desktop Hub	39
5.1.2. pure::variants Integration for Doors	39
5.1.3. pure::variants Integration for PTC Integrity	40
5.1.4. pure::variants Integration for IBM Rational Rhapsody	41
5.1.5. pure::variants Integration for Enterprise Architect	41
5.1.6. pure::variants Integration for Microsoft Office	41
5.1.7. pure::variants Integration for Team Foundation Server	42
5.1.8. Advanced Integration Setup	42
5.2. Update pure::variants Tool Integrations	45
5.3. Uninstall pure::variants Tool Integrations	45
5.4. Basic Setup of pure::variants Tool Integrations	46
5.4.1. Server Connection Setup	47
6. pure::variants Web Integration	50
6.1. IBM Rational DOORS NG Web Integration	50
6.1.1. Requirements for pure::variants Integration Deployment	50
6.1.2. Installation on Apache Tomcat	51
6.1.3. Installation on Websphere Liberty	52
6.1.4. Uninstall the pure::variants Integration for DOORS NG	53
6.1.5. Administrative Setup of the pure::variants Integration for DOORS NG	53
6.1.6. Add pure::variants Integration to DOORS NG	56
6.1.7. Check-up list for a successful deployment	57

1. Introduction

Depending on the engineering tool landscape in place and on the desired data management mode for pure::variants models, the pure::variants setup consists of some or all of the components depicted as orange boxes in figure [Figure 1, “The Big Picture”](#). In this setup guide all components are described that have to be installed, updated, or configured to cover the different possible deployment scenarios. The components are grouped from IT perspective into components running on a server, and components running on the desktop clients. In one deployment scenario, only desktop client components depicted in the upper left might be needed, while in another scenario server components combined with a browser running on client side might be needed. There are also some boxes visualized in blue, e.g. Browser and LDAP, that are not delivered by pure-systems but interact with pure::variants components in certain deployment scenarios. In the following sections each of the orange boxes are explained from the perspective of administration and setup.

Figure 1. The Big Picture

The manual is available in online help inside the installed product as well as in printable PDF format. Get the PDF [here](#).

2. System Requirements

pure::variants has different system requirements, depending on the part of the software going to be installed. The following lists the system requirements for all parts of the software.

2.1. Eclipse based Client

- Operating System
 - Windows 7, 8, 10
 - Windows Server 2003, 2008, 2012, 2016, 2019
 - Linux 64 Bit with X11 Window System installed
 - Mac OS
- Software
 - Oracle Java SE or OpenJDK
 - Supported Java versions:
 - Java 8, 9, 10, 11, 12, 13, 14, 15, 16, 17
 - The Java compatibility is tested with the official Java Standard Edition provided by Oracle (<https://www.java.com/en/download/>) and the OpenJDK provided by Oracle (<https://jdk.java.net/archive/>).

- Eclipse 3.8.0 – 4.23
- Memory
 - Min: 2 GB
 - Recommended: 8 GB
- CPU
 - Min: Dual Core CPU
 - Recommended: 4 CPU cores
- HDD
 - Min: 10 GB free disk space

3. pure::variants Client

pure::variants can be installed using the pure::variants installer as a stand-alone application or it can be installed into an existing Eclipse based tool chain. For both ways to install pure::variants we recommend to use the pure::variants installer.

The pure::variants installer is available for Windows only. If the operating system platform is Linux or MacOS X, pure::variants needs to be installed into an existing Eclipse instance. See [Section 3.1.2, “Install into an existing Eclipse”](#).

In case of very strict firewalls or no network access on the installation machine either install pure::variants as a stand-alone application. ([Section 3.1.1, “Install with pure::variants Installer”](#)) or install pure::variants into an existing Eclipse instance using an update site. ([the section called “Using update site”](#)). These installation methods allow you to first download the installation packages and install pure::variants afterwards.

The installation procedures are described below. Once the initial installation has finished, installation of a license is required to use pure::variants. See following section for more information on license installation.

3.1. Install pure::variants Client

3.1.1. Install with pure::variants Installer

This installation method is available for Windows only. If you do not use Windows please see [Section 3.1.2, “Install into an existing Eclipse”](#).

To be able to successfully install the pure::variants client you need to following:

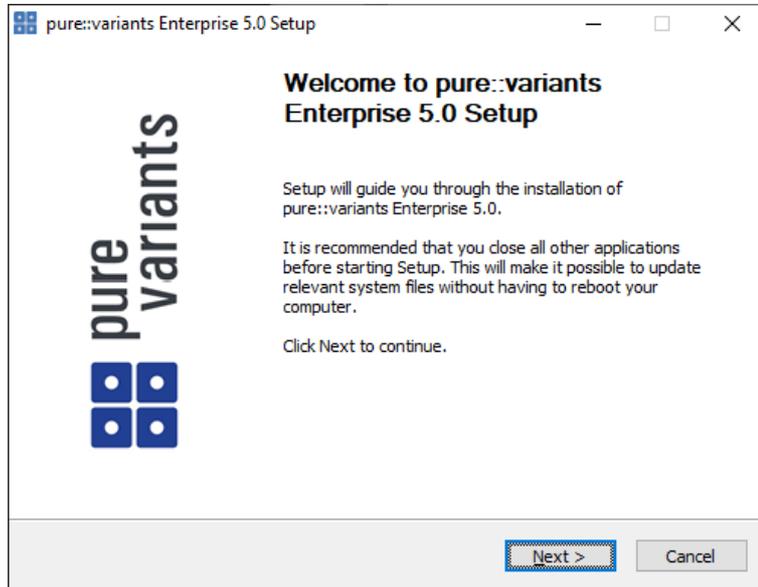
- pure::variants client license
 - If a floating license is used additionally the license server URL is needed to be able to connect and obtain a license form the license server.
- pure::variants client installer or pure::variants update site
- supported 64-Bit Java version installed

The Windows Installer can be downloaded from the pure::variants product web page. Go to <https://www.pure-systems.com/pvde-update>. The product download pages are protected by a password. You need to login by using the email address and the registration number from the license file.

Download the installer package ("pure::variants Windows Installer Package") and extract it. The installer will set up a fresh Eclipse with pure::variants and documentation. Start the installation by double-clicking "Setup Enterprise X.Y.ZZ.exe". Running the pure::variants enterprise installer requires Administrator privileges.

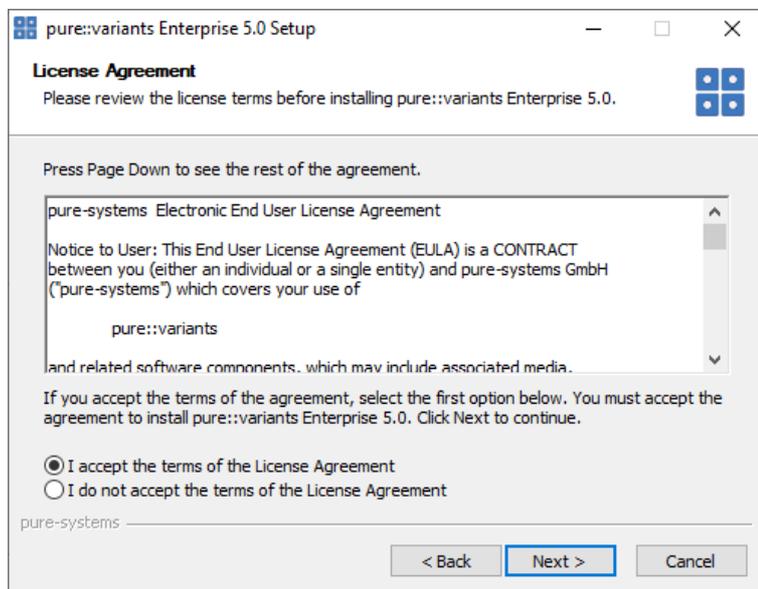
All pure::variants extensions available for the account are automatically included in the Windows Installer download. However, some may not be enabled by default in Installer. Make sure to select the desired extensions during the installation process. Later updates to the extension selection can be done either by reinstalling pure::variants or by following the alternatives described in [the section called “Using update site”](#).

Figure 2. pure::variants Client Installer



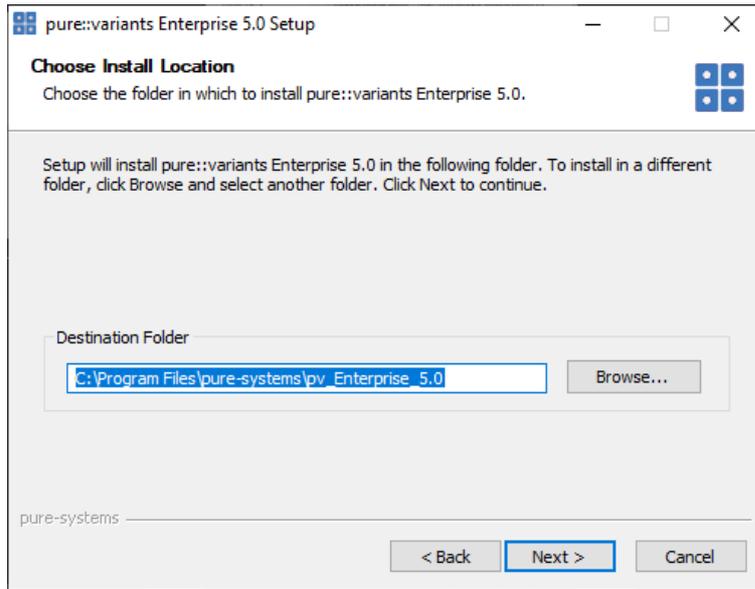
Click *Next*.

Figure 3. Setup pure::variants Client License



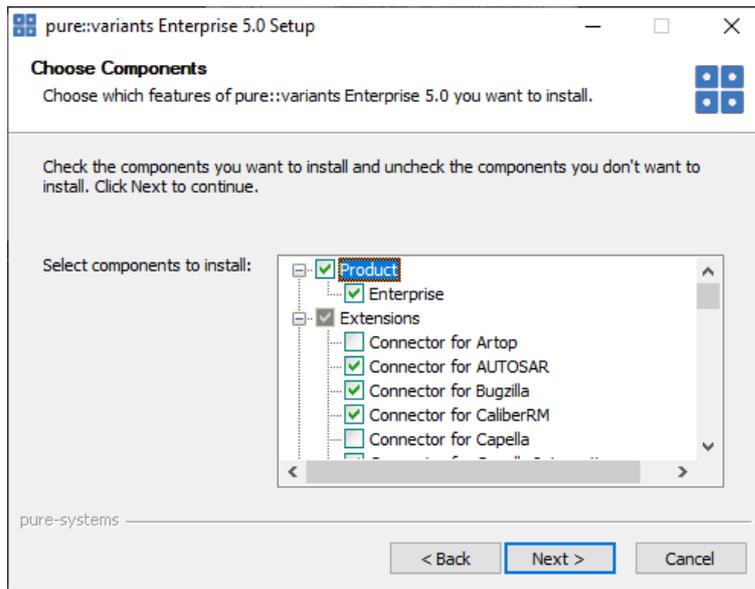
Read the license agreement and after accepting it click *Next*.

Figure 4. Setup pure::variants Client Installation Location

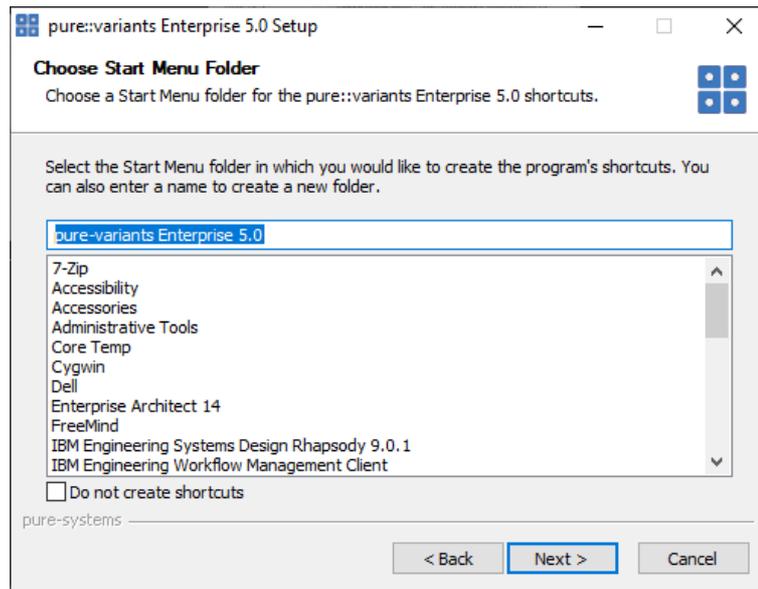


Select the folder where to install the pure::variants client files. Click *Next*.

Figure 5. pure::variants Client Feature Selection



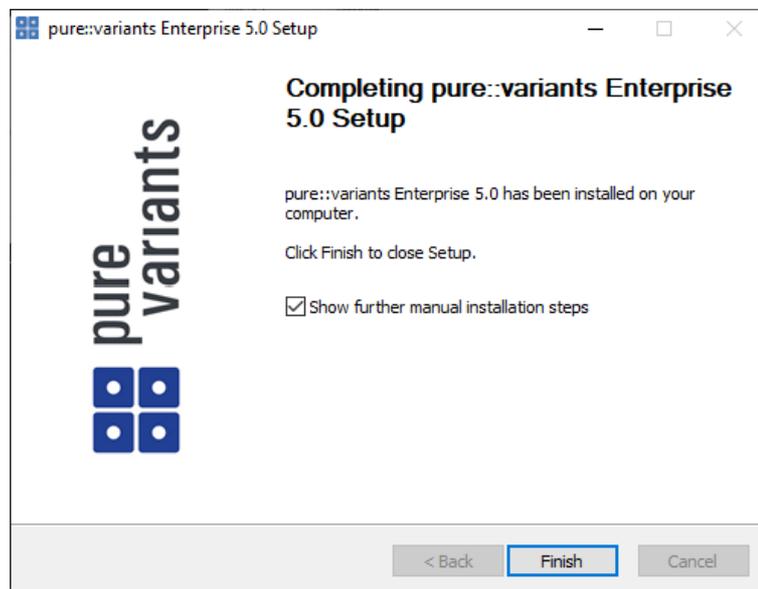
Select the connectors which shall be installed with the pure::variants client. Click *Next* after the feature selection is complete.

Figure 6. Setup pure::variants Client Start Menu

Enter the name for the Windows start menu entry, or disable the creation of the start menu entry. Click *Next*

The next pages may show information about pure::variants integrations, which are installed along with the pure::variants client. If no connector was selected providing an integration, this page will show the *Install* button.

Click *Install* to start the installation process.

Figure 7. Setup pure::variants Client Finish Page

The Option *Show further manual installation steps* will open a text document showing more information about the installed integrations and possible manual installation steps, which have to be performed for the integrations to work properly.

pure::variants Enterprise Installer Command Line Options

The pure::variants installer provides the following command line options:

Table 1. pure::variants Installer Command Line Options

Option	Description
/S	Run the installation in silent mode. No installation dialog is opened. Automatically installs the default selected software packages, or all if used together with option /ALL.
/UPDATE /S	To update an existing installation in silent mode. Same as silent installation, no dialog is opened.
start "" /WAIT "Setup.exe" /UPDATE /S	To update an existing installation in silent mode. Same as silent installation, no dialog is opened, but ensures installer not running in background.
/ALL	Select all packages for installation.
/NODOTNET	Skip installation of the .NET 4 Framework.
/NOINTCOMP	Skip installation of the integration components for Java & .NET.
/JAVA	Location of the Java executable to be used for the installation. Example: /JAVA="C:\Program Files\Java\jre6\bin\java.exe"
/ECLIPSE	Path to an existing Eclipse installation into which to install pure::variants as a feature, instead of installing pure::variants as a stand-alone application. This directory must contain the file eclipse.exe. Example: /ECLIPSE="C:\Program Files\Eclipse 3.8\eclipse"
/D	Path to the directory where to install the pure::variants stand-alone application. Must be the last option on the command line and must not contain any quotes, even if the path contains spaces. Example: /D=C:\Program Files\pure-variants

Example commandline with JAVA path:

```
"D:\5.x.x\pure-variants Setup 5.x.x\Setup Enterprise 5.x.x.exe" /JAVA="C:\Program Files\Java\jre1.8.0_231\bin\java.exe"
```

Install pure::variants in silent mode

The pure::variants Client installer has a silent mode. This mode installs the pure::variants client without user interaction by just using the standard settings of the pure::variants client installer also considering further options on the command line.

To do this, call the installer with command line option /S. See [the section called “pure::variants Enterprise Installer Command Line Options”](#) for all available command line options.

Update pure::variants in silent mode

It is also possible to run the update in background or silent mode to update an existing installation. Note that there will be no console output as well.

To do this, call the installer with command line option /UPDATE /S. To run it in silent mode but not in background `start "" /WAIT "Setup.exe" /UPDATE /S` can be used. See [the section called “pure::variants Enterprise Installer Command Line Options”](#) for all available command line options.

3.1.2. Install into an existing Eclipse

pure::variant can be installed into an existing Eclipse based tool chain. To install pure::variants, the pure::variants installer package downloaded from the pure::variants updatesite can be used. We recommend this for all Windows users.

Alternatively the pure::variants update site can be used directly with the Eclipse client. You can also download an archived update site from the pure::variants update site and use this with the Eclipse client (See [the section called "Using update site"](#)).

Installation Requirements

pure::variants needs to following features to already be installed in the target Eclipse, or the Eclipse instance has to have access to the Eclipse release update site.

- JavaScript Development Tools
 - org.eclipse.wst.jsdt.feature.feature.group
- Eclipse Business Intelligence and Reporting Tools (BIRT)
 - org.eclipse.birt.feature.group
- Graphical Modeling Framework
 - org.eclipse.gmf.feature.group

Using pure::variants Installer

The installation into an existing Eclipse instance is done the same way as installing pure::variants as stand-alone application (See [Section 3.1.1, "Install with pure::variants Installer"](#)).

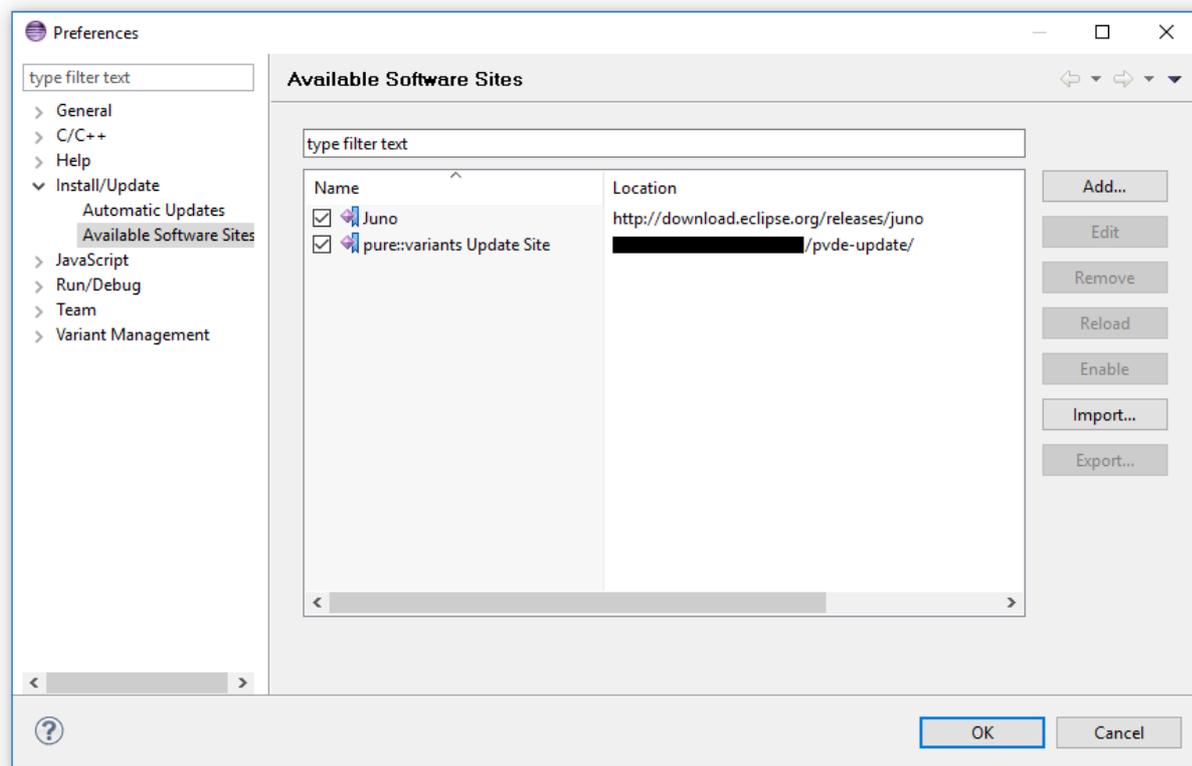
There is one difference: the target Eclipse has to be defined with the */ECLIPSE* command line option.

Using update site

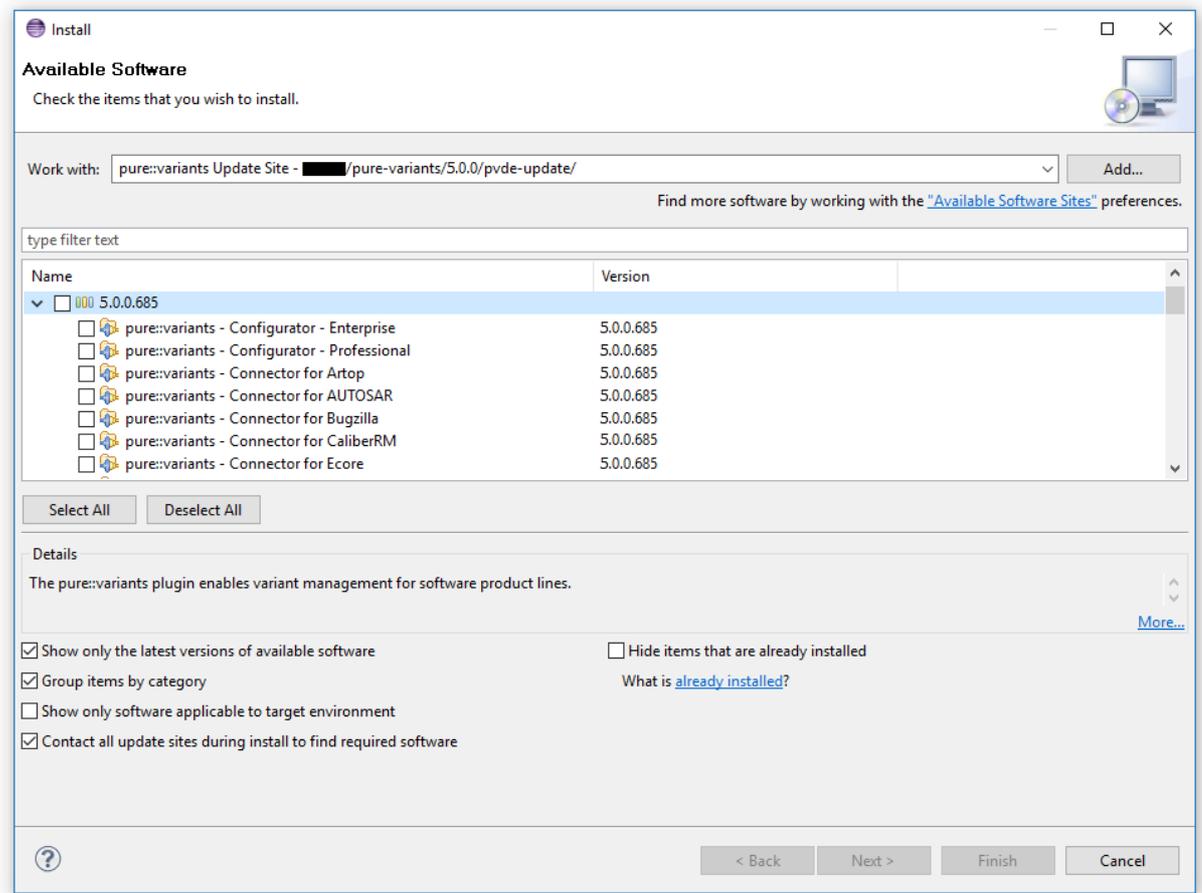
- Start pure::variants (or the Eclipse into which pure::variants has been installed).
- Select "Help"->"Install New Software...".
- Select "pure::variants update site" from the available Software Sites.

If location "pure::variants update site" is not present, enter your location in the edit field, or press "Add" if you have a local update site at hand.

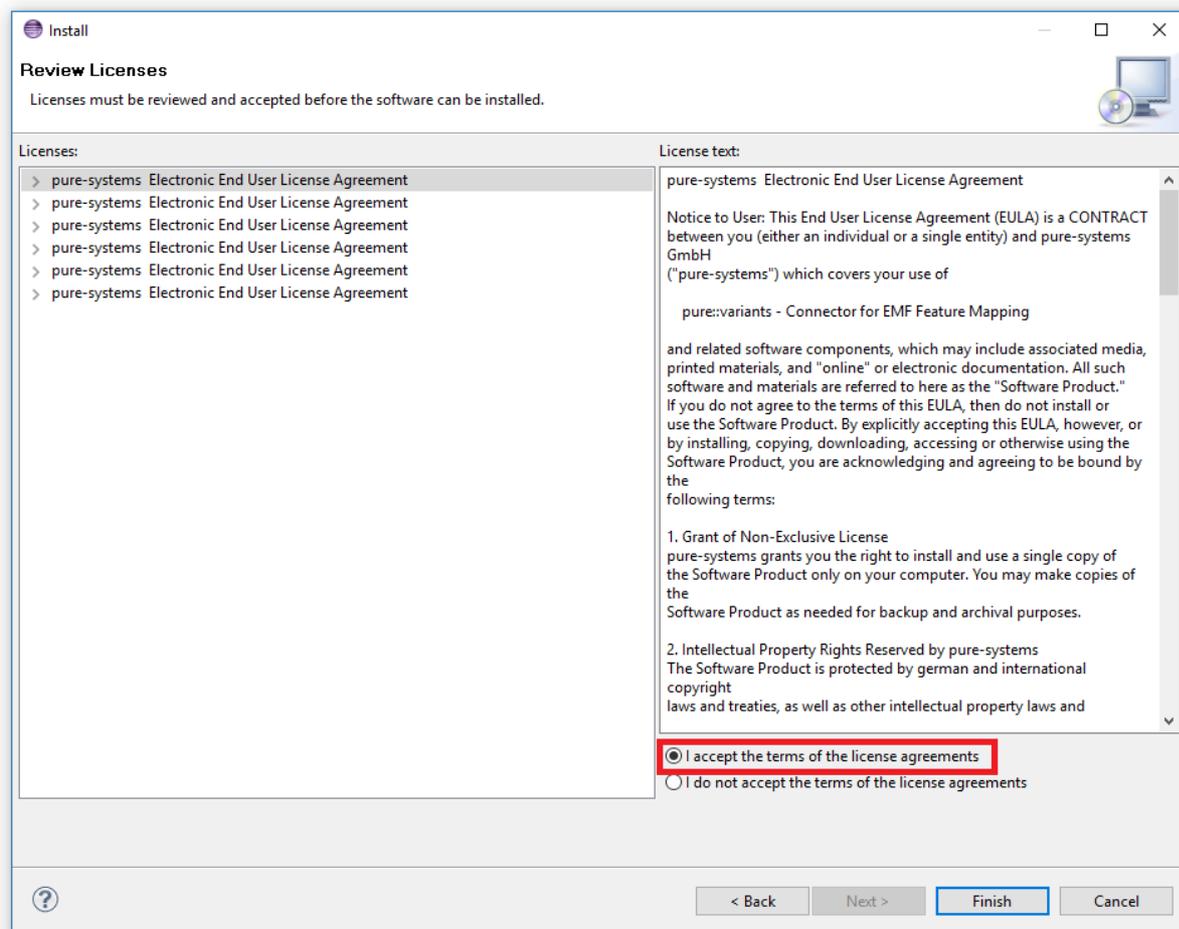
The location of the site depends on the pure::variants product variant. Visit the pure-systems web site (<https://www.pure-systems.com>) or read your registration email to find out which site is relevant for the version of the software you are using.

Figure 8. Update Site Selection

- Unfold the pure::variants update site and select all features to be updated. Select "Next".

Figure 9. Pure::variants Plugin Selection

- Accept license, hit "Next" and then "Finish".

Figure 10. Licence Agreement

- In the dialog select "Install all".
- Restart pure::variants when asked for.

If the direct remote update is not possible (often due to firewall/proxies preventing Eclipse accessing external web sites), please go to the web site using an Internet browser:

- For pure::variants Evaluation use <https://www.pure-systems.com/pv-update>
- For pure::variants Enterprise use <https://www.pure-systems.com/pvde-update>

and download the "Complete Updatesite" archive:

- Start pure::variants (or the Eclipse into which pure::variants has been installed).
- Select "Help"->"Software Updates"->"Find and Install...".
- Select "Search for new features to install" and "Next".
- Click on button "Archived Update Site" or "Local Update Site".
- Use "Browse" to select the downloaded archive file.
- Press "Ok". The pure::variants update site from the archive should be selected.
- All other check boxes should be unselected to speed up the process. Press "Finish".

- Unfold everything below pure::variants update site and select all features to be updated. Select "Next".
- Accept license, hit "Next" and then "Finish".
- In the dialog, select "Install all".
- Restart pure::variants when asked for.

3.2. Update pure::variants Client

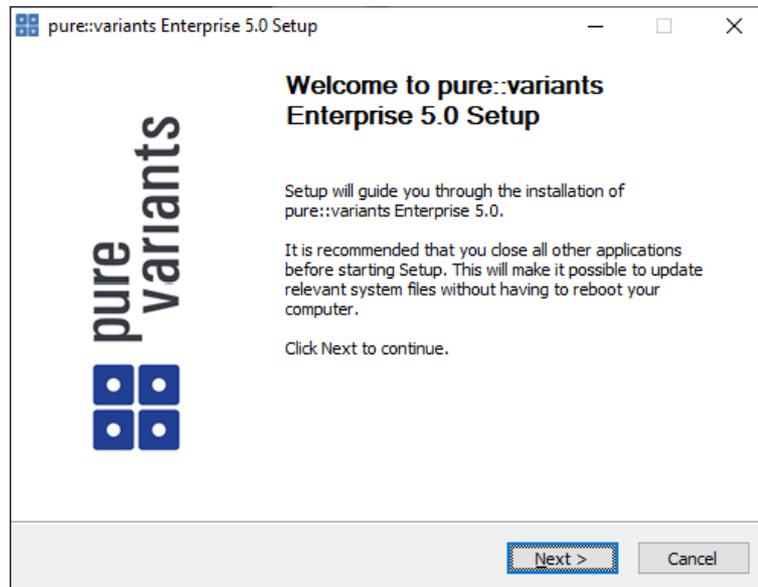
3.2.1. Update with pure::variants Installer

This update method is available for Windows only. If you do not use Windows please see [Section 3.2.2, "Update with Update Action"](#) or [Section 3.2.3, "Update with Eclipse package manager"](#).

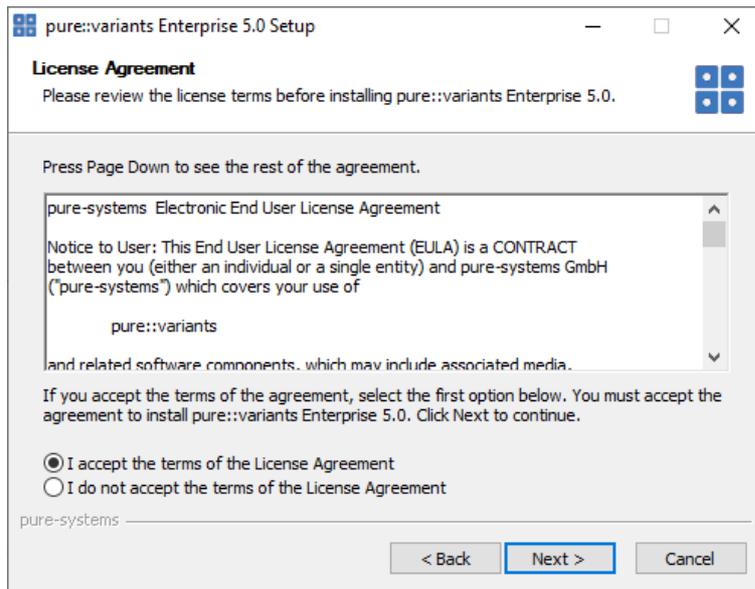
The Windows Installer can be downloaded from the pure::variants product web page. Go to <https://www.pure-systems.com/pvde-update>. The product download pages are protected by a password. You need to login using the email address and the registration number from the license file.

Download the installer package ("pure::variants Windows Installer Package") and extract it. The installer will check for an existing pure::variants client installation and start in update mode if it finds one. Start the update by double-clicking "Setup Enterprise X.Y.ZZ.exe". Running the pure::variants enterprise installer requires administrator privileges.

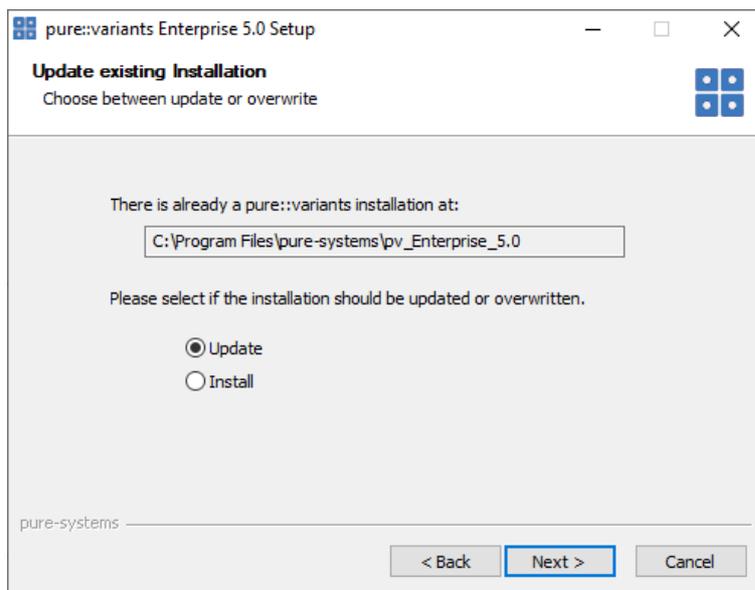
Figure 11. pure::variants Client Installer



Click *Next*.

Figure 12. Setup pure::variants Client License

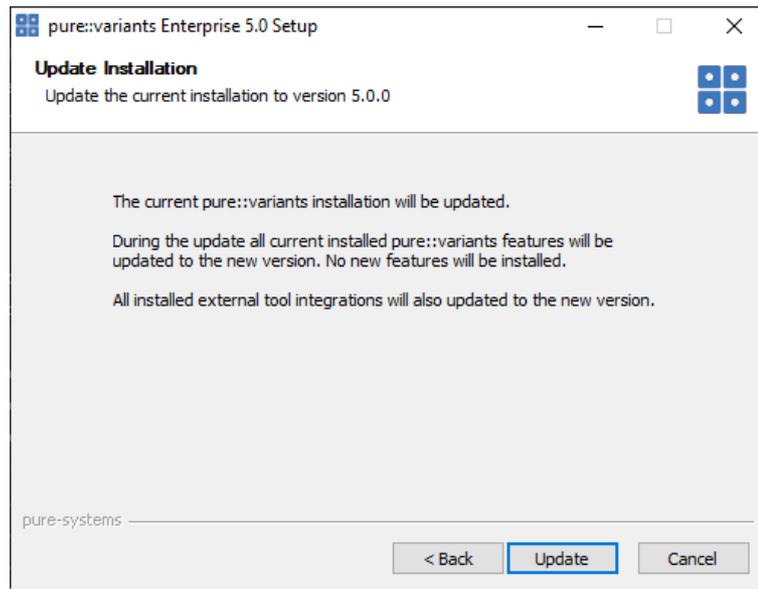
Read the license agreement, and after accepting it click *Next*.

Figure 13. Choose Update Mode

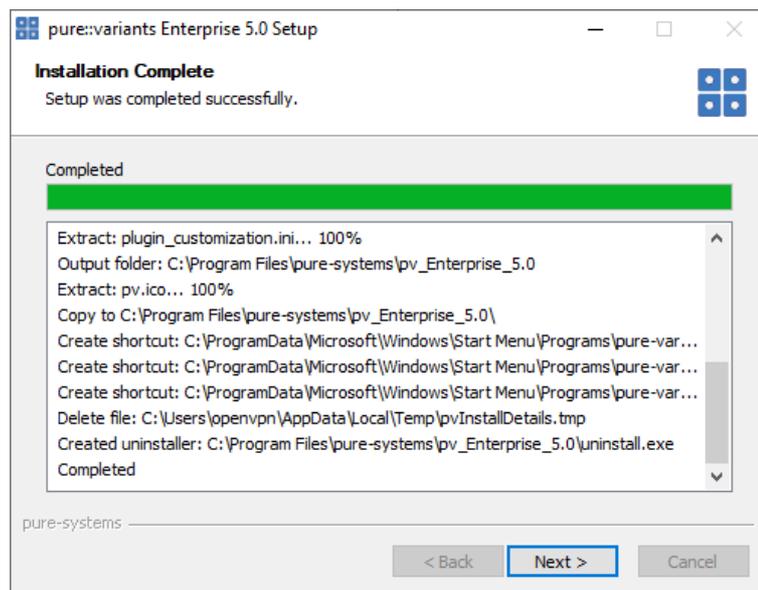
Choose *Update* if the current pure::variants client installation shall just be updated with the same installed feature and settings. The installed pure::variants integrations will also be updated. The installed components cannot be changed. If a change of the installed components is wanted, choose *Install* mode.

Or choose *Install* if the current pure::variants installation shall be removed and a new fresh pure::variants client shall be installed. The *Install* option runs the installer as described in [Section 3.1.1, "Install with pure::variants Installer"](#). Please see this section for further installation steps.

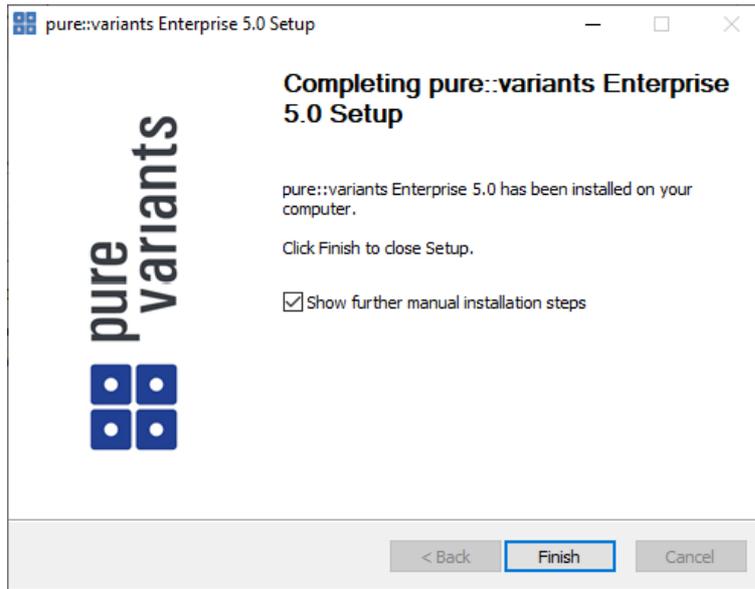
Click *Next*.

Figure 14. pure::variants Start Update

Click *Update* to start the update process.

Figure 15. pure::variants Installation Progress

This page is showing the installation details. Click *Next* after this is finished.

Figure 16. Update pure::variants Client Finish Page

The Option *Show further manual installation steps* will open a text document showing more information about the installed integrations and possible manual installation steps, which have to be performed for the integrations to work properly.

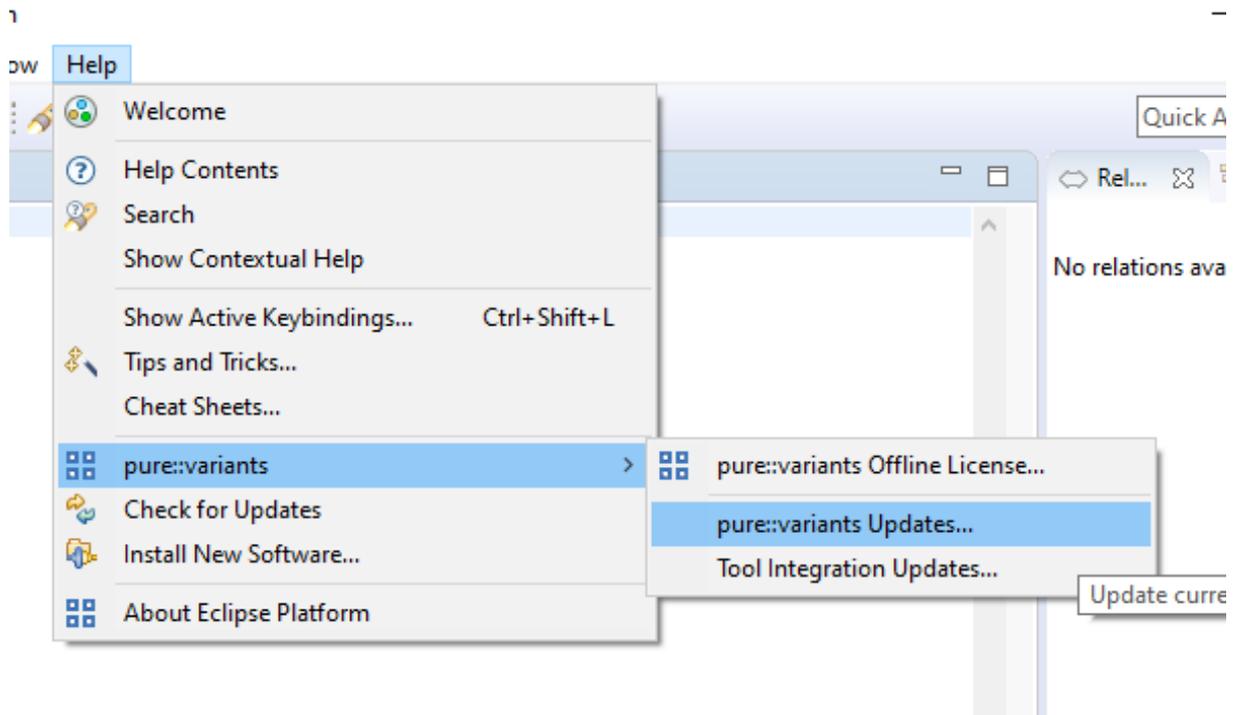
3.2.2. Update with Update Action

pure::variants has a built-in update action which can be used to perform an update with all the currently installed pure::variants extensions. This update action does not update the installed pure::variants integrations automatically. But they can be easily updated with the *Tool Integration Update* action. See [Section 5.2, “Update pure::variants Tool Integrations”](#) for the detailed description.

The update action requires administrator privileges.

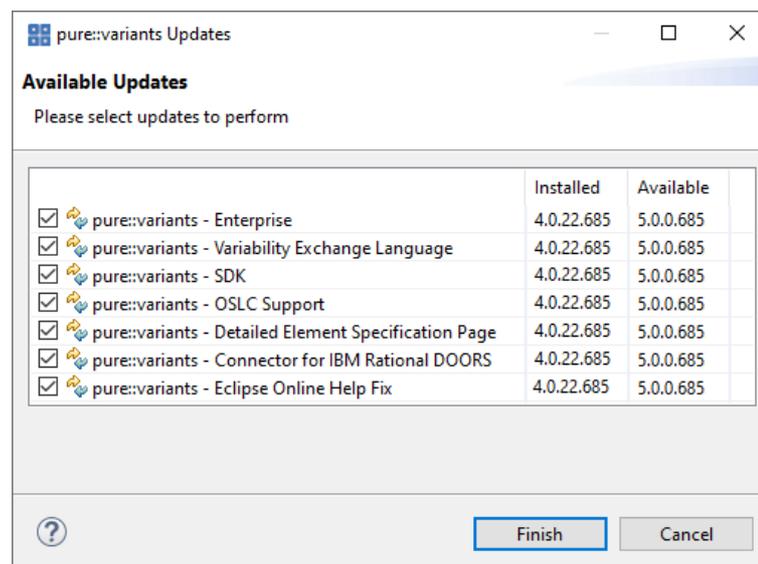
Note

The pure::variants client restarts automatically after the update process finished. So please make sure that all open editors are saved and closed before continuing.

Figure 17. Start pure::variants Client Update

Start the pure::variants client update with the *pure::variants Updates...* action from the *pure::variants Help* menu. The action can be found in the *pure::variants* sub-menu.

If the pure::variants client is not started as Administrator, a dialog comes up to inform that pure::variants has to be started as Administrator.

Figure 18. Start pure::variants Client Update

A dialog comes up and shows all available updates. Select the features to update and click *Finish*. The update process starts and shows the progress in the same window.

Figure 19. Start pure::variants Client Update

After the update process finished, the pure::variants client restarts automatically.

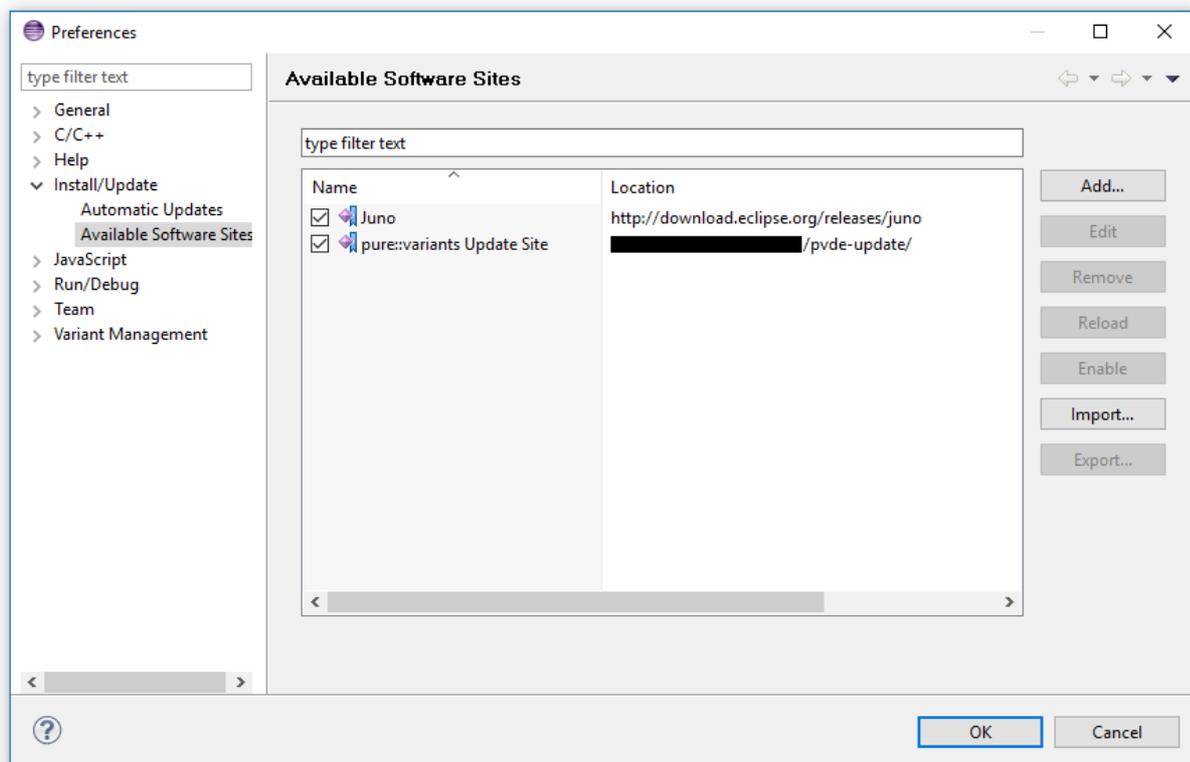
3.2.3. Update with Eclipse package manager

The quickest way to get a update for pure::variants is to run the software updater inside pure::variants:

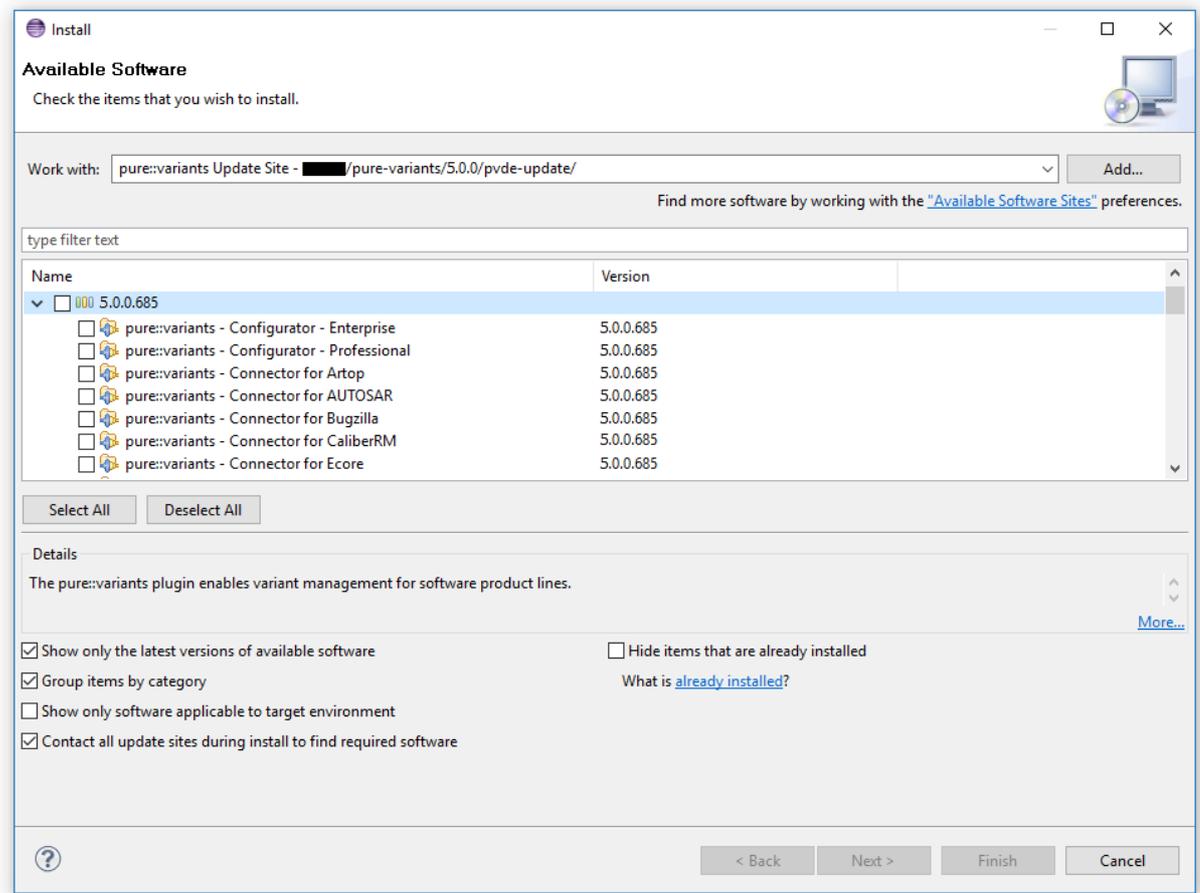
- Start pure::variants (or the Eclipse into which pure::variants has been installed).
- Select "Help"->"Install New Software...".
- Select "pure::variants update site" from the available Software Sites.

If location "pure::variants update site" is not present, enter your location in the edit field, or press "Add" if you have a local update site at hand.

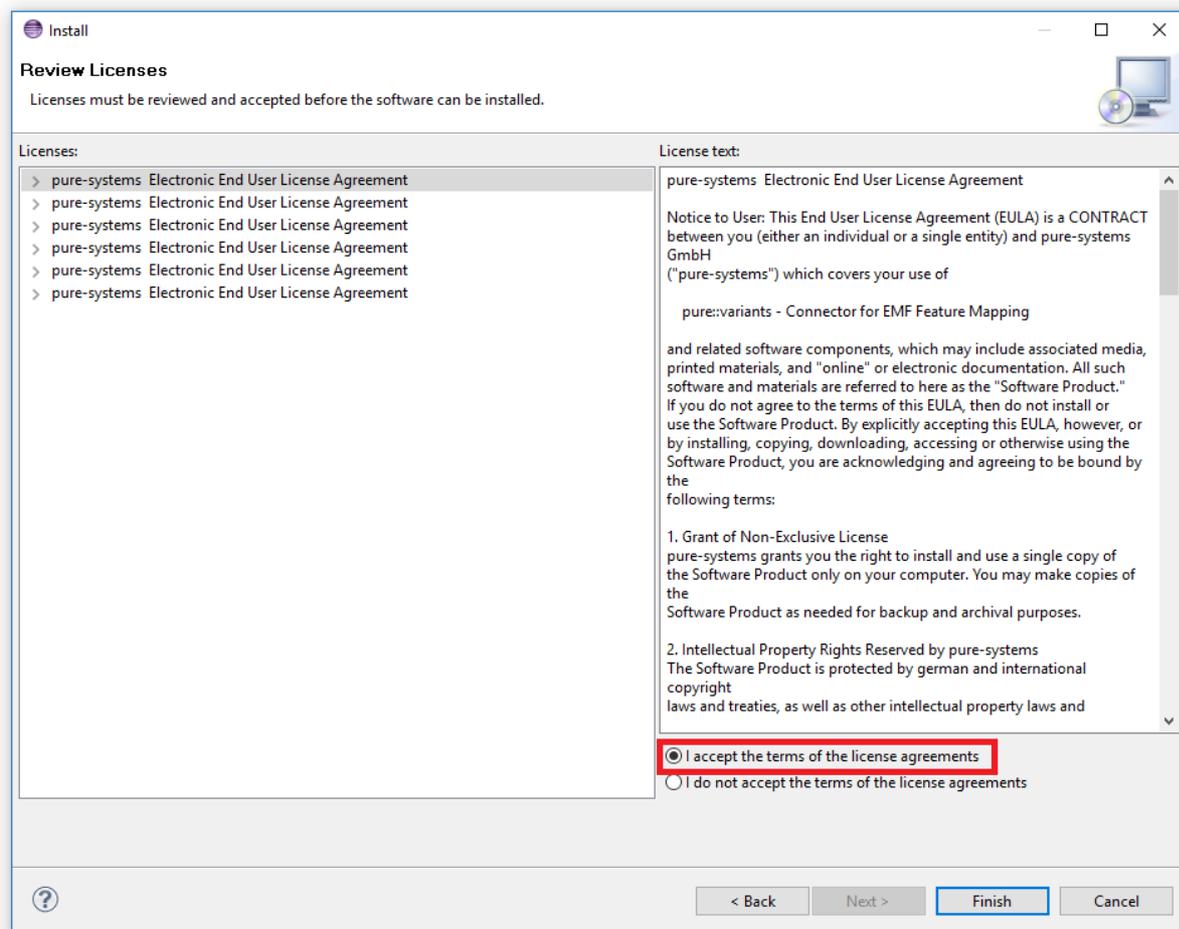
The location of the site depends on the pure::variants product variant. Visit the pure-systems web site (<https://www.pure-systems.com>) or read your registration email to find out which site is relevant for the version of the software you are using.

Figure 20. Update Site Selection

- Unfold the pure::variants update site and select all features to be updated. Select "Next".

Figure 21. pure::variants Plugin Selection

- Accept the license, hit "Next" and then "Finish".

Figure 22. Licence Agreement

- In the dialog select "Install all".
- Restart pure::variants when asked for.

If the online update is not possible (often due to firewall/proxies preventing Eclipse accessing external web sites), please go to the web site using an Internet browser:

- For pure::variants Evaluation use <https://www.pure-systems.com/pv-update>
- For pure::variants Enterprise/Professional use <https://www.pure-systems.com/pvde-update>

and download the "Complete Updatesite" archive:

- Start pure::variants (or the Eclipse into which pure::variants has been installed).
- Select "Help"->"Software Updates"->"Find and Install...".
- Select "Search for new features to install" and "Next".
- Click on button "Archived Update Site" or "Local Update Site".
- Use "Browse" to select the downloaded archive file.
- Press "Ok". The pure::variants update site from the archive should be selected.
- All other check boxes should be unselected to speed up the process. Press "Finish".

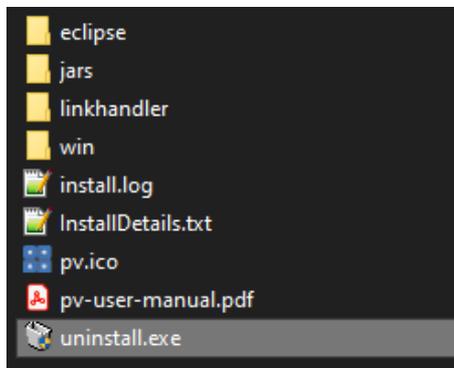
- Unfold everything below pure::variants update site and select the features to be updated. Select "Next".
- Accept the license, hit "Next" and then "Finish".
- In the dialog select "Install all".
- Restart pure::variants when asked for.

3.3. Uninstall pure::variants Client

3.3.1. Uninstall using pure::variants Uninstaller

The uninstaller for the pure::variants client can be started in two different ways. The first is to go to the Windows *Add or remove programs* application and search for *pure::variants Enterprise* and start the uninstaller by using the *Uninstall* action. The uninstaller requires Administrator privileges.

Figure 23. pure::variants Client Uninstaller

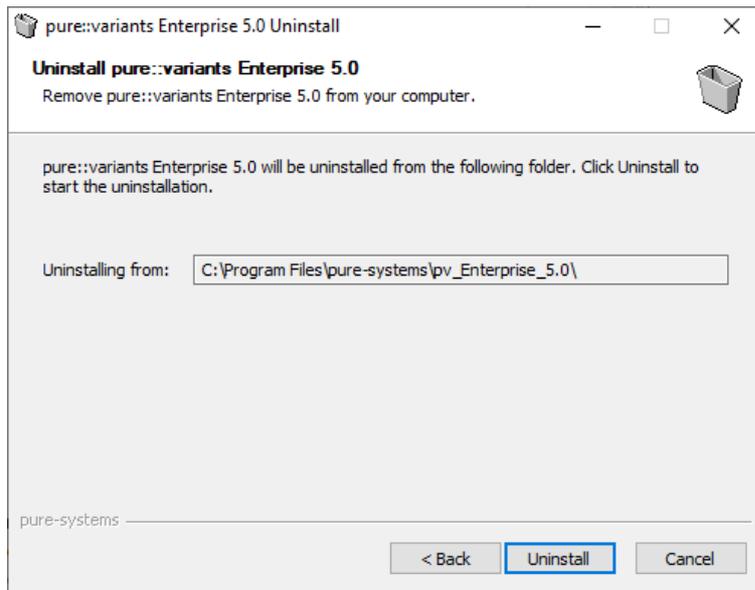


The second way is to navigate to the pure::variants client installation folder and start the uninstaller by double clicking it.

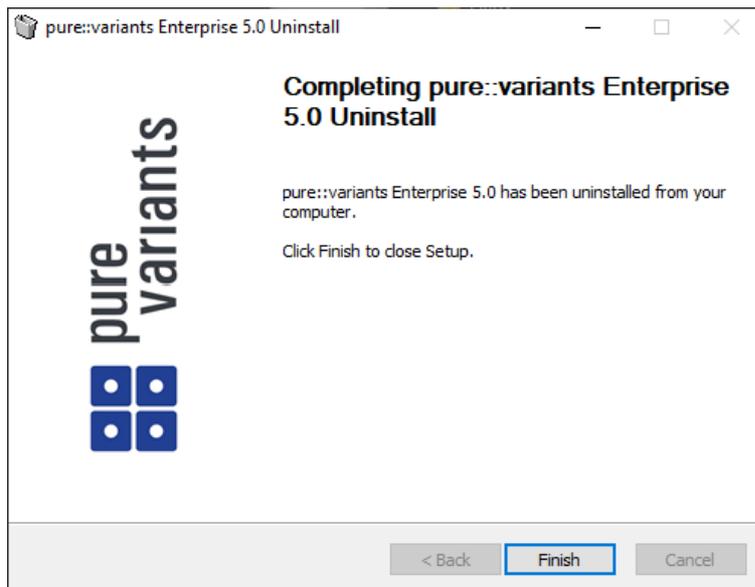
Figure 24. pure::variants Client Uninstaller



Click *Next*.

Figure 25. Uninstall from

Click *Uninstall* to start the uninstall process.

Figure 26. Completing Uninstall

Click *Finish* to close the uninstaller.

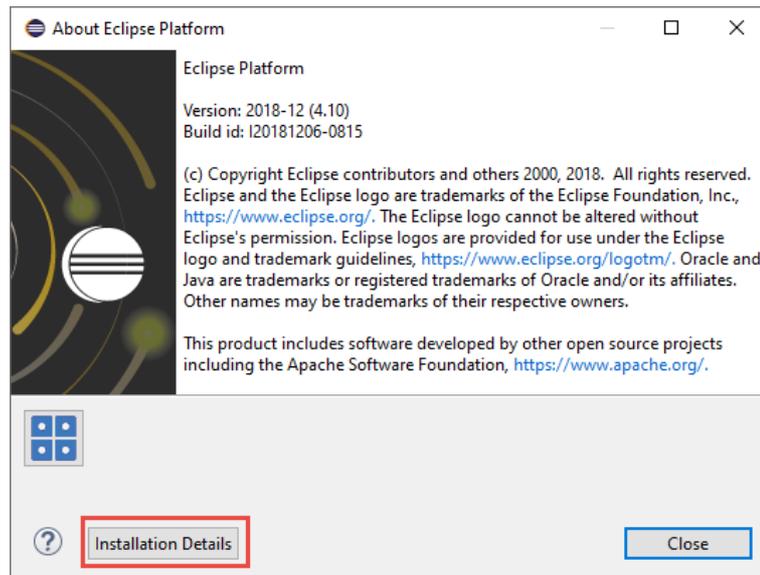
3.3.2. Uninstall pure::variants from existing Eclipse instance

There are two ways to remove pure::variants from an Eclipse instance. You can use the Eclipse command line or remove the pure::variants features one by one in the running Eclipse Instance. Either way a cleanup of the Eclipse instance has to be performed afterwards.

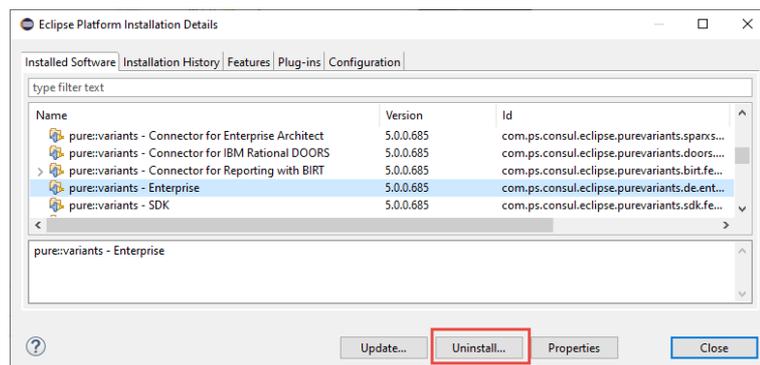
If the Eclipse instance is not needed anymore you can just remove the whole Eclipse installation from the file system. If the Eclipse is of further use, follow one of the installation methods.

Uninstall pure::variants in running Eclipse Instance

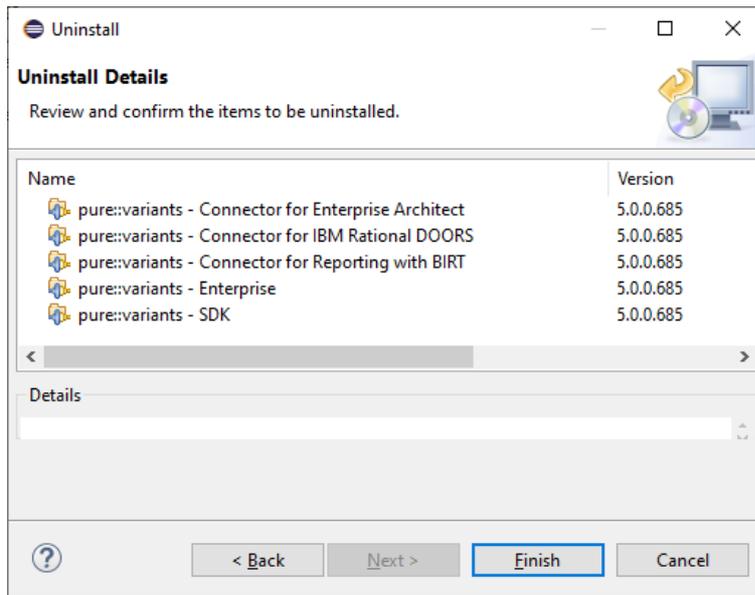
To remove an installed feature from Eclipse using the Eclipse client, open the *About Eclipse Platform* dialog with the *About Eclipse Platform* action in the *Help* menu.

Figure 27. Eclipse About Dialog

Use the *Installation Details* button to access the installation details.

Figure 28. Eclipse About Dialog

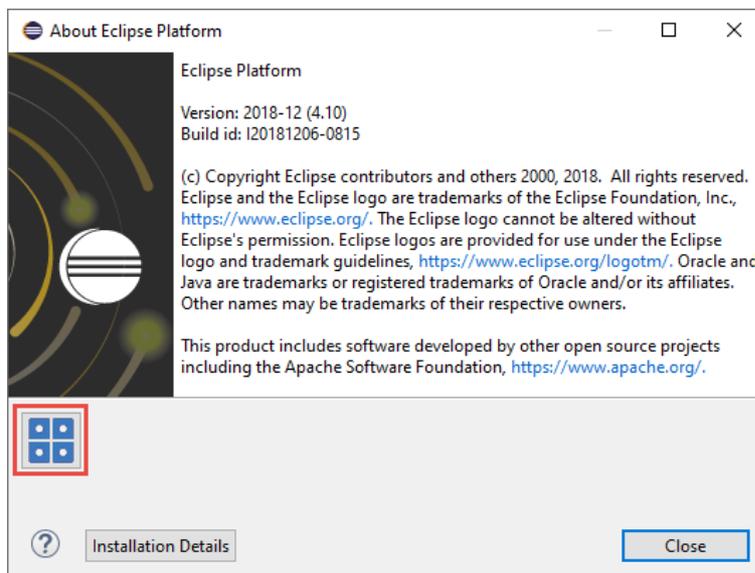
Use the *Uninstall* button to start the uninstallation of the selected features. Selecting multiple features at once is possible.

Figure 29. Eclipse About Dialog

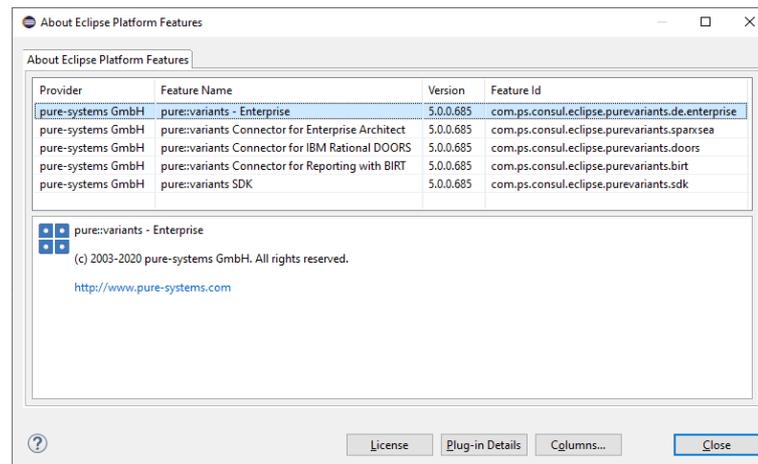
Click *Finish* to start the uninstall process. After it finished, Eclipse will prompt you to restart the application. Click *Restart* to finish the uninstallation.

Uninstall pure::variants using Eclipse uninstall application

To use the uninstall application you need the feature ids of the features you want to uninstall. The feature ids can be found in the *About Eclipse Platform* dialog. Open the dialog with the *About Eclipse Platform* action in the *Help* menu.

Figure 30. Eclipse About Dialog

Click on the pure::variants icon.

Figure 31. Installed pure::variants features

The feature ids are listed in the *Feature Id* column of the upcoming dialog. All feature ids have to be extended by ".feature.group" and are concatenated with ",". The feature id list for the example shown in the previous figure would be:

```
com.ps.consul.eclipse.purevariants.sparsxsea.feature.group,com.ps.consul.eclipse.purevariants.birt.feature.group,com.ps.consul.eclipse.purevariants.de.enterprise.feature.group,com.ps.consul.eclipse.purevariants.doors.feature.group,com.ps.consul.eclipse.purevariants.sdk.feature.group
```

The resulting list of feature ids is used in the following command.

```
"<Eclipse Installation Directory>\eclipse.exe" -nosplash --launcher.suppressErrors -application org.eclipse.equinox.p2.director -uninstallIU "<list of feature ids>" -data "ws" -vmargs -Dequinox.ds.block_timeout=120000 -Dorg.eclipse.ecf.provider.filetransfer.retrieve.readTimeout=120000 -Declipse.p2.mirrors=false -Xms100m -Xmx2048m -Xmnx64m -Xgcpolicy:gencon -XX:MaxPermSize=512M -Xcompressedrefs
```

Cleanup Eclipse after uninstallation

pure::variants stores some settings, license and log files at two locations in the file system. On Windows the first one is C:\Users\\AppData\Roaming\pure-variants-5, and the second C:\ProgramData\pure-variants-5. On Linux based systems the pure-variants-5 folders are located in the users home directory and at /usr/share. These folders should be removed after pure::variants has been completely removed from the computer.

To clean up the Eclipse instance, run the following command.

```
"<Eclipse Installation Directory>\eclipse.exe" -nosplash --launcher.suppressErrors -application org.eclipse.equinox.p2.garbagecollector.application -data "ws" -vmargs -Dequinox.ds.block_timeout=120000 -Dorg.eclipse.ecf.provider.filetransfer.retrieve.readTimeout=120000 -Declipse.p2.mirrors=false -Xms100m -Xmx2048m -Xmnx64m -Xgcpolicy:gencon -XX:MaxPermSize=512M -Xcompressedrefs
```

3.4. Basic Setup of the pure::variants Client

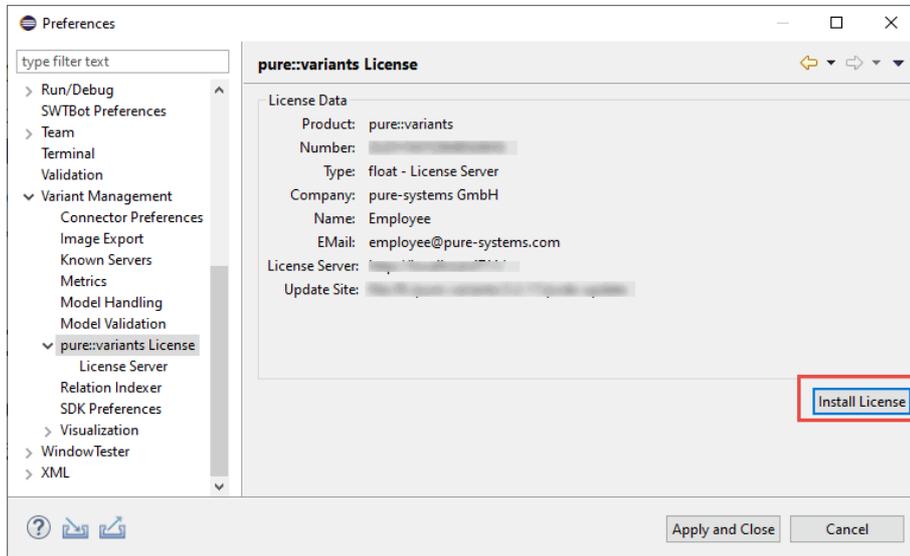
3.4.1. Setup a pure::variants Client License

A valid license file is required in order to use pure::variants. If pure::variants is started and no license is present, then the user is prompted to supply a license. Select the "Yes" button and use the file dialog to specify the license file delivered with pure::variants. The specified license will be stored in the user's application data directory. If you

are using multiple workspaces then the license file has to be installed only once. The pure::variants integrations also use the installed license and thus no further setup step is needed here.

To replace an existing pure::variants license, start pure::variants and open the **Preferences** (menu Window -> Preferences). Navigate to **Variant Management -> pure::variants License** and use the **Install License** button to select the new license.

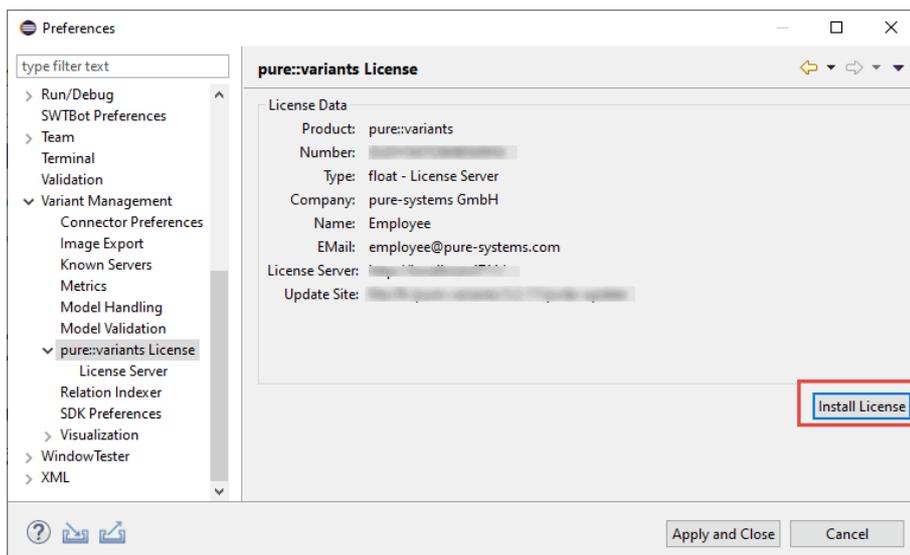
Figure 32. pure::variants License Preferences



3.4.2. Update a pure::variants Client License

If pure::variants is not explicitly asking for a new license, the update can be forced by starting pure::variants and opening menu **Window -> Preferences**. Select **Variant Management -> pure::variants License** and install the license using the provided **Install** button.

Figure 33. pure::variants License Preferences



3.4.3. Add pure::variants Client License using environment variable or Java property

For central or automatic deployed pure::variants clients it may be necessary to also automatically deploy or update the pure::variants client license. For this use case the variable **PVLICENSE** can be used. This variable can either be

introduced as an environment variable or just added as a Java property to the command line starting pure::variants. If this variable is set, the given license is used instead of a possibly previously installed license.

Example for the command line parameter: `-DPVLICENSE=C:/absolute/path/to/the/license/file.lic`

3.5. Trouble Shooting

3.5.1. pure::variants is low on memory

If pure::variants is low on memory it can result in out of memory errors or causing pure::variants to run very slow since Java is trying to free up memory constantly by running the garbage collector.

To solve that problem pure::variants needs to be enabled to use more memory. This can be done by editing the eclipse.ini file, which is located in `<pure::variants installation path>\eclipse\eclipse.ini`.

Add the following three lines to the end of the ini file, if not existing yet. The first line tells Eclipse that there are Java Virtual Machine options following. Xms defines the minimal amount of memory Java is reserving. Xmx defines the maximum amount of memory Java is allowed to use. The default value is 1024 MB. We recommend to set the value to 6144 MB .

```
-vmargs  
-Xms40m  
-Xmx6144m
```

Note

If Eclipse does not start after the eclipse.ini was changed, the maximum amount of memory defined is not valid. There are multiple reasons for this, e.g. Java could not reserve enough memory. Try to decrease the defined maximum memory.

4. pure::variants Connectors

4.1. Installation of pure::variants Connectors

Installing a connector into an existing pure::variants installation works the exact same way like installing the pure::variants client into an existing Eclipse instance. You just have to make sure the depending pure::variants connectors are already installed or they have to be installed together with the new connector. See [the section called "Using update site"](#).

4.2. pure::variants Connector for Capella

To install the pure::variants Connector for Capella, open Capella or Capella Studio and select **Help->Install New Software...** Enter the address of your pure::variants update site. From the list of available features, select "pure::variants - Connector for Capella", "pure::variants - Connector for EMF Feature Mapping" and the pure::variants feature (e.g., "pure::variants - Enterprise").

For installation in Capella, the standard Eclipse update site needs to be set up. Otherwise the installation will fail due to missing dependencies. In Capella 1.1.x the Eclipse update site is configured per default. In Capella 1.2.x, the Eclipse Neon update site still needs to be added. To do that, open **Window->Preferences->Install/Update->Available Software Sites** and add the update site.

- For Capella 1.2.x use <http://download.eclipse.org/releases/neon/>
- For Capella 1.3.x use <https://download.eclipse.org/releases/oxygen/>
- For Capella 1.4.0 use <https://download.eclipse.org/releases/2019-03/>

- For Capella 5.0.0 use <https://download.eclipse.org/releases/2020-06/>

4.3. pure::variants Connector for Team Foundation Server

For using the pure::variants Integration for Microsoft TFS the server has to be prepared and the pure::variants Integration has to be installed.

The work item types, which should be aware of variability information, must be configured with additional attributes. These attributes can be pvRestriction, pvConstraint, pvDefaultSelected and pvName. At least, the attribute pvRestriction should be created (as shown in [Figure 34, “A XML configuration for pvRestriction field.”](#)).

Figure 34. A XML configuration for pvRestriction field.

```
<FIELD name="pvRestriction" refname="PureSystems.Restriction" type="PlainText" />
```

Administrator: For having support while defining restrictions on work items, the control type for the pvRestriction attribute must be configured with "PVRestrictionEditorControl" (see [Figure 35, “A XML configuration for pvRestriction field's control type.”](#)).

Figure 35. A XML configuration for pvRestriction field's control type.

```
<Control FieldName="PureSystems.Restriction" Type="PVRestrictionEditorControl" Label="pvRestriction" LabelPosition="Left" />
```

4.4. pure::variants Connector for PTC Integrity

The pure::variants Connector for PTC Integrity as well as the pure::variants Integrity Integration expect a variant-related preparation of the solution item. For this several changes on the solution and settings are necessary, which are described in the following.

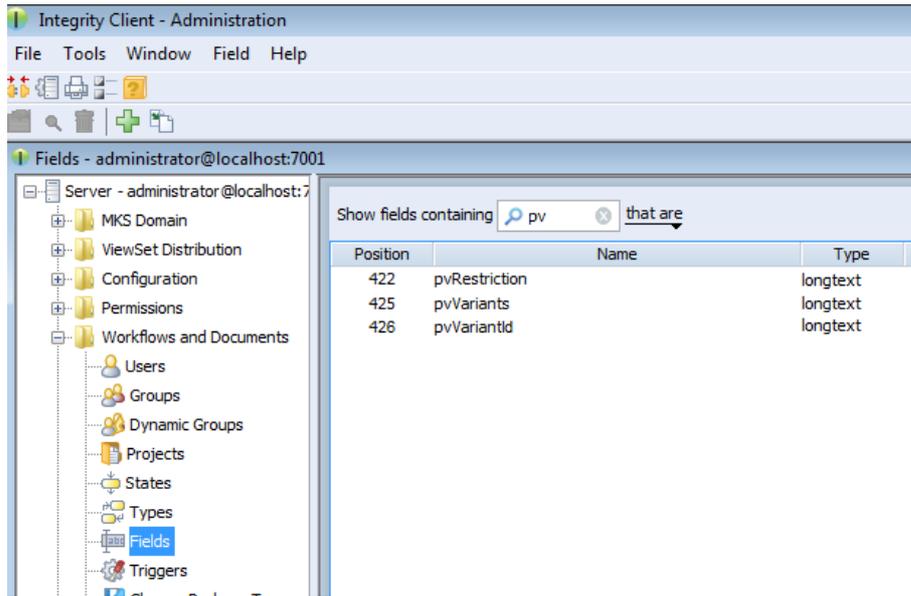
4.4.1. Add additional Fields for pure::variants

The following fields have to be created for the solution item, e.g. **MKS Solution**. In the PTC Integrity Administration open **Workflows and Documents** -> **Fields**. Add the fields by choosing **Create Field...** from the context menu.

Table 2. Additional fields

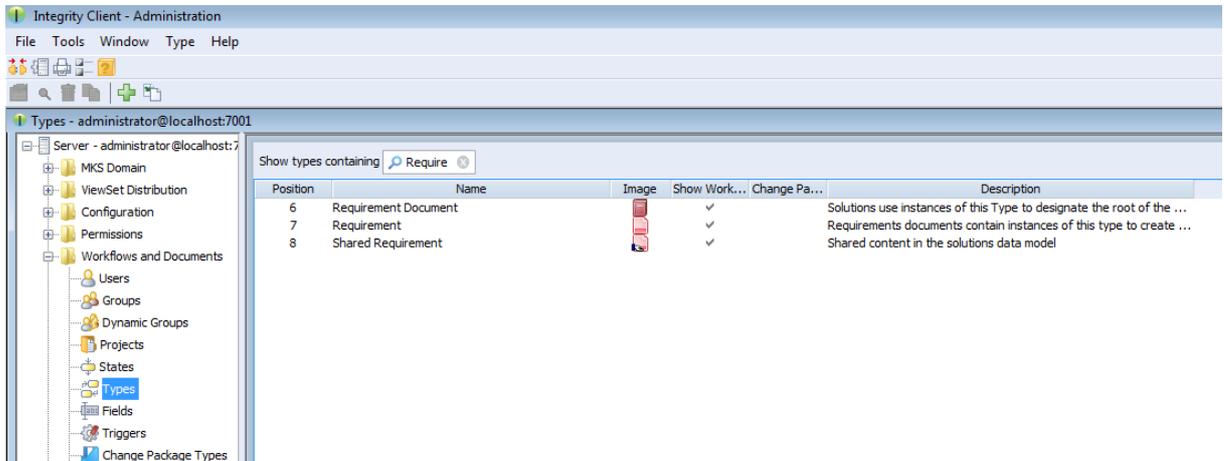
Field	Description
pvRestriction	This field is needed to store the restriction rule on a requirement. Set longtext as the type of the field, and ensure that the field is editable.
pvVariants	This field is needed to store the names of variants a requirement is part of. Set longtext as the type of the field. Ensure that the field is editable.
pvVariantId	This field is needed to store the hexadecimal encoded ID of the pure::variants variant description model which was used to create a requirement document variant in PTC Integrity. Set longtext as the type of the field. Ensure that the field is editable.

Figure 36. Added fields



Open **Workflows and Documents** -> **Types** in the PTC Integrity Administration and filter for all **Requirement** types.

Figure 37. Solution Types

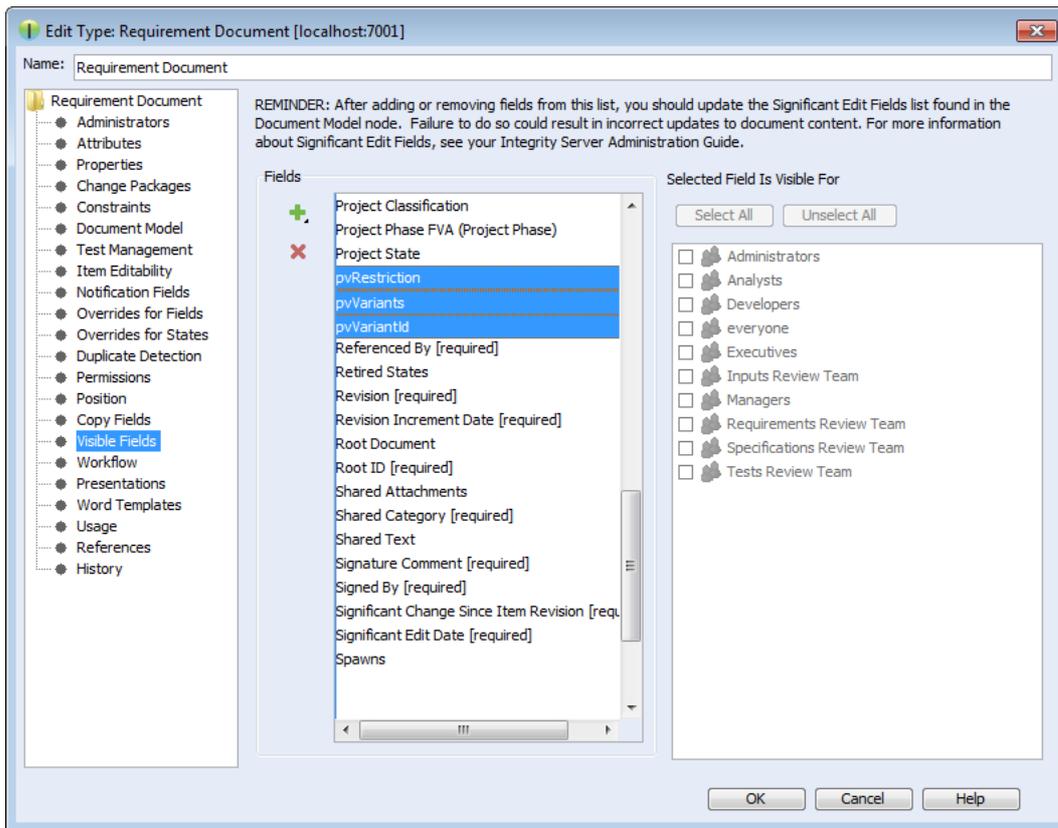


Add the new fields as **Visible Fields** to the following types.

Table 3. Types to add the fields for

Type	Fields to add
Requirements Document	pvRestriction, pvVariants, pvVariantId
Requirement	pvRestriction, pvVariants
Shared Requirement	pvRestriction

Figure 38. Fields added to type Requirement Document



4.4.2. Change Connector and In-Tool Integration Settings

pure::variants uses following settings in order to connect to PTC Integrity. This includes settings for the pure::variants Connector for PTC Integrity, which allows importing and exporting documents, as well as the settings for the In-tool Integration, which allows adding and changing restriction rules in PTC Integrity.

Table 4. Settings

Setting	Default Value	Description
Solution Item	MKS Solution	Used to get configuration properties
Document Type Field	Type	Used for the export of variant documents
Project Field	Project	Used for the export of variant documents
Document Fields	Document Short Title	Used to get several information from imported requirement documents, and to copy fields when exporting variant documents. The first field must always be the document title field
Document Title Field	Document Short Title	Used to get the document title from imported requirement documents, and to calculate the title of exported variant documents
Restriction Rule Field	pvRestriction	Used to get restriction rules from imported requirement documents, and to read and write restriction rules
Variant Enumeration Field	pvVariants	Used to store enumerated variant names in requirement documents

Setting	Default Value	Description
Variant Document ID Field	pvVariantId	Used to store the ID of pure::variants variant description models in variant requirement documents
Requirement Text Field	Text	Used while import to get the text of requirements from requirement documents

Some of these settings can be directly changed before importing and exporting requirement documents. Others can only be changed in the connector configuration file and in the solution type. The following table shows the property names used to change these settings in the configuration file and the solution type.

Table 5. Properties

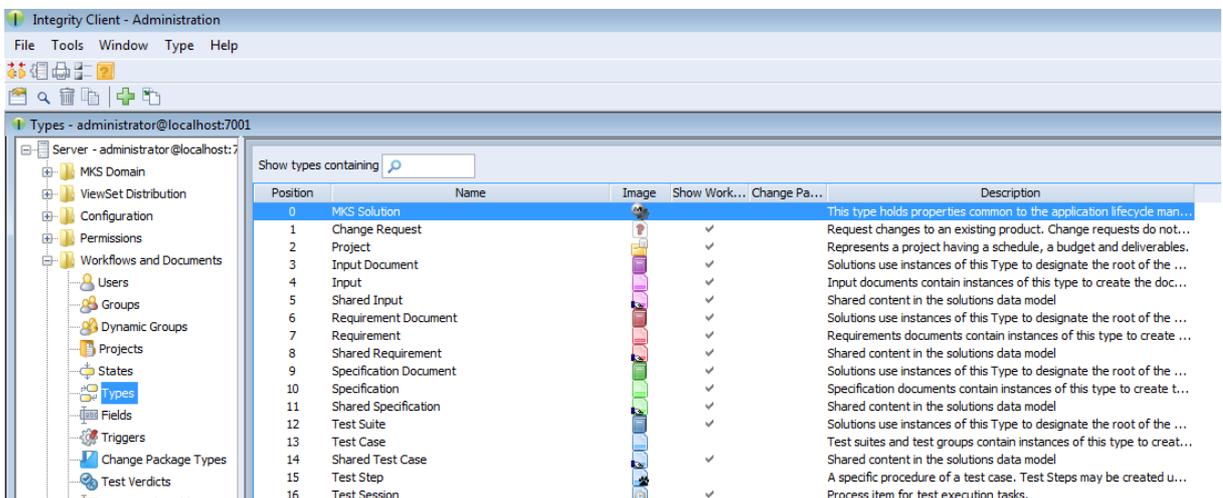
Setting	Config File Property	Solution Type Property
Solution Item	solutionType	
Document Type Field	fieldname.documenttype	PUREVARIANTS.TYPE.FIELD
Project Field	fieldname.documentproject	PUREVARIANTS.PROJECT.FIELD
Document Fields		PUREVARIANTS.VARIANT.FIELDS
Document Title Field	fieldname.documenttitle	PUREVARIANTS.TITLE.FIELD
Restriction Rule Field	attrname.restrictions	PUREVARIANTS.RESTRICTION.FIELD
Variant Enumeration Field	attrname.variants	PUREVARIANTS.VARIANTS.FIELD
Variant Document ID Field	fieldname.variantid	PUREVARIANTS.VARIANTID.FIELD
Requirement Text Field	fieldname.requirementtext	PUREVARIANTS.TEXT.FIELD

To change these settings in the connector configuration file, create the file **pvIntegrity.properties** in directory **%APPDATA%\pure-variants-5**. For each setting to change, add a line with the config file property of the setting assigned to the new value. To change for instance the default field used for storing restriction rules to **MyRestriction** and the default field used for storing the enumerated variants to **MyEnumeratedVariants**, you would add the following two lines to the configuration file (the comments are optional):

```
# Field used to store restriction rules
attrname.restrictions=MyRestriction
# Field used to store enumerated variant names
attrname.variants=MyEnumeratedVariants
```

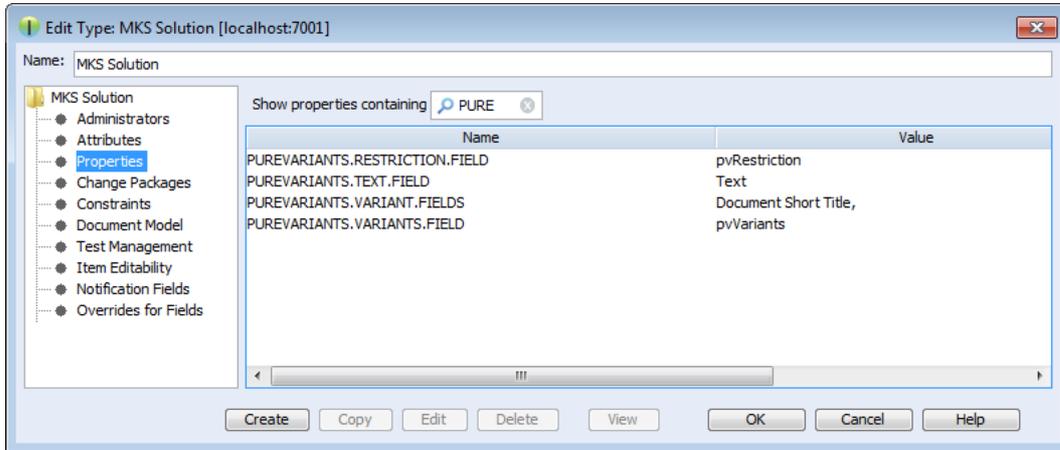
To change these settings on the solution item, open the Administration of PTC Integrity. Switch to **Workflows and Documents -> Types** and select the solution type (e.g. **MKS Solution**).

Figure 39. Type MKS Solution



Right-click the solution type and choose **Edit Type** from the context menu. Then switch to **Properties**, click the **Create** button and enter the solution type property name of the setting you want to change as name. Add the new default value as value and click **OK**.

Figure 40. Added Properties

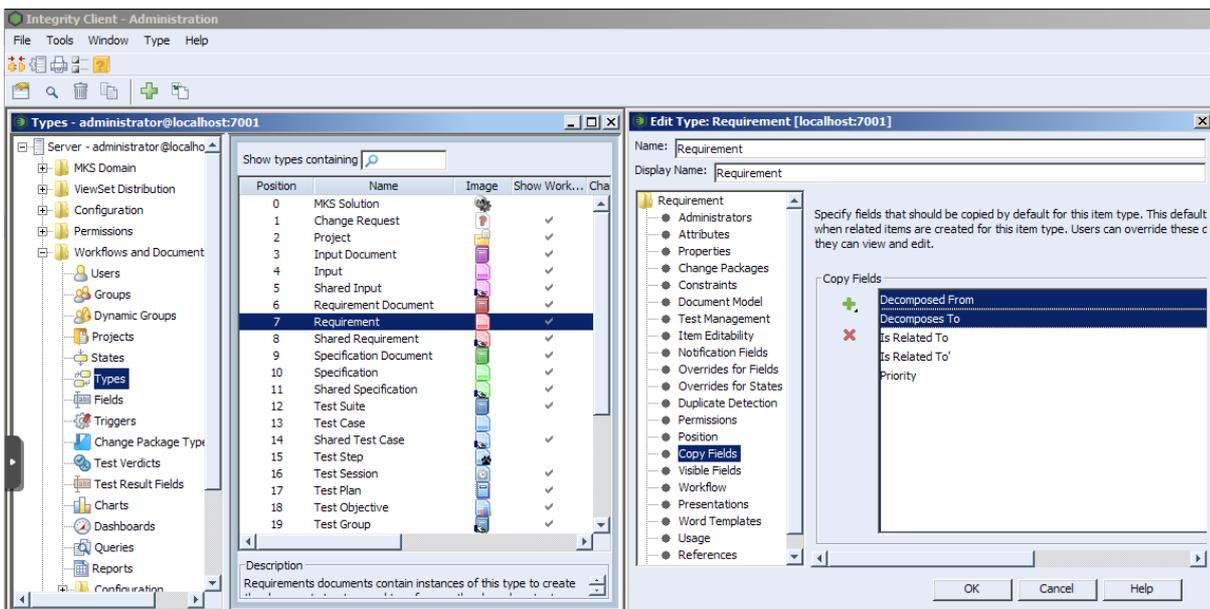


4.4.3. Change Fields Copied for Variant Creation

Several fields of the original document are copied while creating a document variant. This includes the fields mentioned in section Section 4.4.1, “Add additional Fields for pure::variants” but also some fields that are copied by default. The list of fields that are to be copied by default can be configured by the Integrity administrator.

To additionally copy for instance decompose relationships, the administrator has to open the Administration of PTC Integrity. Then switch to **Workflows and Documents -> Types** and edit type **Requirement**. Open the **Copy Fields** list and add the fields **Decomposes To** and **Decomposed From**. This way fields could be added for the types **Requirements Document**, **Requirement**, and **Shared Requirement**.

Figure 41. Decompose Fields added

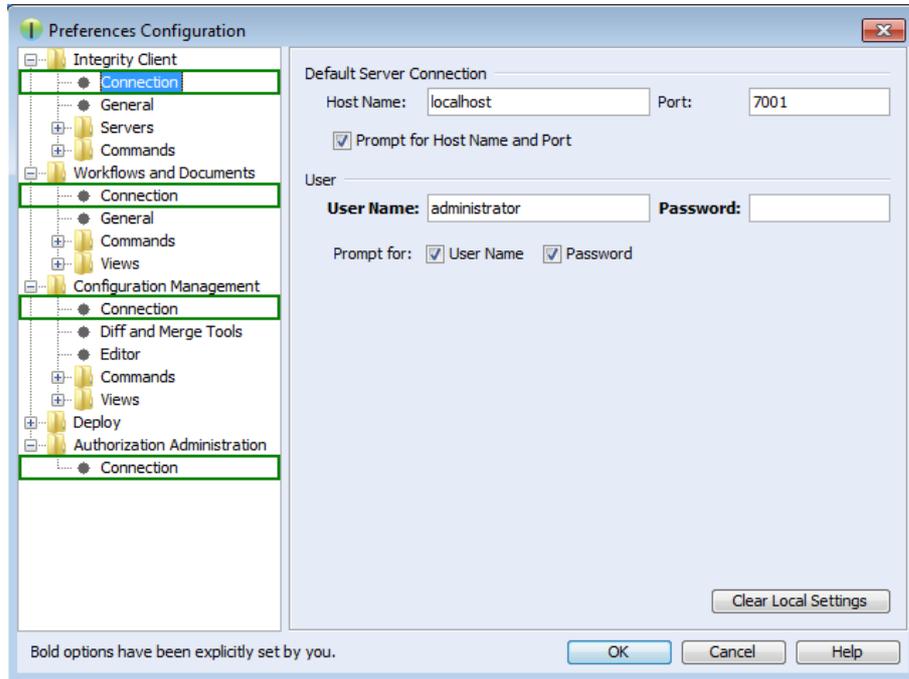


4.4.4. Enable PTC Integrity Client Access

The connector and the integration for PTC Integrity require access to the Integrity client in order to work. Start the PTC Integrity client and open menu **File -> Preferences**. For the entries **Integrity Client**, **Workflow and**

Documents, Configuration Management, and Authorization Administration click the **Connection** section and enable **Prompt for Host Name and Port**, prompt for **User Name**, and prompt for **Password**.

Figure 42. pure::variants Credentials



Note

These connection settings always are used even if you have the option "Prompt for Host Name and Port" enabled and change the connection settings in the corresponding dialog when importing documents from Integrity into pure::variants or exporting variants to Integrity.

4.5. Connector for IBM Rational Rhapsody

If you are working with Rhapsody Model Manager (RMM) projects, additional setup steps are needed.

First, you need to make sure that all required software is installed. That includes RMM (Architecture Management) on the Jazz server and Rational Team Concert (RTC) on your client machine. Also in Rhapsody, the Rhapsody Model Manager add-on needs to be installed.

4.5.1. Preparing IBM Rational Team Concert

In RTC two integrations need to be installed: The *IBM Rational Rhapsody integration for Rational Team Concert* and the *pure::variants Integration for RTC transformation*. The IBM Rational Rhapsody integration for RTC is needed for RTC to work with RMM projects. Please consult the RTC documentation for installation instructions.

The *pure::variants Integration for RTC transformation* is needed during transformation of RMM projects. Without it, the transformation will fail. To install the pure::variants Integration for RTC transformation, please open RTC and use **Help > Install New Software** to install all contents of archived update site `com.ps.consul.eclipse.rtc.integration.feature_[version].zip`. You can find the archived update site zip in your Rhapsody integration installation folder (default is `C:\Program Files\pure-systems\pv_Enterprise_5.0\com.ps.consul.eclipse.ui.rhapsody.integration`).

4.5.2. Preparing pure::variants

For the transformation of RMM projects to work, you still need to set the RTC executable location. You can do this in the pure::variants preferences at **Window > Preferences > Variant Management > Connector Preferences >**

Connector for IBM Rational Rhapsody. Alternatively, you can define an environment or Java system variable that is named `PV_RTC_EXEC_PATH` and whose value points to the RTC executable location.

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

Additionally, you can set a custom location of RTC/EWM's eclipse.ini and lscm.bat files. You can set an environment or Java system variable that is named `PV_RTC_INI_PATH` to set the location of the eclipse.ini file and `PV_RTC_LSCM_BAT` to set the location of the lscm.bat file. If this is not set, the transformation will look for it in the default location.

4.6. Connector for Codebeamer

This chapter describes the installation instructions specific to the Codebeamer connector.

4.6.1. Installation of pure::variants Client

Follow the steps as described in '3.1. Install pure::variants Client', in case installing pure::variants into an eclipse client, see chapter '3.1.2. Install into an existing Eclipse'.

4.6.2. Installation of Server Component and pure::variants Widget to Codebeamer

Following components that are delivered as part of the pure::variants Enterprise installation package need to be installed on the Codebeamer server:

1. The pure::variants server component for Codebeamer to be deployed on the Codebeamer server.

The Jar file are packed in a zip file indicating the compatible pure::variants version for identification: e.g. `com.ps.consul.codebeamer.vel.jar-5.0.11.685.zip`, where '5.0.11.685' stands for the supported pure::variants version.

2. The '*pure::variants Widget to Codebeamer*' that needs to be deployed on the codeBeamer server packed in a zip archive.

The server component is a Spring based custom component running in the application context as defined for codeBeamer (for details see <https://codebeamer.com/cb/wiki/18830>).

Before deploying, unzip the jar files 'com.ps.consul.codebeamer.vel.jar' and 'pvcore.jar' from the zip archive of the server component. Also make sure the server certificate that is used by the Codebeamer server is trusted on the pure::variants client side.

Similarly, the folder 'pv_integration' contained in the zip for the to the pure::variants Widget to codeBeamer' needs to be unzipped.

4.6.3. Installation without running in a docker container

Follow the steps to install:

1. Stop the codeBeamer server
2. Copy 'com.ps.consul.codebeamer.vel.jar' and 'pvcore.jar' found in the zip archive to `<codeBeamer>/tomcat/webapps/cb/WEB-INF/lib`
3. Copy following files included in 'pv_integration' to following locations:
`'pv_integration/widget'` to `<codebeamer>/tomcat/webapps/pv-widget/`, e.g. `/home/appuser/codebeamer/tomcat/webapps/pv-widget/`
4. Restart the Codebeamer server

4.6.4. Installation in a docker image

When running codeBeamer server in a docker container (<https://codebeamer.com/cb/wiki/5562876>), following additional information needs to be defined in the docker compose configuration file:

```
volumes:
  - ./com.ps.consul.codebeamer.vel.jar:<codebeamer>/tomcat/webapps/ROOT/WEB-INF/lib/
  com.ps.consul.codebeamer.vel.jar
  - ./pvcore.jar:<codebeamer>/tomcat/webapps/ROOT/WEB-INF/lib/pvcore.jar
e.g.
- ./libs/com.ps.consul.codebeamer.vel.jar:/home/appuser/codebeamer/tomcat/webapps/ROOT/WEB-
INF/lib/com.ps.consul.codebeamer.vel.jar
- ./libs/pvcore.jar:/home/appuser/codebeamer/tomcat/webapps/ROOT/WEB-INF/lib/pvcore.jar
```

To deploy the 'pure::variant Widget to codeBeamer' following additional information needs to be added to the docker compose configuration file:

```
volumes:
  - ./pv_integration/widget:<codebeamer>/tomcat/webapps/pv-widget/
e.g.
- ./pv_integration/widget:/home/appuser/codebeamer/tomcat/webapps/pv-widget/
```

Then follow the steps to install:

1. Shut down the docker container first
2. Copy 'com.ps.consul.codebeamer.vel.jar' and 'pvcore.jar' found in the zip archive to a location accessible by docker, and as defined in the volumes mapping (see above)
3. Copy the folder 'pv_integration' including all content to a location accessible by docker, and as defined in the volumes mapping (see above). When updating, please make sure to remove old content of the complete directory first.
4. Restart the docker container
5. In Codebeamer, add following the "externalWidgetExtensions" section to the Application Configuration (<https://<codebeamer>/sysadmin/configConfiguration.spr>) as System Administrator:

```
...},
"externalWidgetExtensions" : {
  "uri" : "https://<codebeamer>/pv-widget/extension.json"
}
}
```

4.6.5. Permissions

The communication between the pure::variants client and codeBeamer uses the REST API. In order to use the rest API end point your user need to have 'api_permission'.

To do this, make sure the user group that the users are assigned to in codeBeamer have this permission set.

4.6.6. Getting Version Information of the Server Component

Use following REST call to query the version information of the server component, this way it can be checked if the server component is running correctly:

https://<path_to_codebeamer>/rest/v3/ps/vel/version

Note: Use the credentials (basic authentication) of a codeBeamer user with 'api_permission'.

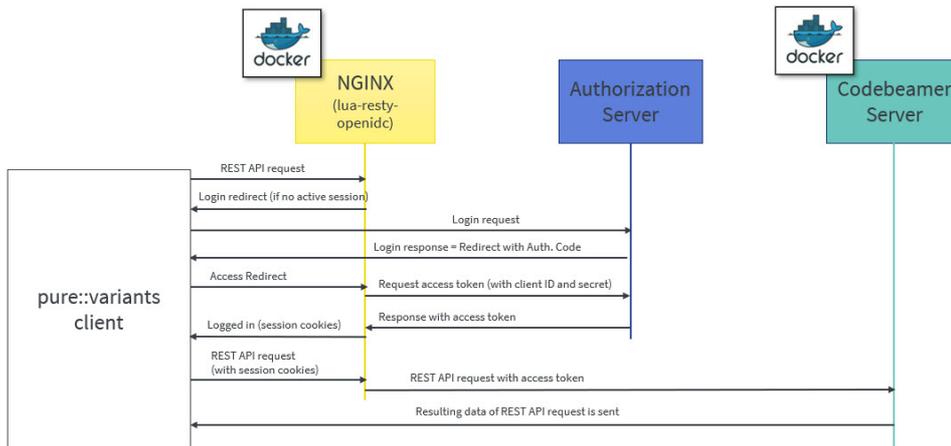
4.6.7. Configuration To Enable Open ID Connect (OIDC) Authentication

OpenID Connect (OIDC) is an authentication protocol that is an extension of OAuth 2.0.

According to this, a dedicated system (Authorization server/ Identity provider) takes care of authenticating a user and issuing access and id token if authentication was successful. This token can be used by clients to obtain data from the Resource Server, in this case the codeBeamer server. The REST API of codeBeamer requires such access token to enable this way of authentication.

To obtain the access token an Authentication Proxy needs to be deployed between the client and the server. All REST API calls are redirected through it, while the authentication process including the refreshing of the tokens is also managed by this proxy in the background.

Figure 43. Authorization process using a proxy



Client registration steps:

1. During the client registration process, both codeBeamer and the Authentication Proxy needs to be registered.
2. The provided configuration files use the 'lua openresty' library for NGINX, implementing OIDC.
3. The registered client's Client ID and Client secret should be added in the configuration files of docker-compose and NGINX (see later).

Following chapter describe the docker configuration files and their parameters.

4.6.8. docker-compose.yml for the NGINX Proxy

This file configures the auth-proxy service that is required by the pure::varaints client. The docker container image named "oidc-auth-proxy" will be created and started, on which NGINX service will be available, which in-turn will be used by pure::variants client to make the REST calls.

Following parameters are to be set:

- **build:** Builds a docker image from a dockerfile. The path is a directory of the host system.
- **ports:** Specifies the port to which NGINX is listening to. 9943:9943 shows the mapping between host port and container port (host port: docker container port).

Note: The port used in oidc-auth-proxy-nginx.conf should be used as docker container port.

- **volumes:** Contains the data which will be used by docker container. It is of the format source:target [:mode] where, source are the host files and target are container path where volumes are mounted.

Any dependent container(s) that will be used by oidc-auth-proxy or any additional container that needs to be built together can be deployed on the same docker machine by adding in the new container configuration under services.

Following code listing shows an example:

```

version: '3.1'
services:
  oidc-auth-proxy:
    build:
      context: .
      dockerfile: oidc-auth-proxy.dockerfile
    ##Specify the port to which nginx is listening to. (host port:docker port)
    ports:
      - 9943:9943
    volumes:
      - ./oidc-auth-proxy-nginx.conf:/usr/local/openresty/nginx/conf/nginx.conf:ro
      - ./server.crt:/usr/local/openresty/nginx/server.cert:ro
      - ./server.key:/usr/local/openresty/nginx/server.key:ro
      - ./cacerts.crt:/usr/local/openresty/nginx/cacerts.crt:ro
    restart: always

```

4.6.9. oidc-auth-proxy.dockerfile for the NGINX Proxy

This file contains the set of commands that has to be executed to build a docker image. lua-resty-openidc library for NGINX is used to authenticate user against Open ID Connect provider. Hence this file contains the command to load the base image of openresty from docker hub and then Install the required packages on the current docker image, followed by command to start the NGINX.

Following code listing shows an example:

```

FROM openresty/openresty:alpine-fat
RUN apk add --update openssl-dev git && luarocks install lua-resty-openidc
CMD ["/usr/local/openresty/bin/openresty", "-g", "daemon off;"]

```

4.6.10. oidc-auth-proxy-nginx.conf for the NGINX Proxy

The directives that need to be adapted are as follows:

- Set **listen** port to which NGINX should listen to.
- The **server name** can be domain name or ip address of the host machine on which docker is running.
- The **redirect_uri_path** should match the uri pre-registered in Authorization server during client registration.
- OpenID Connect defines a **discovery** mechanism where OpenID Server publishes its metadata at a well known url of the format: `https://server.com/.well-known/openid-configuration`
- The **client_id** and **client_secret** are obtained from the Authorization Server after the client registration.
- **proxy_pass** value can be a docker container on which codeBeamer application is running, e.g. `http://container-name:8090`; or it can be a codeBeamer application server url to which a request should be forwarded, e.g. `http` or `https://server-name:port(optional)`;

Following code listing shows an example:

```

events {
  worker_connections 128;
}

http {
  resolver 127.0.0.11 ipv6=off;
  lua_package_path '~:/lua/?.lua;;';
  lua_ssl_trusted_certificate /usr/local/openresty/nginx/cacerts.crt;
  lua_ssl_verify_depth 5;
  lua_shared_dict discovery 5m;
  lua_shared_dict jwks 5m;

  server {
    listen 9943 ssl; ##mention the port to which nginx should listen to

```

```

server_name codebeamer.example.com; ##domain name or ip address of the host on which
docker is running
ssl_certificate /usr/local/openresty/nginx/server.cert;
ssl_certificate_key /usr/local/openresty/nginx/server.key;
ssl_protocols TLSv1.2 TLSv1.3;

location / {
    access_by_lua_block {
        local opts = {
            ##redirect_uri should match the uri pre-registered in Authorization server during client
            registration.
            redirect_uri_path = "/login/oauth/authenticate.spr",
            ##OpenID Connect defines a discovery mechanism where OpenID Server publishes its metadata at
            a well known url of the format: https://server.com/.well-known/openid-configuration
            discovery = "https://jas.example.com:9643/oidc/endpoint/jazzop/.well-known/openid-
            configuration",
            ##Client_id and client_secret obtained from the authorization server after the client
            registration.
            client_id = "<set here the client ID>",
            client_secret = "<set here the client secret>",
            scope = "openid profile email",
            access_token_expires_leeway = 30,
            accept_none_alg = false,
            accept_unsupported_alg = false,
            renew_access_token_on_expiry = true,
            access_token_expires_in=3600,
            session_contents = {access_token=true, id_token=true}
        }
        local res, err = require("resty.openidc").authenticate(opts)
        if err then
            ngx.status = 500
            ngx.say(err)
            ngx.exit(ngx.HTTP_INTERNAL_SERVER_ERROR)
        end
        ngx.req.set_header("Authorization", "Bearer " .. res.access_token)
        ngx.req.set_header("X-User", res.id_token.email)
    }
    ##proxy_pass value can be a docker container on which codebeamer application is running.
    For ex., http://container-name:8090;
    ##or it can be a codebeamer application server url to which a request should be forwarded.
    For ex., http or https://server-name:port(optional);
    proxy_pass http://codebeamer-app:8090;
}
}
}

```

4.6.11. Steps to Setup a Docker-container the NGINX Proxy

Following are the steps to create a docker container image named “oidc-auth-proxy”. NGINX will available on this docker container on the specified port.

1. Place the files provided (docker-compose.yml, oidc-auth-proxy.dockerfile, oidc-auth-proxy-nginx.conf) in a folder.
2. Place certificates to be used within the same folder. This will be used by NGINX for SSL handshake.
3. Modify the NGINX configuration file oidc-auth-proxy-nginx.conf as explained in the previous section (oidc-auth-proxy-nginx.conf).
4. Run docker-compose up to create/start a container.

5. pure::variants Tool Integrations

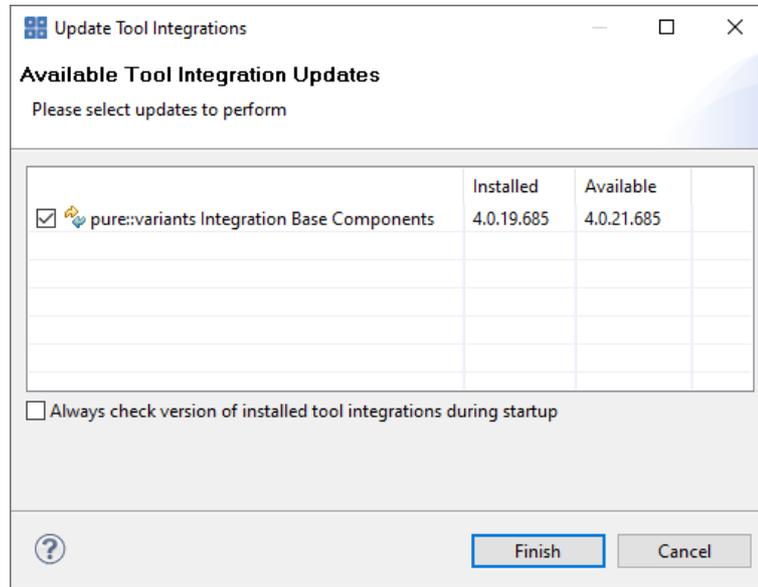
5.1. Install pure::variants Tool Integrations

For using the pure::variants Integrations for several tools the integrations have to be installed. If pure::variants is installed using the Installer, the tool integrations are installed along with the corresponding pure::variants con-

nector. If the installer was not used or the integration was not installed along with the corresponding connector then follow the next steps. Usually a dialog comes up after pure::variants starts informing about not installed or not up-to-date integrations.

If this dialog does not come up automatically it can be opened using the following menu entry in the pure::variants **Help** menu. Go to **Help->pure::variants** the item **Tool Integration Updates**.

Figure 44. Install Tool Integrations



A dialog comes up listing all available Tool Integrations. Just select the integrations you want to install or update and finish the dialog. pure::variants will guide you through the installation process. If an automatic update is possible, pure::variants will just perform the update, without showing an installer.

The option **Always check version of installed tool integrations during startup** enables a check if all available tool integrations are already installed and up to date. This check is performed each time pure::variants starts.

5.1.1. pure::variants Desktop Hub

If the pure::variants client was installed with the installer the pure::variants Desktop Hub is already installed along with the pure::variants client. If the installer was not used or the pure::variants Desktop Hub is missing for another reason the installation can be triggered from within the pure::variants client.

Please see [Section 5.1, “Install pure::variants Tool Integrations”](#) for installing the **pure::variants Integration Base Components**.

5.1.2. pure::variants Integration for Doors

Please see [Section 5.1, “Install pure::variants Tool Integrations”](#) for installing the integration executable.

After finishing the installation successfully, the Doors client has to be started with a command line option, which enables the Integration menu within Doors.

The command line should look similar to this: `doors -a "<Menu installation path>\pure-variants"`. Without this command line option the Integration cannot be triggered and so not be used.

On Windows platforms it is also possible to add the directory to the registry key `HKLM\Software\Telelogic\Doors\Doors version\Addins`.

1. Open the Registry editor

2. Browse to Doors installation in HKEY_LOCAL_MACHINE\SOFTWARE\Telelogic\DOORS\<DOORS version number>\Config
3. Right click config Key to add a new string value
 - Value Name set to **Addins**
 - Value Data set to the path of the pure::variants menu directory

With administrative access to the Doors installation the Add-In can also be installed for all users of this installation using the shared DXL library. See the Doors Help topic "Configuring Doors" for more information.

Note

This requires adaptations of the pure::variants Integration menu DXL scripts.

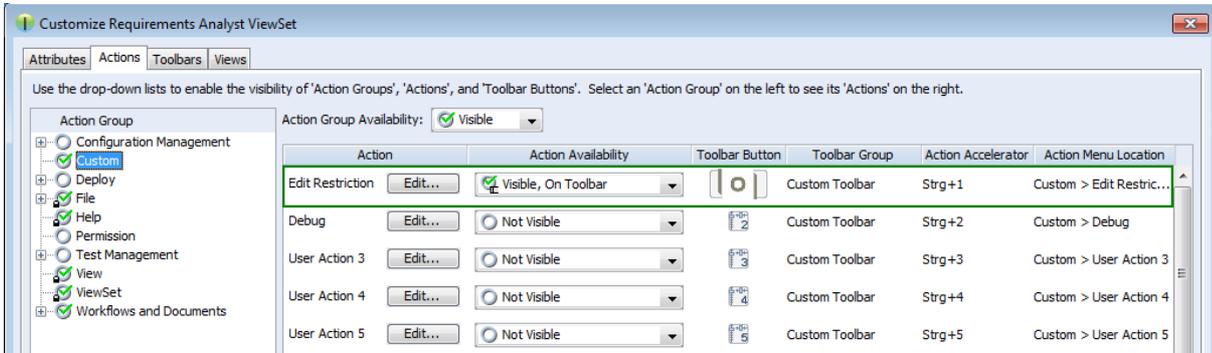
Now the Integration should be available in Doors. To verify if the installation was successful, open a Doors module and select from the menu **pure::variants** the item **Open pure::variants Integration**. If the pure::variants Integration window opens, the Integration was installed correctly.

5.1.3. pure::variants Integration for PTC Integrity

Please see [Section 5.1, "Install pure::variants Tool Integrations"](#) for installing the integration executable.

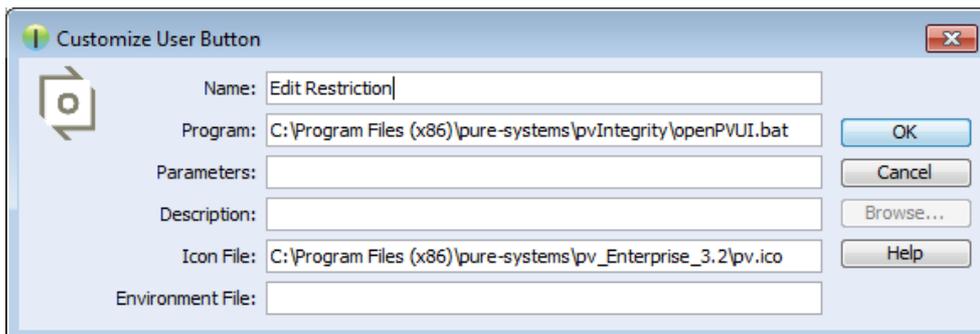
For starting the integration within PTC Integrity some additional steps have to be performed. Start the PTC Integrity client application. Open menu **ViewSet** -> **Customize** and switch to page **Actions**. Click on action group **Custom** to view the custom user actions.

Figure 45. Custom Actions



Click the button **Edit** of a user action to customize it. Name the action **Edit Restriction**. As the program to execute enter or browse to the path of file **openPVUI.bat** which is located in the installation path of the pure::variants Integration for PTC Integrity. There is also an icon file **pv.ico** in this directory you can use.

Figure 46. Custom Button "Edit Restriction"



5.1.4. pure::variants Integration for IBM Rational Rhapsody

Please see [Section 5.1, “Install pure::variants Tool Integrations”](#) for installing the integration executable.

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

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

Now the Integration should be available for the given project. You can open the Integration window at **Tools > pure::variants**.

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

5.1.5. pure::variants Integration for Enterprise Architect

Please see [Section 5.1, “Install pure::variants Tool Integrations”](#) for installing the integration executable.

Now the Integration should be available in Enterprise Architect. Select **Extensions->Add-In Windows** to open the Add-In window, which shows the pure::variants Integration user interface. Furthermore, you can enable or disable the Integration at **Extensions -> Manage Add-Ins...** (Since Enterprise Architect 14, you can find both entries in tab **Specialize**).

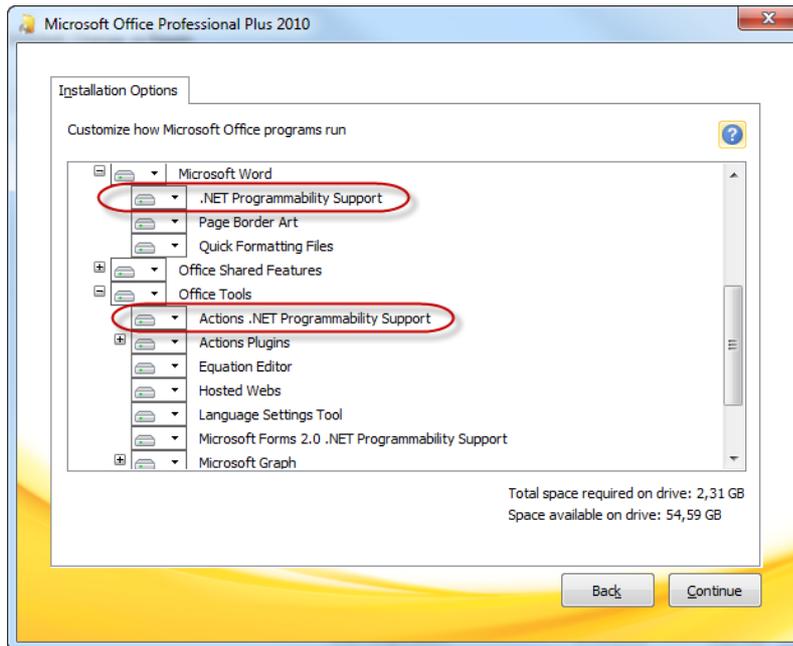
5.1.6. pure::variants Integration for Microsoft Office

The Integration will not work if the following features are not installed with Microsoft Office:

- Microsoft Word / .NET Programmability Support (if using Microsoft Office Word Integration)
- Microsoft Excel / .NET Programmability Support (if using Microsoft Office Excel Integration)
- Office Tools / Actions .NET Programmability Support

They can be added to the Microsoft Office installation as follows:

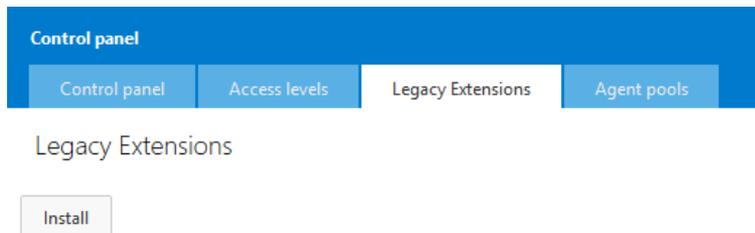
1. Open the Windows Control Panel and navigate to **Programs and Features**
2. Right-click on your Microsoft Office Installation and select **Change**
3. Select **Add or Remove Features**
4. Add the features marked in [Figure 47, “Adding Missing Features to the Office Installation”](#) to your Office Installation.
5. Press **Continue** and close the Dialog.

Figure 47. Adding Missing Features to the Office Installation

Now the installation of the pure::variants Integration for Microsoft Office should run successfully.

5.1.7. pure::variants Integration for Team Foundation Server

Administrator: At first, please download the pure::variants windows installer package from your pure::variants update site and extract the pure::variants TFS integration zip archive with name *com.ps.consul.web.ui.tfs2015-x.x.x.zip*. To install the integration navigate to the **Control panel**->**Legacy Extensions** and press **Install**. Thereby browse to your local copy of pure::variants TFS Integration zip archive.

Figure 48. Install pure::variants TFS Integration

Please ensure that the pure::variants TFS Integration is enabled, after installation.

5.1.8. Advanced Integration Setup

If you are using a pure::variants model or license server, establishing a connection with that server may need extra configuration. This may be the case, for example, if the server is located behind a proxy server or the communication with the server is encrypted and a self-signed certificate is used.

The steps needed to do in these cases differ, depending on which type of integration you are using. For java-based integrations, such as the Integration for IBM Rational Rhapsody and the Integration for PTC Integrity, it is necessary to specify proxy and certificate settings manually in *pv.properties* files. For .NET-based integrations (all other integrations), the Windows proxy and certificate settings are used automatically, so no additional setup is necessary.

Advanced Setup of Java-based Integrations

The following integrations are Java-based:

- pure::variants Integration for IBM Rational Rhapsody
- pure::variants Integration for PTC Integrity

All extra configurations in java-based integrations can be done by manually editing file `pv.properties`. You can find it in two different locations:

- If you want to configure the settings for all users on the machine, please edit

```
%PROGRAMDATA%/pure-variants-5/pv.properties
```

- If you want to configure the settings only for the current user, please edit

```
%APPDATA%/pure-variants-5/pv.properties
```

Note that:

- In case the file does not exist, you need to create it and any necessary folders first.
- Before editing `pv.properties`, make sure that no pure::variants Integration is running (e.g. Integration for Word, Excel, Doors, or the p::v Desktop Hub), otherwise your changes may be overwritten when closing the integration.
- Path delimiters in any paths you enter must be forward slashes or escaped backward slashes (/ or \). Otherwise the path cannot be read.
- All property names are case-sensitive.

Proxy Settings

The following properties can be set to configure your proxy settings.

Table 6. Proxy Settings

Property Name	Comments based on Java system property documentation
<code>http.proxyHost</code>	The hostname, or address, of the proxy server
<code>http.proxyPort</code>	The port number of the proxy server
<code>https.proxyHost</code>	The hostname, or address, of the proxy server in case HTTPS is used
<code>https.proxyPort</code>	The port number of the proxy server in case HTTPS is used
<code>http.nonProxyHosts</code>	Indicates the hosts that should be accessed without going through the proxy. Typically this defines internal hosts. The value of this property is a list of hosts, separated by the ' ' character. In addition the wildcard character '*' can be used for pattern matching. For example <code>http.nonProxyHosts=*.foo.com localhost</code> will indicate that every hosts in the <code>foo.com</code> domain and the <code>localhost</code> should be accessed directly even if a proxy server is specified. The default value excludes all common variations of the loopback address.
<code>java.net.useSystemProxies</code>	Set this to "true" to use Windows' global proxy settings (default: false), which are set in the Internet Explorer or in the Windows system settings. If one of the above properties is set, it overrides the respective Windows system property.

For example to use Windows' proxy settings, you would need to append this line to `pv.properties`:

```
java.net.useSystemProxies=true
```

Or to set all properties manually, you would need to append something like this:

```
http.proxyHost=YourHTTPProxyHost
http.proxyPort=80
https.proxyHost=YourHTTPSProxyHost
https.proxyPort=443
http.nonProxyHosts=*.foo.com|localhost
```

HTTPS Connection with License Server

The following HTTPS-related properties can be set. For more details, please refer to the respective Java system property documentation.

Table 7. HTTPS Settings

Property Name	Comments
javax.net.ssl.trustStore	Path to your trust store
javax.net.ssl.trustStorePassword	Password of your trust store
javax.net.ssl.trustStoreType	Trust store type (e.g. JKS)
javax.net.ssl.keyStore	Path to your key store
javax.net.ssl.keyStorePassword	Password of your key store
javax.net.ssl.keyStoreType	Key store type (e.g. JKS)
javax.net.debug	Activation of debug mode (e.g. "all" to write all possible debug logs)
com.sun.net.ssl.checkRevocation	Enable certificate revocation checking

For example when using a self-signed certificate that is stored in trust store `D:/sandbox/servercert/cert-trusted.jks` you could append the following lines:

```
javax.net.ssl.trustStore=D:/sandbox/servercert/cert-trusted.jks
javax.net.ssl.trustStorePassword=password
```

However, it is also possible to permanently accept a self-signed certificate when trying to first connect to your model or license server. pure::variants or the integrations will open a certificate acceptance dialog on the first connection attempt.

Advanced Setup of .NET-based Integrations

The following integrations are .NET-based:

- pure::variants Desktop Hub
- pure::variants Integration for Doors
- pure::variants Integration for Microsoft Excel and Microsoft Word
- pure::variants Integration for Enterprise Architect
- pure::variants Integration for Zuken CR-8000

Since pure::variants 4.0.19, the way the connection to pure::variants model or license servers is done has changed. Therefore, no advanced setup as for java integrations is necessary anymore. The proxy and certificate settings configured in Windows are used for the connection. So if the connection works in a browser that uses the Windows certificate and proxy settings (e.g. Chrome or Edge), the connection should work in all .NET-based integrations, too. The only exception from that rule are the supported security protocols:

Per default, the following security protocols are supported in .NET-based integrations: SSL3, TLS 1.0 - 1.3. To instead let Windows decide which protocols to support, you need to add registry entry "SystemDefaultTlsVersions"

as documented here: <https://docs.microsoft.com/en-us/mem/configmgr/core/plan-design/security/enable-tls-1-2-client#configure-for-strong-cryptography>.

If for some reason you want to switch back to the old server connection behavior as used in all java integrations, you can do that as described in the following section.

Switching Back to Previous Server Connection Behaviour

There are two ways to switch back to the previous server connection behaviour. Either you add the Windows environment variable `PV_FORCE_JAVA_SOAP_SERVER_CONNECTION` with value `true`, or you add line `forceJavaSoapConnection=true` to file `pv.properties` (see the section called “Advanced Setup of Java-based Integrations” for instructions how to edit `pv.properties`).

After switching back to the previous server connection behaviour, you may need to configure connection settings (e.g., proxy settings) in the same way as for java integrations.

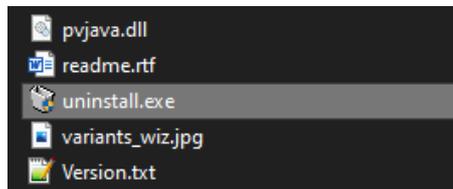
5.2. Update pure::variants Tool Integrations

To update an pure::variants tool integration the same mechanism as for installing a pure::variants tool integration is used. Please consult section [Section 5.1, “Install pure::variants Tool Integrations”](#) for a detailed description.

5.3. Uninstall pure::variants Tool Integrations

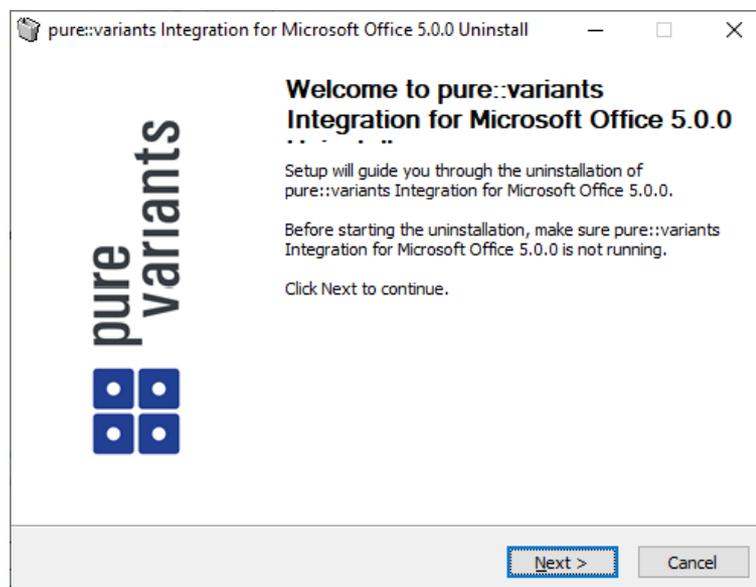
The uninstaller for the pure::variants integration can be started in two different ways. The first one is to go to the Windows *Add or remove programs* application and search for the pure::variants integration and start the uninstaller by using the *Uninstall* action. The uninstaller requires Administrator privileges.

Figure 49. pure::variants Integration Uninstaller



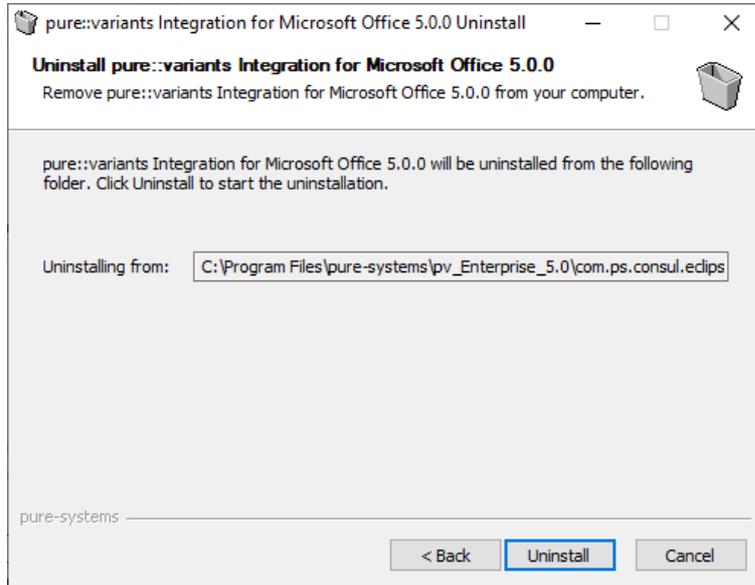
The second possibility is to navigate to the pure::variants client installation folder and start the uninstaller by double clicking it.

Figure 50. pure::variants Integration Uninstaller



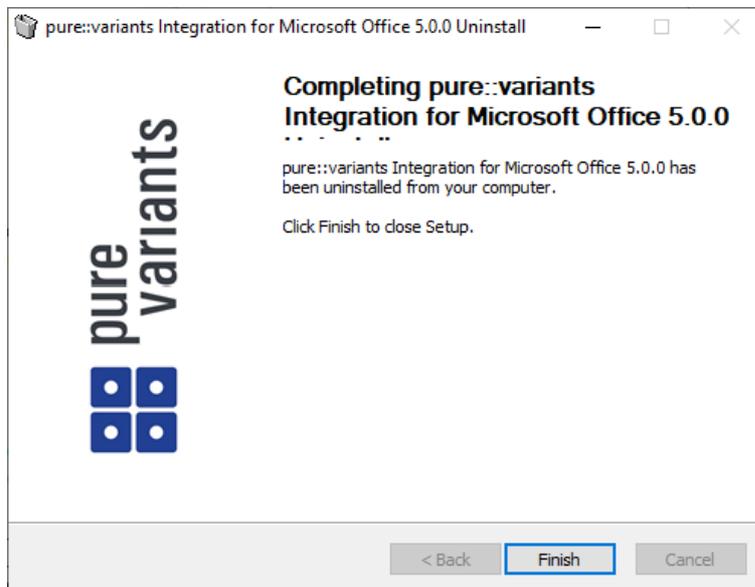
Click *Next*.

Figure 51. Uninstall from



Click *Uninstall* to start the uninstall process.

Figure 52. Completing Uninstall



The installation is successfully finished. Click *Finish* to close the uninstaller.

5.4. Basic Setup of pure::variants Tool Integrations

When you first use the Desktop Hub after installation, it is necessary to check whether the license preferences are correct. To this end, open the preferences dialog via the  button in the Desktop Hub window or by selecting **Hub Configuration** from the pure::variants tray menu.

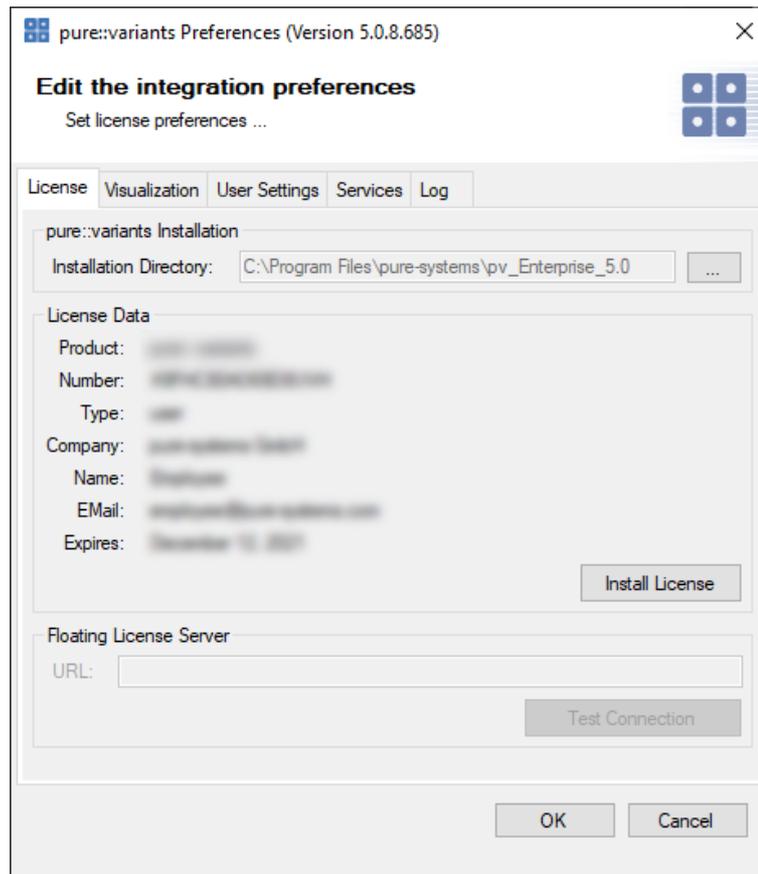
A dialog opens that shows the path to your pure::variants installation and your license information (see [Figure 53, “Preferences Dialog”](#)). If any of the information is missing, you need to enter it. Use the ... button in the

pure::variants Installation group to enter the installation directory, and the **Install License** button to specify your license.

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

Now you can use the Desktop Hub.

Figure 53. Preferences Dialog



5.4.1. Server Connection Setup

If you are using a pure::variants floating license, establishing a connection with the pure::variants license server may need extra configuration. This may be the case, for example, if the license server is located behind a proxy server or the communication with the server is encrypted and a self-signed certificate is used.

Since pure::variants 4.0.19, the way the connection to pure::variants model or license servers is done has changed. Therefore, no advanced setup should be necessary anymore, since the settings configured in Windows are used (e.g., proxy settings, certificates). However, it is still possible to switch back to the previous server connection behavior.

Switching Back to Previous Server Connection Behaviour

This sections describe the switch back to the Java based Soap connection method. It is not recommended to use this but may help to fix connection problems. This mechanism is available for .net based integrations only.

.net based integrations are

- [Section 5.1.2, “pure::variants Integration for Doors”](#)

- pure::variants Integration for Microsoft Office
- [Section 5.1.5, “pure::variants Integration for Enterprise Architect”](#)
- [Section 5.1.7, “pure::variants Integration for Team Foundation Server”](#)

There are two ways to switch back to the previous server connection behaviour. Either you add the Windows environment variable `PV_FORCE_JAVA_SOAP_SERVER_CONNECTION` with value `true`, or you add line `forceJavaSoapConnection=true` to file `pv.properties` (see below for instructions how to edit `pv.properties`).

Once you have switched back, you may again need extra configuration to connect to a license or model server. See [the section called “Advanced Integration Setup”](#) for extra configuration setps.

Advanced Integration Setup

This steps are necessary only, if you have switched the Soap server connection method to the `ols` behavior. See [the section called “Switching Back to Previous Server Connection Behaviour”](#).

If you are using a pure::variants floating license, establishing a connection with the pure::variants license server may need extra configuration. This may be the case, for example, if the license server is located behind a proxy server or the communication with the server is encrypted and a self-signed certificate is used.

All extra configurations can be done by manually editing file `pv.properties`. You can find it in two different locations:

- If you want to configure the settings for all users on the machine, please edit

```
%PROGRAMDATA%/pure-variants-5/pv.properties
```

- If you want to configure the settings only for the current user, please edit

```
%APPDATA%/pure-variants-5/pv.properties
```

When editing `pv.properties`, please note that:

- In case the file does not exist, you need to create it and any necessary folders first.
- Before editing `pv.properties`, make sure that no pure::variants Integration is running (e.g. Integration for Word, Excel, Doors, or the p::v Desktop Hub), otherwise your changes may be overwritten when closing the integration.
- Path delimiters in any paths you enter must be forward slashes or escaped backward slashes (`/` or `\\`). Otherwise the path cannot be read.
- All property names are case-sensitive.

Proxy Settings

The following properties can be set to configure your proxy settings.

Table 8. Proxy Settings

Property Name	Comments based on Java system property documentation
<code>http.proxyHost</code>	The hostname, or address, of the proxy server
<code>http.proxyPort</code>	The port number of the proxy server
<code>https.proxyHost</code>	The hostname, or address, of the proxy server in case HTTPS is used
<code>https.proxyPort</code>	The port number of the proxy server in case HTTPS is used

Property Name	Comments based on Java system property documentation
http.nonProxyHosts	Indicates the hosts that should be accessed without going through the proxy. Typically this defines internal hosts. The value of this property is a list of hosts, separated by the ' ' character. In addition the wildcard character '*' can be used for pattern matching. For example http.nonProxyHosts=*.foo.com localhost will indicate that every hosts in the foo.com domain and the localhost should be accessed directly even if a proxy server is specified. The default value excludes all common variations of the loopback address.
java.net.useSystemProxies	Set this to "true" to use Windows' global proxy settings (default: false), which are set in the Internet Explorer or in the Windows system settings. If one of the above properties is set, it overrides the respective Windows system property.

For example to use Windows' proxy settings, you would need to append this line to `pv.properties`:

```
java.net.useSystemProxies=true
```

Or to set all properties manually, you would need to append something like this:

```
http.proxyHost=YourHTTPProxyHost
http.proxyPort=80
https.proxyHost=YourHTTPSProxyHost
https.proxyPort=443
http.nonProxyHosts=*.foo.com|localhost
```

HTTPS Connection with License Server

The following HTTPS-related properties can be set. For more details, please refer to the respective Java system property documentation.

Table 9. HTTPS Settings

Property Name	Comments
javax.net.ssl.trustStore	Path to your trust store
javax.net.ssl.trustStorePassword	Password of your trust store
javax.net.ssl.trustStoreType	Trust store type (e.g. JKS)
javax.net.ssl.keyStore	Path to your key store
javax.net.ssl.keyStorePassword	Password of your key store
javax.net.ssl.keyStoreType	Key store type (e.g. JKS)
javax.net.debug	Activation of debug mode (e.g. "all" to write all possible debug logs)
com.sun.net.ssl.checkRevocation	Enable certificate revocation checking

For example when using a self-signed certificate that is stored in trust store `D:/sandbox/servercert/cert-trusted.jks` you would need to append the following lines:

```
javax.net.ssl.trustStore=D:/sandbox/servercert/cert-trusted.jks
javax.net.ssl.trustStorePassword=password
```

HTTPS Connection with Model Access Service (pure::variants Desktop Hub only)

Per default, the self-signed certificate that is used for securing the model access service connection is only generated for the current user. Thus, when there are multiple users working on the same machine, each user would use a different self-signed certificate and each user would get a security exception the first time he uses the model access service (e.g., when working with DoorsNG).

To prevent that, you as an administrator, can manually configure which certificate is used for all users and register it in the Windows *Trusted Root Certification Authorities* store to make sure no security exception is shown. To achieve that, you can set the following properties in

```
%PROGRAMDATA%/pure-variants-5/pv.properties
```

Table 10. Model Access Service Settings

Property Name	Comments
enableHttpService	true if the model access service should be enabled
httpServicePort	Port that the model access service should run on
enableHTTPS	true if the connection should be secured
keystore	Path to your key store
keystorePassword	Password of the given key store

For example, if you wanted to enable the model access service for all users, use a secure connection, and use your own keystore that is located at `C:/ProgramData/pure-variants-5/selfsigned.jks`, you would need to append the following lines to `pv.properties`:

```
# enable the model access service
enableHttpService=true
# secure the connection
enableHTTPS=true
# the port on which the service listens
httpServicePort=9443
# the path to your keystore, which must be a java keystore and which contains your certificate
keystore=C:/ProgramData/pure-variants-5/selfsigned.jks
# the password to the keystore
keystorePassword=password
```

6. pure::variants Web Integration

6.1. IBM Rational DOORS NG Web Integration

The pure::variants Integration for DOORS NG is distributed in a WAR archive (`com.ps.consul.web.ui.doorsng-x.x.x.war`) and can be found in the **pure::variants Windows Installer package** on the pure::variants update site.

Note

For brevity we have renamed the war-archive to **pvwidget.war**. At least for Apache Tomcat, the war-archive name implies the context-path, so the pure::variants Integration's Catalog will be reachable at **`https://[pv-server-FQDN]:[port]/pvwidget/catalog.xml`**.

Remember FQDN is the fully qualified domain name. The port number might be optional, if configured with standard SSL port (443)

6.1.1. Requirements for pure::variants Integration Deployment

The deployment of pure::variants Integration for DOORS NG has the following requirements:

- Extension must be hosted on a web server application that can be configured to run with Oracle JDK/JRE or OpenJDK.
- Extension must be accessible from a web server via HTTPS.
- The web server must not require any form of authentication to read the extension files.

- The certificate that is installed in the web server must be a valid certificate and must match the server's domain.
- Java Runtime Environment (JRE) or Java Development Kit (JDK) version 1.6 or later.

6.1.2. Installation on Apache Tomcat

Software Requirements

Apache Tomcat 8.5.x is recommended for deployment.

Installation of the pure::variants Integration for DOORS NG

The pure::variants Integration for DOORS NG must be deployed on your application server. This has to be done once by the system administrator. It entails copying of *pvwidget.war* into the *webapps* directory, which is located in the Apache Tomcat installation directory.

Note

In a Tomcat deployment, name of the war file becomes the context path of the web application. In order to configure a nested context path for a web application, the name of the war should contain the individual context path parts separated by '#'. For example, to make the pure::variants Integration's Catalog accessible by a url like <https://localhost:8443/pv/widget/dng/catalog.xml>, the name of the war file needs to be named as *pv#widget#dng.war*. And the war file should be placed into the *webapps* directory. For further details please see the Official Tomcat [documentation](#).

Configuration of Trust Store

As pure::variants Integration does client requests to DOORS NG application server, the Integration's client must be configured with the trusted SSL-certificates. One way to do this is to create a keystore, add the DOORS NG SSL-certificate to it and configure Tomcat's Java-Runtime with this keystore as a trust store.

In case of **Windows**, please create and open *[TOMCAT_INSTALL_DIR]/bin/setenv.bat* file to add the following lines:

```
set "JAVA_OPTS=%JAVA_OPTS% -Djavax.net.ssl.trustAnchors=[TRUST_STORE_PATH] "
set "JAVA_OPTS=%JAVA_OPTS% -Djavax.net.ssl.trustStore=[TRUST_STORE_PATH] "
set "JAVA_OPTS=%JAVA_OPTS% -Djavax.net.ssl.trustStorePassword=[TRUST_STORE_PASSWORD] "
```

An example configuration could look like this:

```
set "JAVA_OPTS=%JAVA_OPTS% -Djavax.net.ssl.trustAnchors=c:\keystore.jks "
set "JAVA_OPTS=%JAVA_OPTS% -Djavax.net.ssl.trustStore=c:\keystore.jks "
set "JAVA_OPTS=%JAVA_OPTS% -Djavax.net.ssl.trustStorePassword=password "
```

In case of **Linux**, please create and open *[TOMCAT_INSTALL_DIR]/bin/setenv.sh* file to add the following lines:

```
export JAVA_OPTS=$JAVA_OPTS " -Djavax.net.ssl.trustAnchors=[TRUST_STORE_PATH] "
export JAVA_OPTS=$JAVA_OPTS " -Djavax.net.ssl.trustStore=[TRUST_STORE_PATH] "
export JAVA_OPTS=$JAVA_OPTS " -Djavax.net.ssl.trustStorePassword=[TRUST_STORE_PASSWORD] "
```

An example configuration could look like this:

```
export JAVA_OPTS=$JAVA_OPTS " -Djavax.net.ssl.trustAnchors=c:\keystore.jks "
export JAVA_OPTS=$JAVA_OPTS " -Djavax.net.ssl.trustStore=c:\keystore.jks "
export JAVA_OPTS=$JAVA_OPTS " -Djavax.net.ssl.trustStorePassword=password "
```

Update or reinstallation of pure::variants Integration for DOORS NG

Stop the Apache tomcat. Go to the Apache Tomcat installation directory, go in to **work** directory then go in to **Catalina** directory and delete the **localhost** directory. Go back to Apache Tomcat installation directory, go in to

webapps directory and **delete** the previously installed **pvwidget.war** and **pvwidget** directory. Now **copy** the new **pvwidget.war** in the **webapps** directory and start the Apache Tomcat again. No configuration change in Apache Tomcat is required in case of an update.

6.1.3. Installation on Websphere Liberty

Software Requirements

WebSphere Liberty Kernel v19.0.0.6+ is recommended for deployment.

Server Setup

Please follow the following steps for Websphere Liberty setup:

1. Install the WebSphere Liberty according to the [official documentation](#).
2. Create a server by running the following command:

```
bin\server create [SERVER_NAME]
```

3. Please add the following lines into the **server.config.dir/server.xml** file:

```
<featureManager>
  <feature>jsp-2.3</feature>
  <feature>ssl-1.0</feature>
</featureManager>
```

Please run the following command:

```
bin\installUtility install [SERVER_NAME]
```

SSL Configuration

Please ensure that the server is configured with SSL. Please refer to the [Securing communications with Liberty](#) section of the official documentation.

Installation of the pure::variants Integration for DOORS NG

Copy the war archive file **pvwidget.war** to **server.config.dir/apps**. Add following line into the **server.config.dir/server.xml** file:

```
<webApplication contextRoot="pvwidget" location="${server.config.dir}/apps/pvwidget.war"/>
```

Note

Nested context path can be configured by specifying the whole path in contextRoot attribute of the webApplication. For example, to make the pure::variants Integration's Catalog accessible by a url like <https://localhost:8443/pv/widget/dng/catalog.xml>, the contextRoot attribute should be configured as **contextRoot="pv/widget/dng"**.

The final **server.config.dir/server.xml** file should look similar to this:

```
<?xml version="1.0" encoding="UTF-8"?>
<server description="new server">
  <!-- Enable features -->
  <featureManager>
    <feature>jsp-2.3</feature>
    <feature>ssl-1.0</feature>
  </featureManager>

  <httpEndpoint host="*" httpPort="-1" httpsPort="9443" id="defaultHttpEndpoint"/>
```

```
<ssl id="defaultSSLConfig" keyStoreRef="defaultKeyStore" sslProtocol="SSL"/>
<keyStore id="defaultKeyStore" location="keystore.jks" password="{xor}LykrOiwR"
type="JCEKS"/>

<webApplication contextRoot="pvwidget" location="{server.config.dir}/apps/pvwidget.war"/>
</server>
```

Configuration of Trust Store

As pure::variants Integration does client requests to DOORS NG application server, the Integration's client must be configured with the trusted SSL-certificates. One way to do this is to create a keystore, add the DOORS NG SSL-certificate to it and configure Liberty's Java-Runtime with this keystore as a trust store. Please create and open **server.config.dir/jvm.options** (see [Directory locations and properties](#)) to add the following lines:

```
-Djavax.net.ssl.trustAnchors=[TRUST_STORE_PATH]
-Djavax.net.ssl.trustStore=[TRUST_STORE_PATH]
-Djavax.net.ssl.trustStorePassword=[TRUST_STORE_PASSWORD]
```

An example configuration could look like this:

```
-Djavax.net.ssl.trustAnchors=c:\keystore.jks
-Djavax.net.ssl.trustStore=c:\keystore.jks
-Djavax.net.ssl.trustStorePassword=password
```

Update or reinstallation of the pure::variants Integration for DOORS NG

Stop the WebSphere Liberty Server. Replace the **pvwidget.war** in the **server.config.dir/apps** directory. **Start** the web server again.

6.1.4. Uninstall the pure::variants Integration for DOORS NG

Uninstall on Apache Tomcat

In order to uninstall pure::variants Integration for DOORS NG, stop the Apache Tomcat. Please go to Apache Tomcat installation directory and then go to **webapps** directory. Delete the **pvwidget.war** file and the **pvwidget** directory from the webapps folder.

Go back to Apache Tomcat installation directory, go in to **work** directory and then go in to **Catalina** directory and delete the **localhost** directory from it. Start the Apache Tomcat.

Uninstall on WebSphere Liberty

In order to uninstall the pure::variants Integration for DOORS NG, please stop the WebSphere Liberty server. Go to **server.config.dir/apps** folder and remove the **pvwidget.war**. Edit the **server.config.dir/server.xml** and remove the following line from the file.

```
<webApplication contextRoot="pvwidget" location="{server.config.dir}/apps/pvwidget.war"/>
```

Save the file and start the WebSphere Liberty server again.

6.1.5. Administrative Setup of the pure::variants Integration for DOORS NG

In order to visualize the variability, following settings need to be configured on the Jazz side.

JTS Advanced Settings

Browse to JTS home page. Click **Manager Server** and then click **Advanced Properties** in the left menu.

Scroll down to **OpenSocial gadget enable SSO** and set its value to **true** as shown in figure [Figure 54, "OpenSocial gadget enable SSO Setting"](#).

Figure 54. OpenSocial gadget enable SSO Setting

com.ibm.team.repository.service.opensocial.gadgetprovider.OpenSocialGadgetProviderRestService Preview		
Property	Current Value	
OpenSocial gadget supplier whitelist	<input type="text"/>	
	Default Value	
Property	Current Value	Default Value
OpenSocial gadget enable SSO	<input type="text" value="true"/>	false

For the versions ELM 7.0.2 iFix004, ELM 7.0.1 iFix009, CLM 6.0.6.1 iFix018, CLM 6.0.6 iFix022 and onwards, an extra configuration is required. In **External resources allowlist**, please enter the URL of the widget. For example if pure::variants Integration for DOORS NG is accessible at <https://jazz.server.net:8443/pvwidget/catalog.xml> then enter <https://jazz.server.net:8443/pvwidget/> in the allowlist as shown in the figure [Figure 55, “External resources allowlist”](#).

Figure 55. External resources allowlist

com.ibm.team.repository.service.opensocial.gadgetprovider.OpenSocialGadgetProviderRestService Preview		
Property	Current Value	
External resources allowlist	<input type="text" value="https://<server>:<port>/pvwidget/"/>	
	Default Value	
Property	Current Value	Default Value
OpenSocial gadget enable SSO	<input type="text" value="true"/>	false

Further, scroll down to **Jazz Authentication Services**, in **Jazz Authentication Proxy SSO Cookies** field and replace its value with `LtpaToken, LtpaToken2, JSESSIONIDSSO, JSA_SESSION_IDENTITY`. In the next field in **Jazz Authentication Proxy SSO Whitelist**, write the URL of the server hosting the widget. Assuming that pure::variants Integration DOORS NG war was renamed as `pvwidget.war`, the URL must end with `/pvwidget/vel` as shown in the figure [Figure 56, “Jazz Authentication Proxy SSO Settings”](#).

Figure 56. Jazz Authentication Proxy SSO Settings

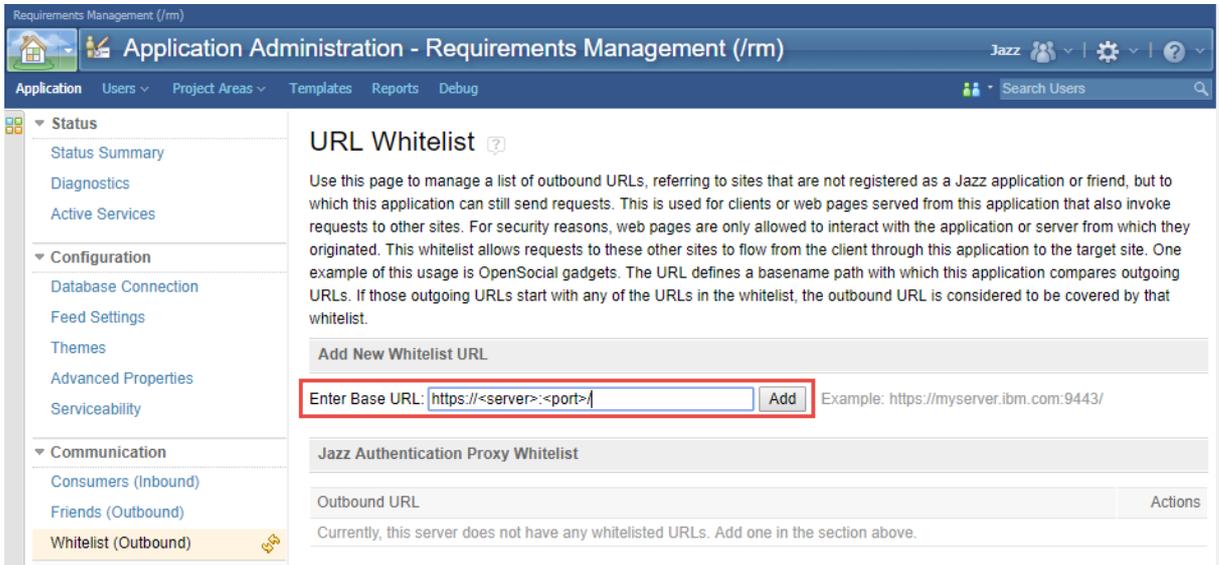
Jazz Authentication Services		
com.ibm.team.repository.internal.service.auth.AuthTokenScrubTask Edit		
Property	Current Value	Default Value
Jazz Authentication Token scrubber task run delay	86400	86400
com.ibm.team.repository.internal.service.auth.impl.AuthConfigurationProperties Preview		
Property	Current Value	
Jazz Authentication Proxy SSO Cookies	<input type="text" value="LtpaToken, LtpaToken2, JSESSIONIDSSO, JSA_SESSION_IDENTITY"/>	
	Default Value	
	LtpaToken, LtpaToken2, JSESSIONIDSSO	
Jazz Authentication Proxy SSO Whitelist	<input type="text" value="https://<server>:<port>/pvwidget/vel"/>	

Jazz RM Whitelist

As shown in [Figure 57, “Adding A URL In RM Whitelist”](#), go to Jazz RM application administration page. Click the **Whitelist (Outbound)** on the left menu. In the **Enter Base URL** field of the **Add New Whitelist URL** section,

enter the **Base URL** of the server where the pure::variants Integration for DOORS NG is installed. Click **Add** button and the newly added URL should be added to the **Outbound URL** list on the same page.

Figure 57. Adding A URL In RM Whitelist



Note

On clicking the **Add** button, Jazz tries to access the URL being added. And if the URL is accessible only then the URL is added to the list of **Outbound URL**.

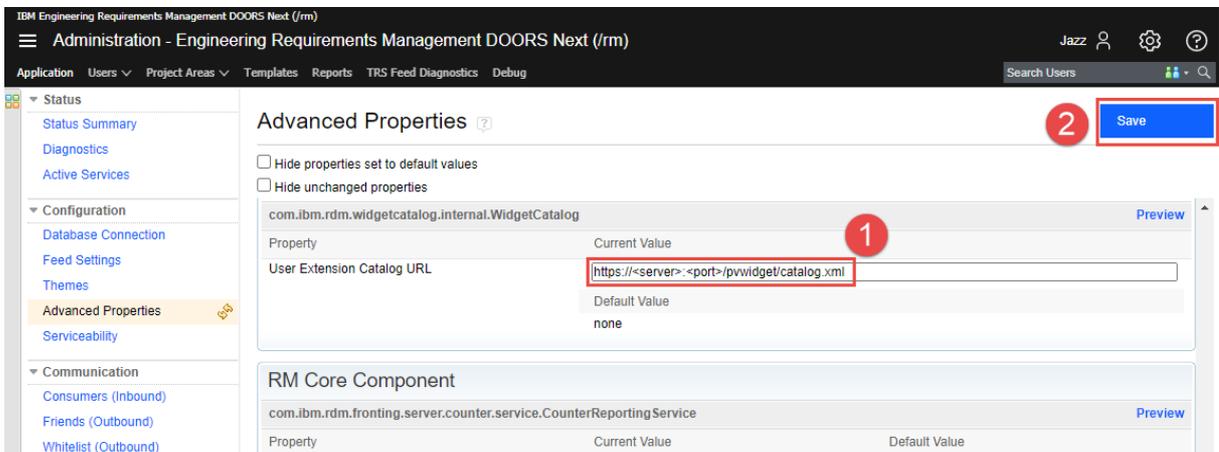
User Extension Catalog URL

Browse to Jazz RM Advanced Settings page. RM Advanced Settings page can be accessed by pointing the web browser to following URL:

```
https://<server>:<port>/rm/admin#action=com.ibm.team.repository.admin.configureAdvanced
```

Scroll to **User Extension Catalog URL** as shown in figure [Figure 58, “RM User Catalog Configuration”](#).

Figure 58. RM User Catalog Configuration



Considering that the context path of the pure::variants Integration is **pwwidget**, enter the Catalog URL as `https://<server>:<port>/pwwidget/catalog.xml` and **Save** the settings.

Adding DOORS NG Integration in an already existing Catalog

Jazz RM allows configuration of only one Catalog URL. If you already have a Catalog URL configured in RM Advanced Properties. Then the pure::variants Integration for DOORS NG can be added to the existing catalog document. Lets consider that pure::variants Integration for DOORS NG is deployed on a web server and accessible via a URL like this:

```
https://server.local.com:8443/pvwidget/catalog.xml
```

Open the catalog xml document that is already configured in the RM Advanced Properties. Add the following ju:catalog-entry in the list of existing ju:catalog-entry nodes.

```
<ju:catalog-entry>
  <dc:title>pure::variants Integration</dc:title>
  <dc:description>Enables adding variability information to requirements in a DOORS NG module.
  Enables editing of Restrictions and Calculations. Also provides the functionality of creating
  a Variability Preview.</dc:description>
  <ju:gadget rdf:resource="https://server.local.com:8443/pvwidget/pvscl.xml" />
  <ju:icon rdf:resource="https://server.local.com:8443/pvwidget/pv.png" />
  <ju:preview rdf:resource="https://server.local.com:8443/pvwidget/preview.png" />
  <ju:thumbnail rdf:resource="https://server.local.com:8443/pvwidget/thumbnail.png" />
  <ju:category>pure::variants</ju:category>
  <ju:category>Requirements</ju:category>
</ju:catalog-entry>
```

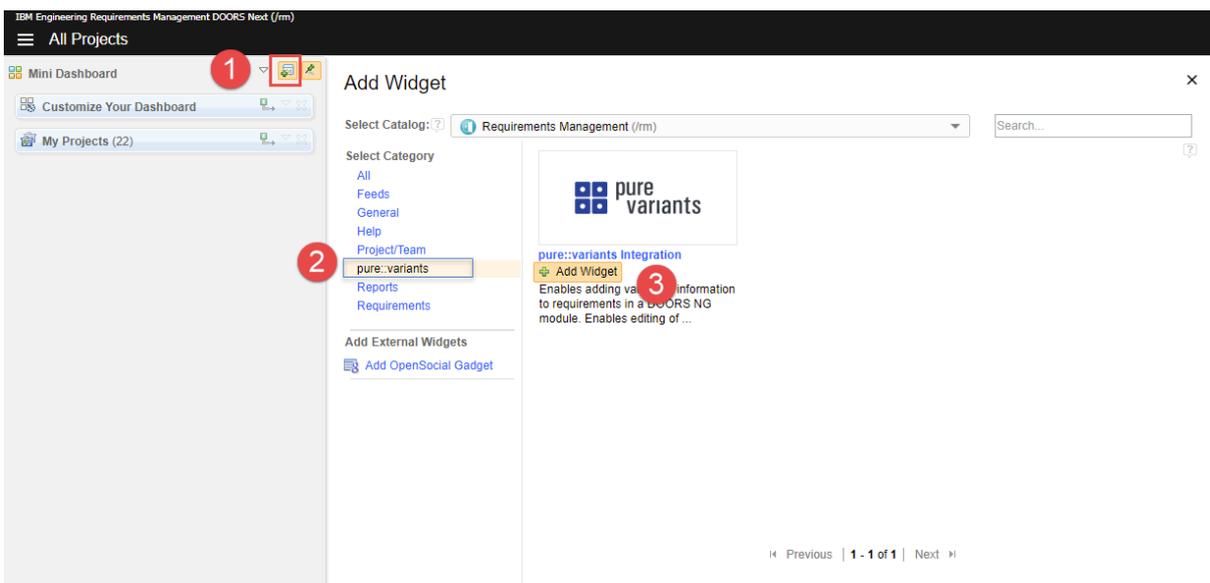
Note

Please note that the starting URL in the ju:gadget, ju:icon, ju:preview and ju:thumbnail depends on the deployment of the pure::variants Integration. Try to access the ju:gadget URL in browser and see if the pvscl xml document is shown or not. If the xml document is shown in the browser then the pure::variants widget should be available in the 'Add Widget' dialog.

6.1.6. Add pure::variants Integration to DOORS NG

In order to interact with the Integration, it needs to be added inside Jazz' *Mini Dashboard*. Hence, open the *Mini Dashboard* on the left side panel inside DOORS NG. At the top left of Mini Dashboard, click the **Add Widget** button. A dialog opens as shown in the figure [Figure 59, "Adding Integration inside Mini Dashboard"](#).

Figure 59. Adding Integration inside Mini Dashboard



On the left side of the dialog, in the widget category, click **pure::variants**. Then under the pure::variants Integration click the **Add Widget**.

For further instructions see section *Adding an OpenSocial gadget* in the documentation of IBM's *Rational Collaborative Lifecycle Management*

Note

If **pure::variants Integration for DOORS NG** is going to be used in **Web Hub** mode with **Jazz V7.0** then it is required that the pure::variants Web Components' relative domain-name must be same as DOORS NG's relative domain-name (e.g. *.**local.net**). For example, if DOORS NG is accessible via `https://server.local.net:9443/rm` then pure::variants Web Components must be accessible via URL like `https://otherserver.local.net:8443/pv`.

If a common domain-name is not possible, the WebComponents should be configured regarding Same-Site attribution for Cookies. For this, please see the optional step in our Tomcat resp. WebSphere configuration guide.

6.1.7. Check-up list for a successful deployment

The following list contains all necessary steps to be have taken to make successful deployment of the DOORS NG integration along with the DOORS NG application:

- Verify the web application server is started with a Java/JDK/JRE, which is not provided by IBM

Hint: Choose any other Java vendor, like Oracle, Amazon, Adoptium, OpenJDK instead.

- Verify the web application server is configured to be accessed via HTTPS/SSL endpoint.

Hint: Try to access via browser, using the HTTPS scheme, e.g. `https://<server:port>/<widget>/pvscl.xml`

- Verify the web application server is configured to have the SSL certificate of the DOORS NG application trusted.

Hint: See section *Configuration of Trust Store* ([Apache Tomcat](#) or [WebSphere Liberty](#)) for defining a truststore, which contains the DOORS NG application's SSL certificate.

- Verify the web application server can resolve the hostname of the DOORS NG application's URL.

Hint: Log-in to the web application server's machine, and try to follow the link of DOORS NG application URL, with help of browser application, or any other command-line tool, like `wget` or `curl`.

- Verify the web application server has deployed the DOORS NG integration functional, by accessing the the following URL: `https://<server:port>/<widget>/vel` (Please substitute with proper server and port, if other than 443, and use the appropriate context-path).

Hint: If successful, you should see a webpage showing the following line:

```
{"errorClass":"","errorMessage":"Module URI not received","errorCode":400}
```

