pure::variants Setup Guide

Parametric Technology GmbH

Version 7.0.0.685 for pure::variants 7.0

2025

Table of Contents

1. Introduction

This setup guide describes how to install and update pure::variants Enterprise deployments in an organisation's IT landscape. Its intended main audience are tool administrators.

Depending on the engineering tool landscape in place and on the desired data management mode for pure::variants models, the pure::variants deployment consists of some or all of the components depicted as orange boxes in figure Figure 1, “The Big Picture”. Blue boxes denote required components or services not developed or provided by Parametric Technology.

In this setup guide setup and update of all of these pure::variants components is described. The components are grouped from IT perspective into components running on server infrastructure, and components running on the personal desktop client machines of users.

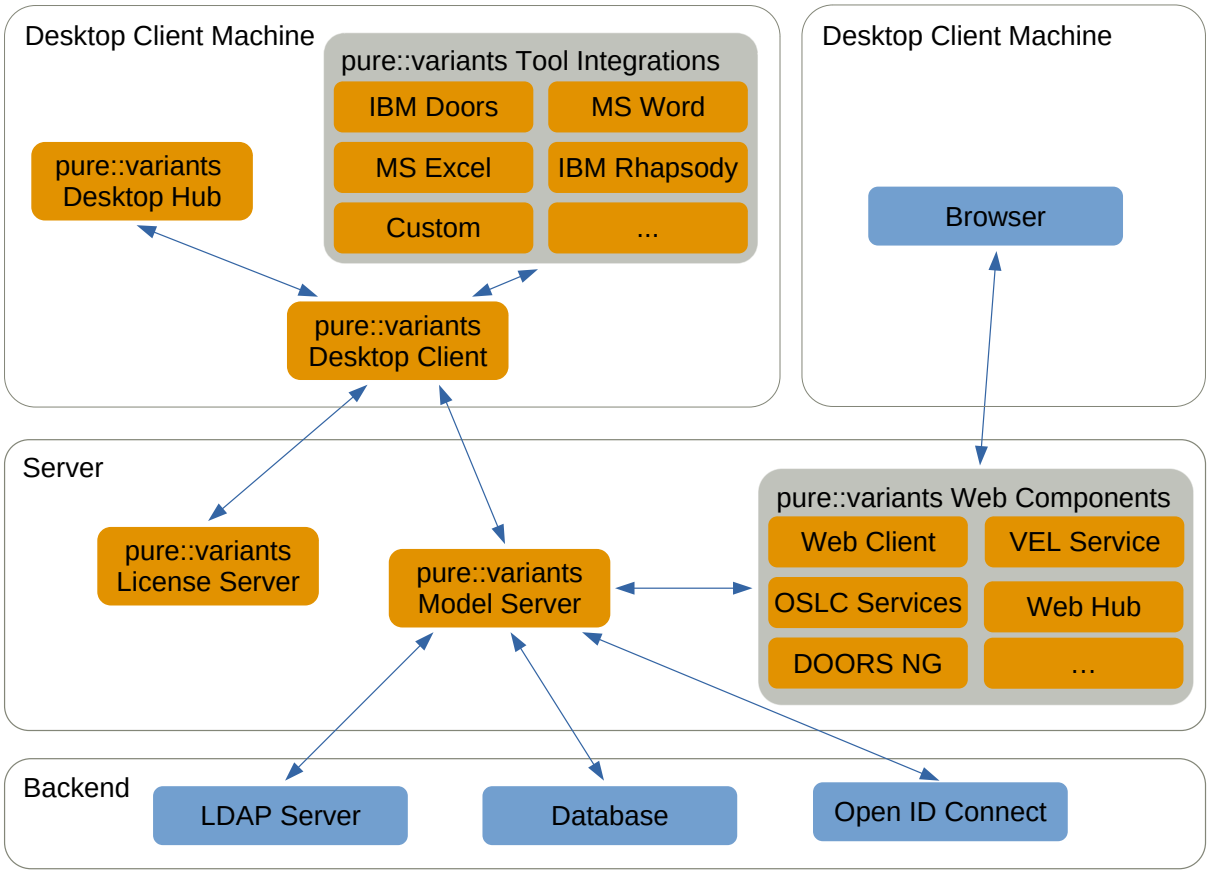
Every pure::variants Enterprise deployment requires the installation of the pure::variants Desktop Client on at least one system for administrative purposes (Section 3, “pure::variants Desktop Client”) and the installation of the pure::variants License server (Section 4, “pure::variants License Server”).

pure::variants Enterprise supports local, file-based operation and the centralized storage of pure::variants projects on a pure::variants Enterprise Model Server. Both modes of operation are supported by the pure::variants Desktop Client. The pure:::variants Enterprise Web Client supports only the centralized storage on a pure::variants Model Server.

If based on the organisation's needs the pure::variants Model Server, the pure::variants Enterprise Web Client or other services provided by the pure::variants Web Components have to be deployed, the recommended approach is to make use of the pure::variants Docker Deployment templates (Section 5, “pure::variants Deployment Templates for Docker”). These provide a streamlined approach to deploy and update the required components and services.

The following sections provide detailed insights in each of the individual pure::variants Deployment components 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. pure::variants Desktop Client

- Operating System

  Only 64-bit operating running on x64 systems are supported:

  - Windows 10, 11
  - Windows Server 2016, 2019, 2022
  - Linux with X11 Window System installed
  - Mac OS
    - For Macs with Apple Silicon Rosetta 2 needs to be installed.

- Software
  - Oracle Java SE or OpenJDK

    Supported Java versions:

    - Java 17 - 23
  - 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 4.24 – 4.34
- Memory
  - Min: 4 GB
  - Recommended: 8 GB
- CPU
  - Min: Dual Core CPU
  - Recommended: 4 CPU cores
- HDD
  - Min: 10 GB free disk space

2.2. Model Server with Database

- Operating System

  Only 64-bit operating running on x64 systems are supported:

  - Windows 10, 11
  - Windows Server 2016, 2019, 2022
  - Linux
- Software
  - Oracle 19c -23c
  - MSSQL Server 2016 - 2022
  - PostgreSQL 13.x - 17.x with PostgreSQL ODBC Driver version 13.x or newer.
- Memory
  - Min: 4 GB
  - Recommended: 8 GB + plus 1GB per active user
- CPU
  - Min: 4 CPU cores
  - Recommended: 4 CPU cores (up to 10 active users) and + 1 CPU core for each additional 10 active users.
- HDD
  - Min: 10 GB free disk space
- Database Storage Requirements
  - MSSQL: 20 MB storage space per 1000 elements and revision

- Oracle: 4 MB storage space per 1000 elements and revision
- Postgres: 15 MB storage per 1000 elements and revision

2.3. License Server

- Operating System

  Only 64-bit operating running on x64 systems are supported:

  - Windows 10, 11
  - Windows Server 2016, 2019, 2022
  - Linux
- Memory
  - Min: 512 MB
  - Recommended: 1 GB
- CPU
  - Minimum and Recommended: Dual Core CPU
- HDD
  - Min: 10 GB free disk space

2.4. Web Components

- Operating System

  Only 64-bit operating running on x64 systems are supported:

  - Windows 10, 11
  - Windows Server 2016, 2019, 2022
  - Linux
- Software
  - Oracle Java SE or OpenJDK

    Supported Java versions:

    - Java 17 - 23
  - 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/).
  - Apache Tomcat 9.0.0 - 9.0.102 or WebSphere Liberty Kernel v19.0.0.6+
    - There are known issues with Apache Tomcat 9.0.88 and newer, compatibility is not ensured.
  - Supported Browsers: At most 6 month old versions are supported.
    - Chrome
    - Firefox

  - Edge
- Memory
  - Minimum: 2 GB
  - Recommended: 32 GB

    Each active user browser session requires per open model with a unique revision with 1000 elements about 100 MB memory.
- CPU
  - Min: 4 CPU Cores
  - Recommended: 4 CPU Cores (up to 10 active users) and + 1 CPU Core for each additional 10 active users.
- HDD
  - Min: 10 GB free disk space

3. pure::variants Desktop Client

The pure::variants Desktop Client 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 Desktop 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 Desktop Client you need to following:

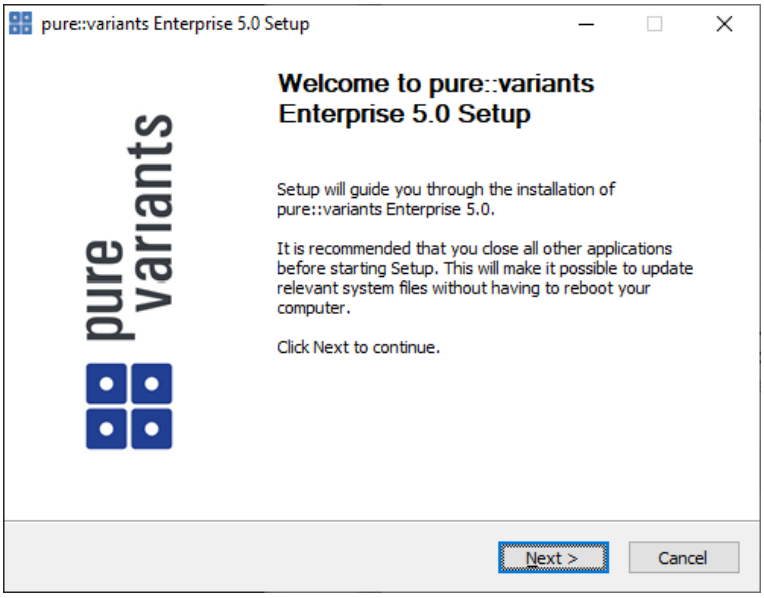
- pure::variants Desktop Client license
  - If a floating license is used additionally the license server URL is needed to be able to connect and obtain a license form the license server.
- pure::variants Desktop Client installer or pure::variants update site
- supported 64-Bit Java version installed

The Windows Installer can be download 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.

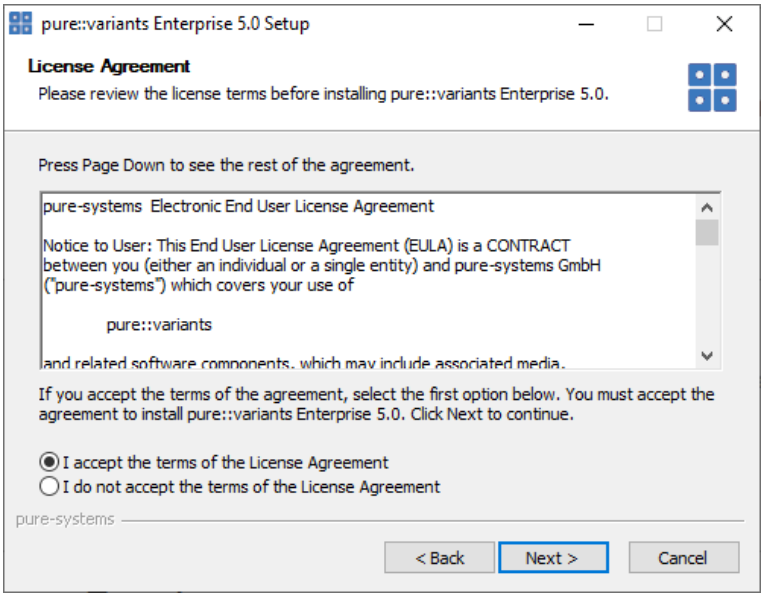
All pure::variants extensions available for the account are automatically included in the Windows Installer download. However, some may not be enabled by default in Installer. Make sure to select the desired extensions during the installation process. Later updates to the extension selection can be done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

Figure 2. pure::variants Desktop Client Installer



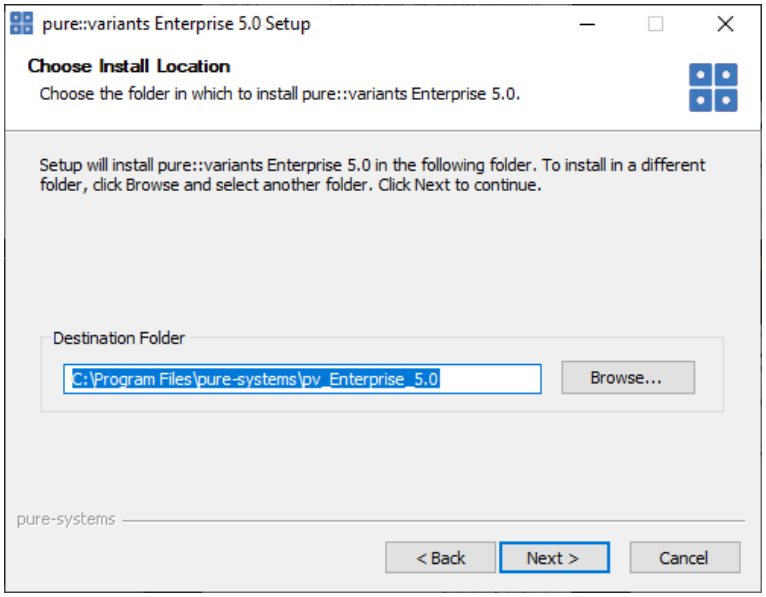
Click *Next*.

Figure 3. Setup pure::variants Desktop Client License



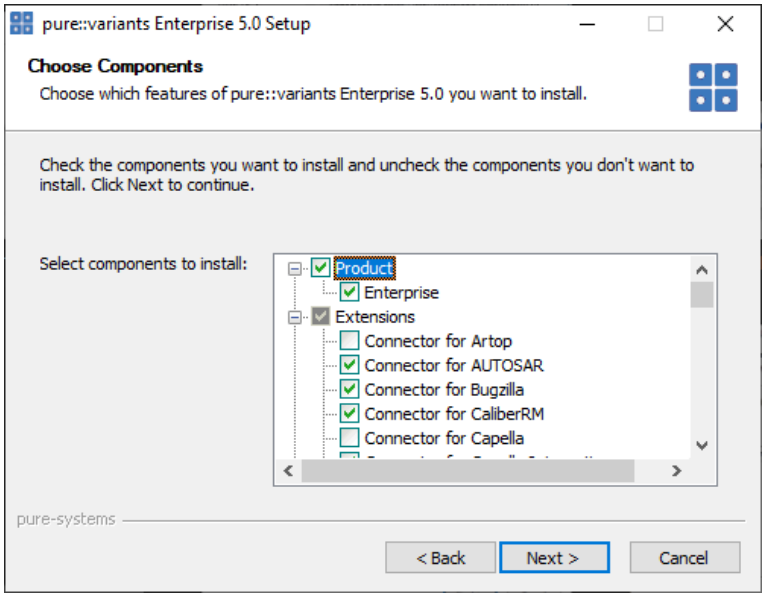
Read the license agreement and after accepting it click *Next*.

Figure 4. Setup pure::variants Desktop Client Installation Location



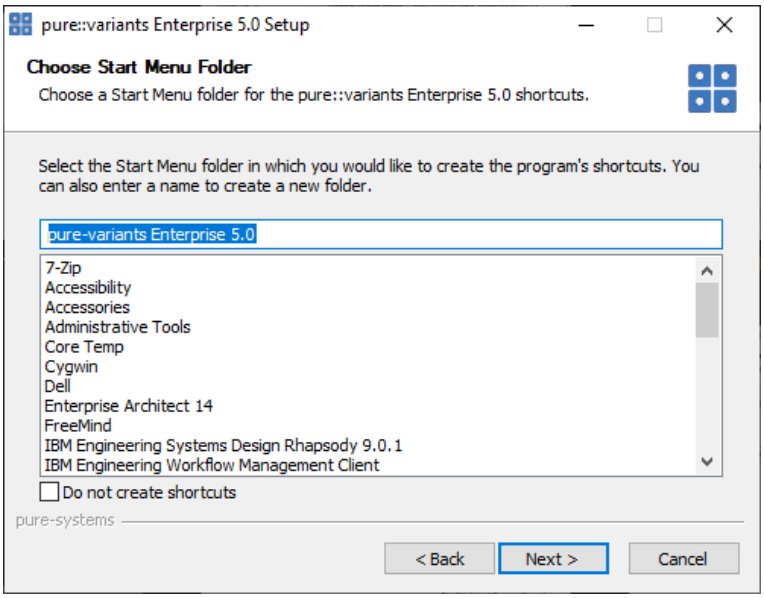
Select the folder where to install the pure::variants Desktop Client files. Click *Next*.

Figure 5. pure::variants Desktop Client Feature Selection



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

Figure 6. Setup pure::variants Desktop Client Start Menu

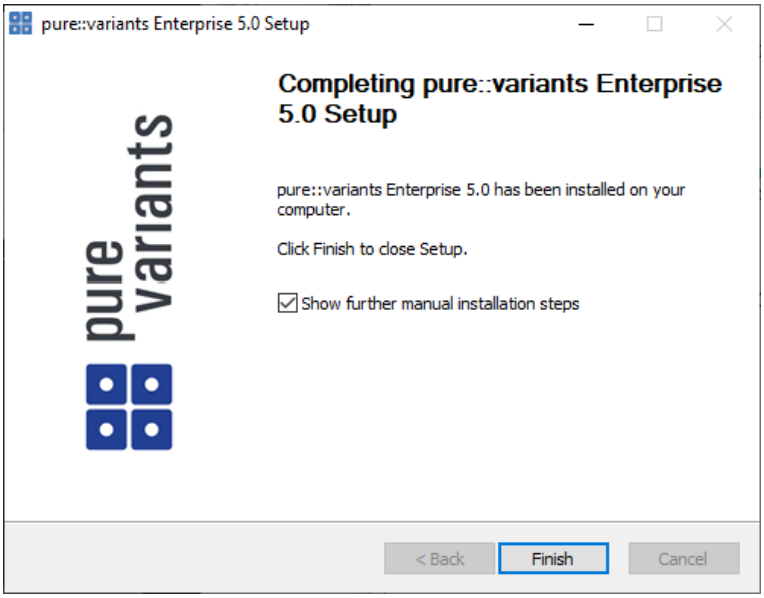


Enter the name for the Windows start menu entry, or disable the creation of the start menu entry. Click *Next*

The next pages may show information about pure::variants integrations, which are installed along with the pure::variants Desktop 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 Desktop 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 exsiting 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 eclipsec.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 Desktop Client installer has a silent mode. This mode installs the pure::variants Desktop Client without user interaction by just using the standard settings of the pure::variants Desktop 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 download 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.

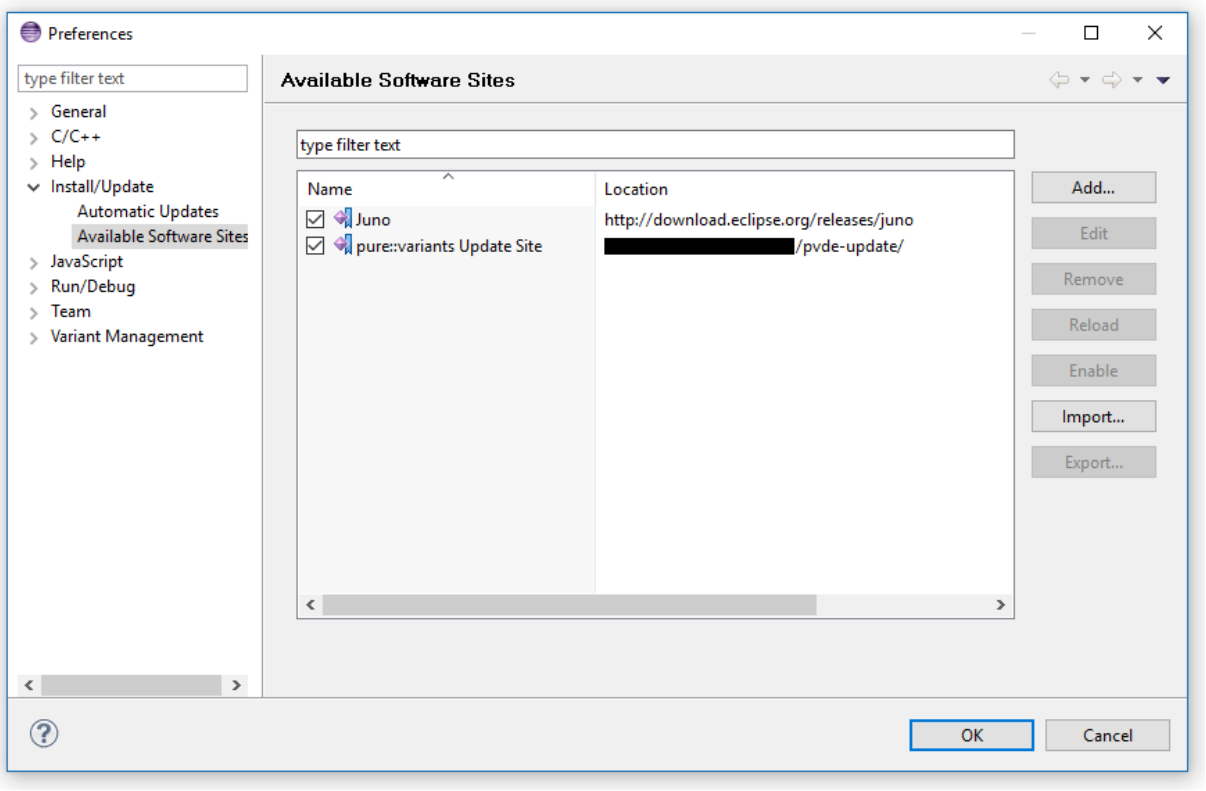
Using update site

- 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.

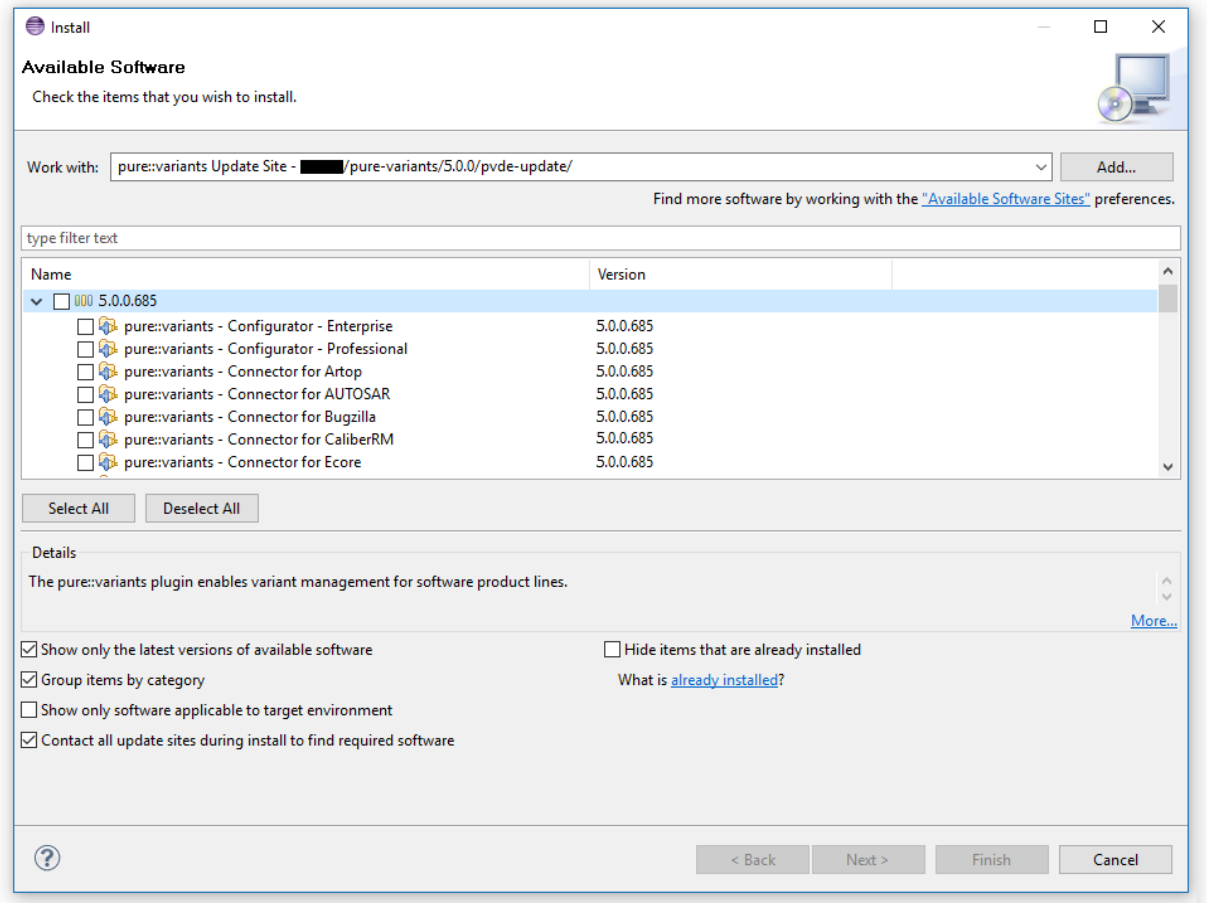
  The location of the site depends on the pure::variants product variant. Visit the Parametric Technology web site (https://www.pure-systems.com) or read your registration email to find out which site is relevant for the version of the software you are using.

Figure 8. Update Site Selection



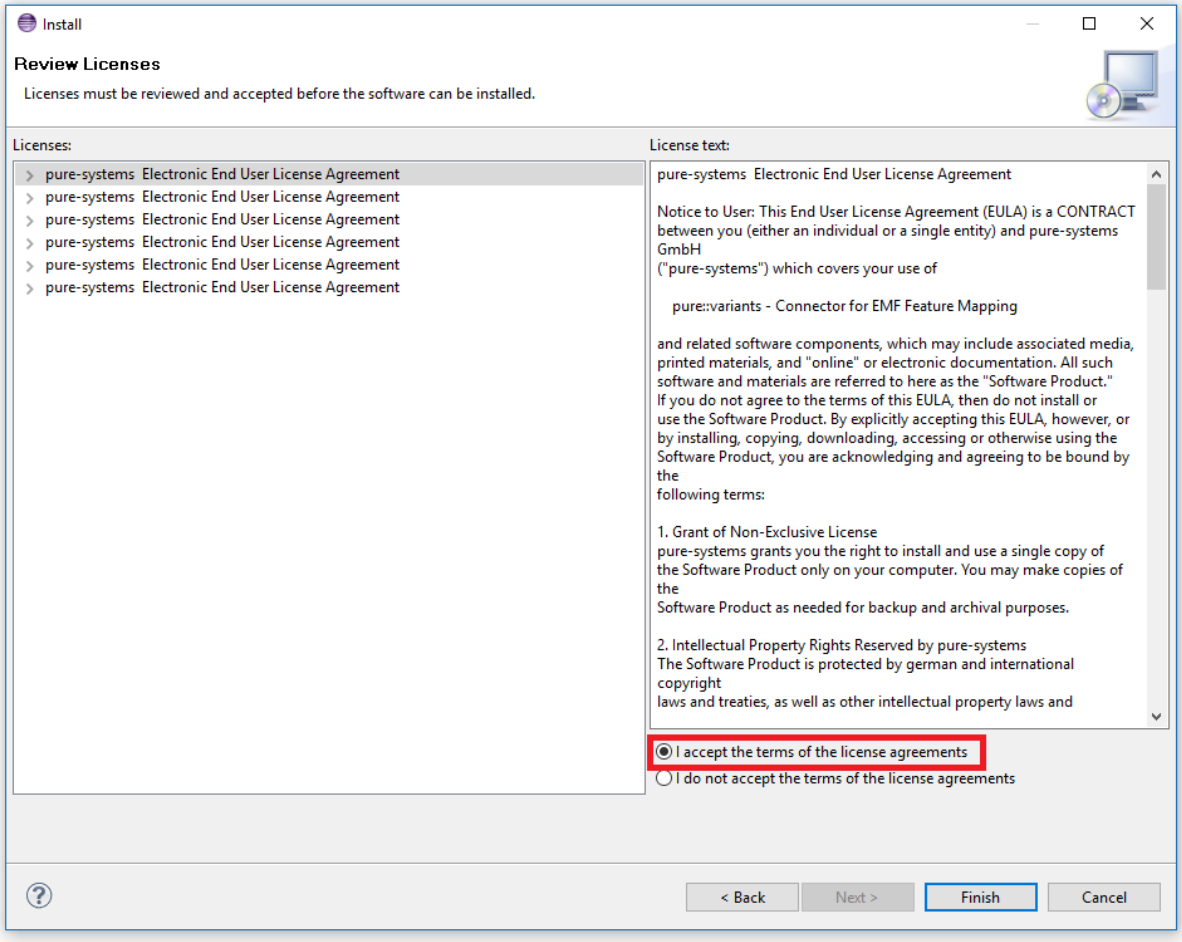
- Unfold the pure::variants update site and select all features to be updated. Select "Next".

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 Desktop Client

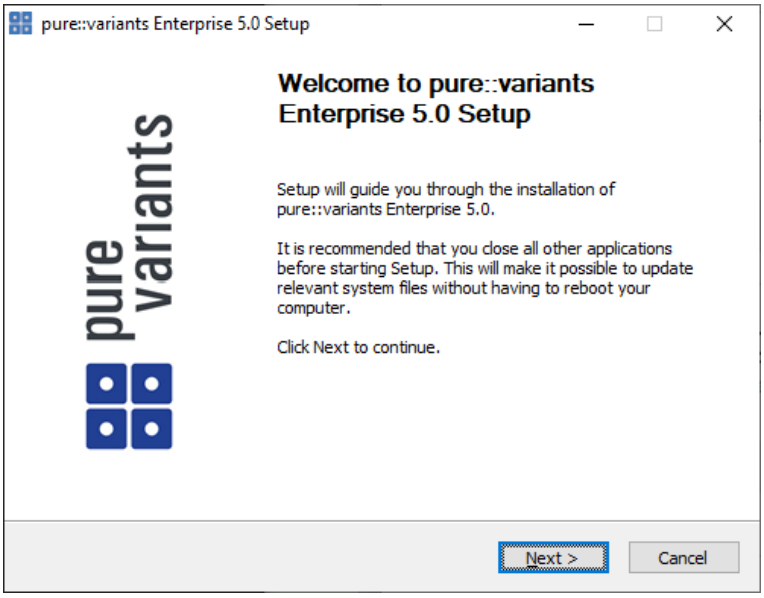
3.2.1. Update with pure::variants Installer

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.

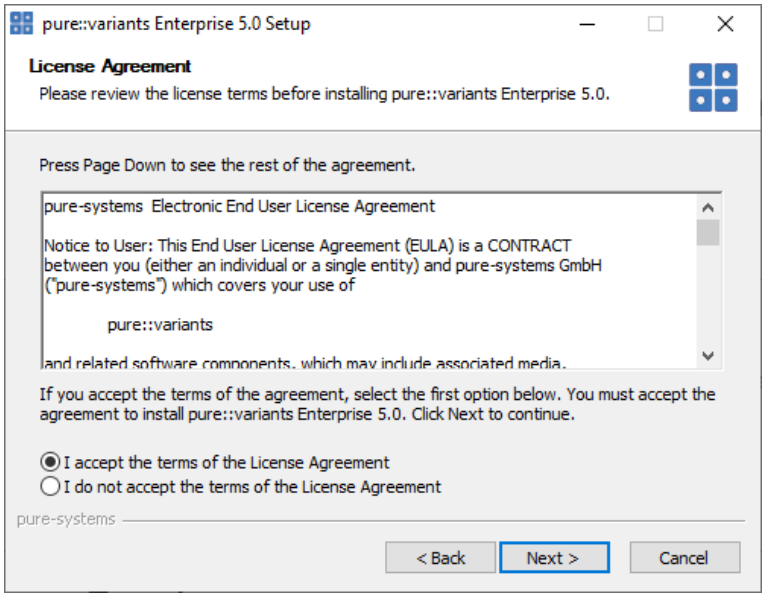
Download the installer package ("pure::variants Windows Installer Package") and extract it. The installer will check for an existing pure::variants Desktop Client installation and start in update mode if it finds one. Start the update by double-clicking "Setup Enterprise X.Y.ZZ.exe". Running the pure::variants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

Figure 11. pure::variants Desktop Client Installer



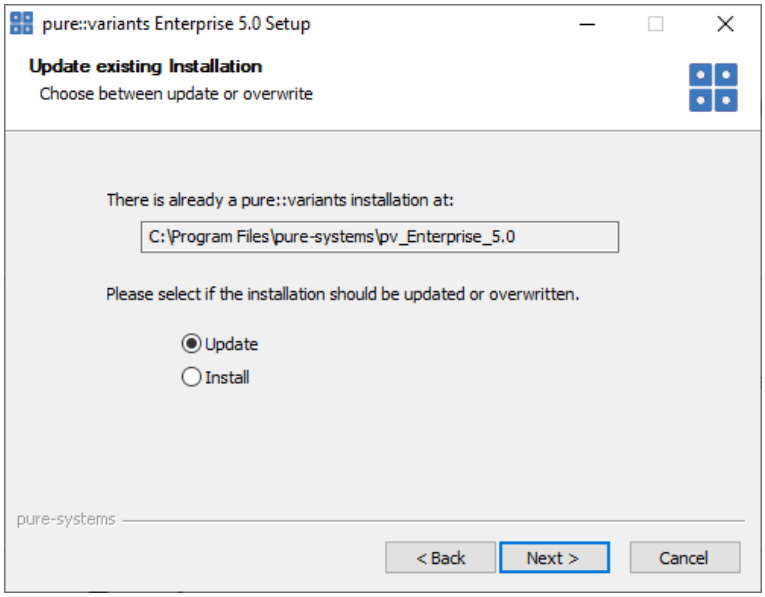
Click *Next*.

Figure 12. Setup pure::variants Desktop Client License



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

Figure 13. Choose Update Mode

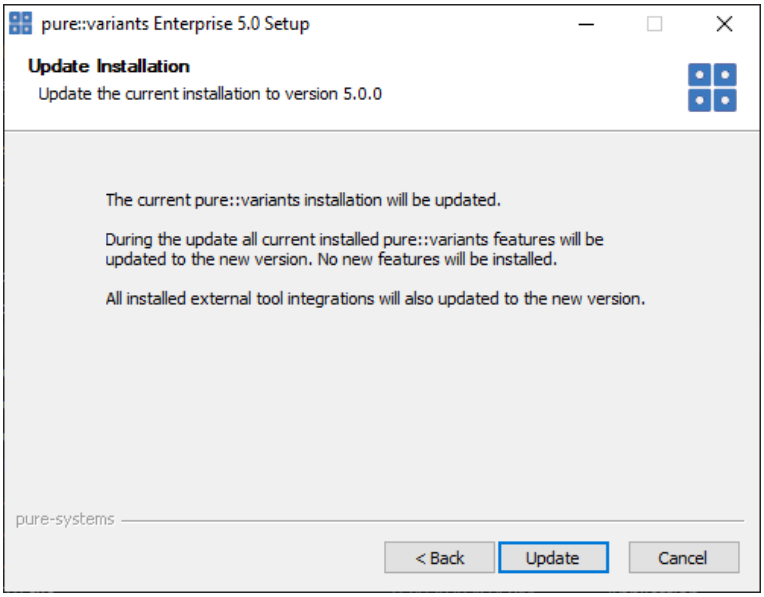


Choose *Update* if the current pure::variants Desktop Client installation shall just be updated with the same installed feature and settings. The installed pure::variants integrations will also be updated. The installed components cannot be changed. If a change of the installed components is wanted,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

Or choose *Install* if the current pure::variants installation shall be removed and a new fresh pure::variants Desktop 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.

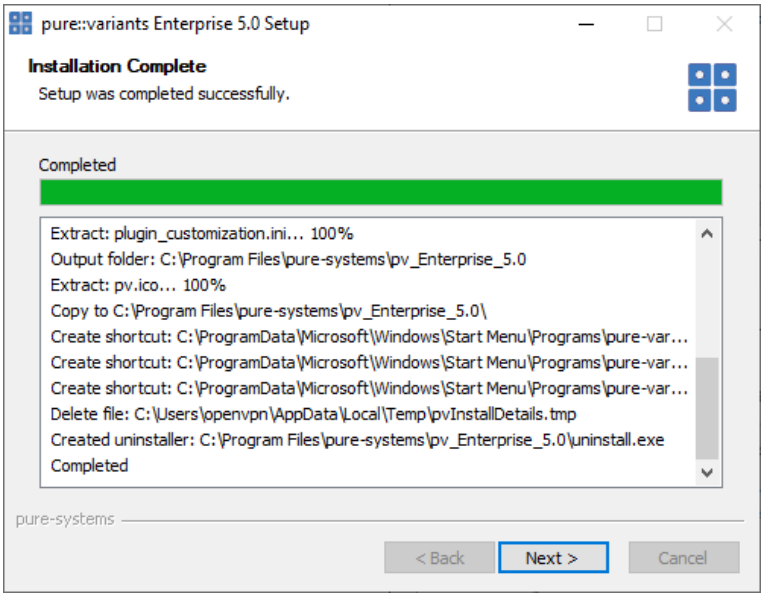
Click *Next*.

Figure 14. pure::variants Start Update



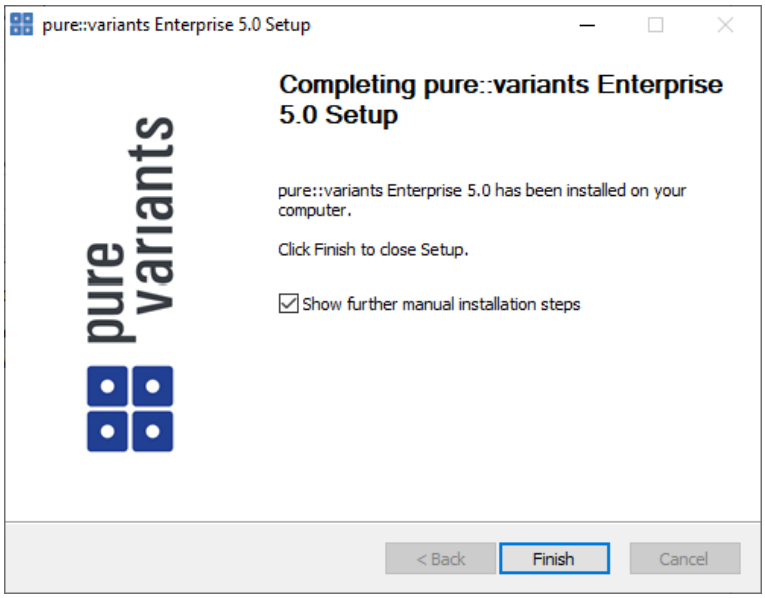
Click *Update* to start the update process.

Figure 15. pure::variants Installation Progress



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

Figure 16. Update pure::variants Desktop 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

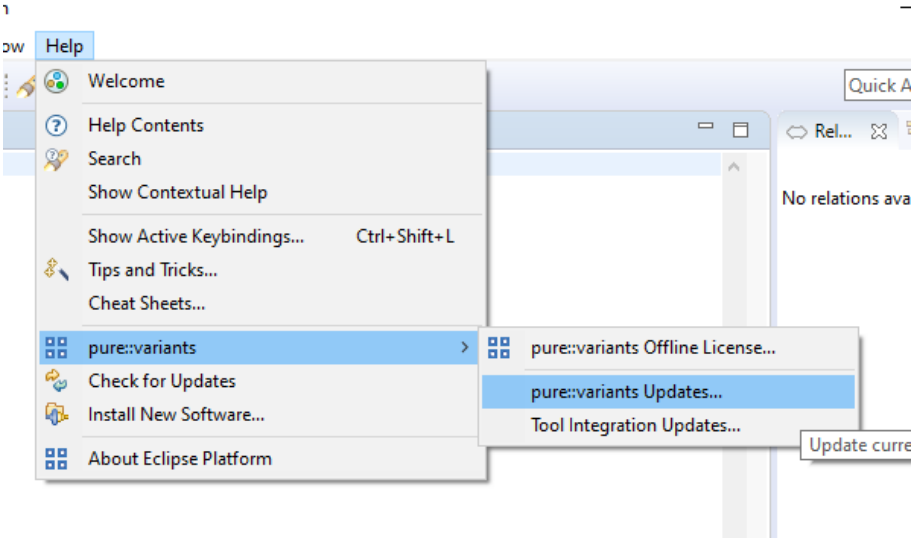
pure::variants has a built-in update action which can be used to perform an update with all the currently installed pure::variants extensions. This update action does not update the installed pure::variants integrations automatically. But they can be easily updated with the *Tool Integration Update* action. See Section 9.2, “Update pure::variants Tool Integrations” for the detailed description.

The update action requires administrator privileges.

Note

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

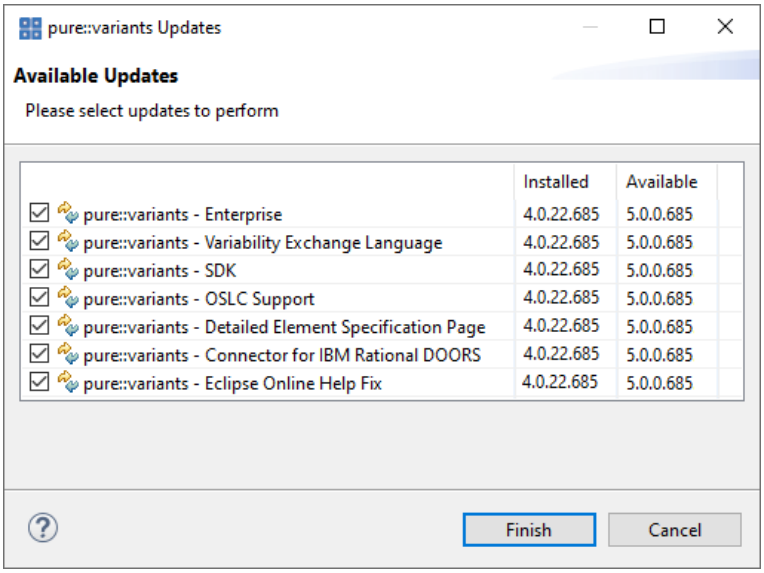
Figure 17. Start pure::variants Desktop Client Update



Start the pure::variants Desktop 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 Desktop 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 Desktop Client Update



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

Figure 19. Start pure::variants Desktop Client Update

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

3.2.3. Update with Eclipse package manager

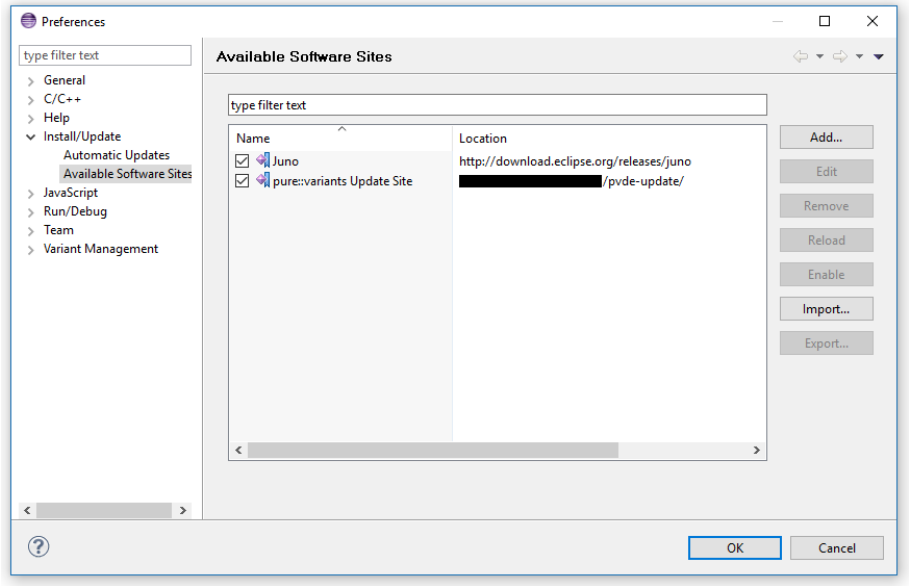
The quickest way to get a update for pure:.variants is to run the software updater inside pure::variants:

- 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.

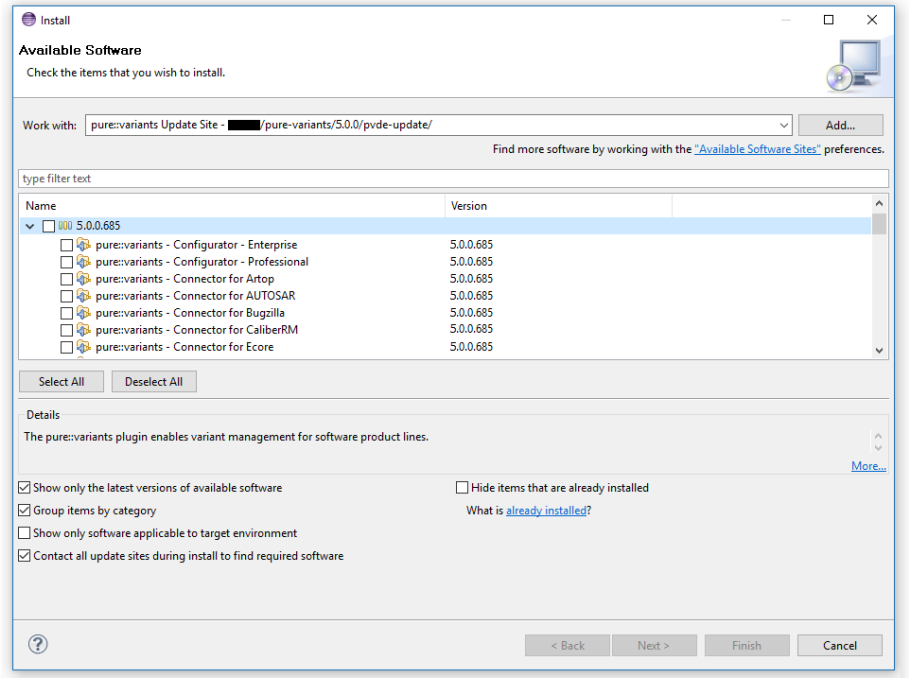
  The location of the site depends on the pure::variants product variant. Visit the Parametric Technology web site (https://www.pure-systems.com) or read your registration email to find out which site is relevant for the version of the software you are using.

Figure 20. Update Site Selection



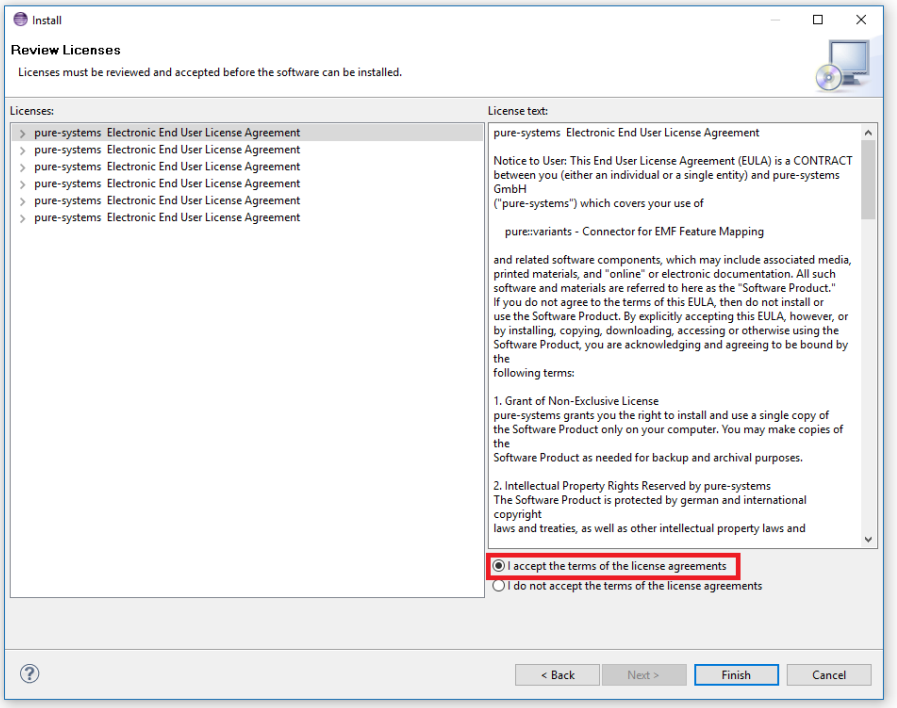
- Unfold the pure::variants update site and select all features to be updated. Select "Next".

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:

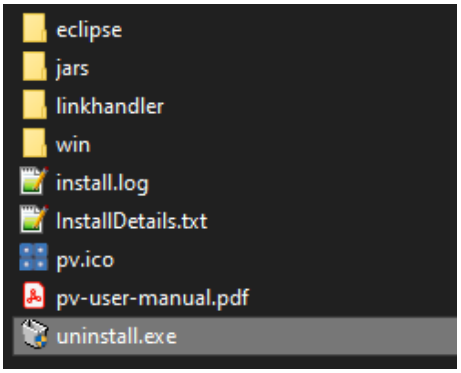
- Start pure::variants (or the Eclipse into which pure::variants has been installed).
- Select "Help"->"Software Updates"->"Find and Install...".
- Select "Search for new features to install" and "Next".
- Click on button "Archived Update Site" or "Local Update Site".
- Use "Browse" to select the downloaded 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 Desktop Client

3.3.1. Uninstall using pure::variants Uninstaller

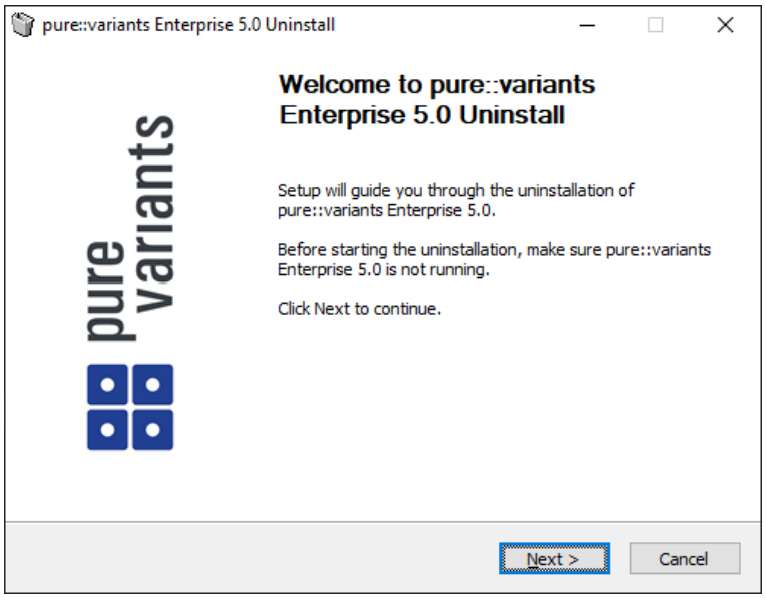
The uninstaller for the pure::variants Desktop Client can be started in two different ways. The first is to go to the Windows *Add or remove programs* application and search for *pure::variants Enterprise* and start the uninstaller by using the *Uninstall* action. The uninstaller requires Administrator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

Figure 23. pure::variants Desktop Client Uninstaller



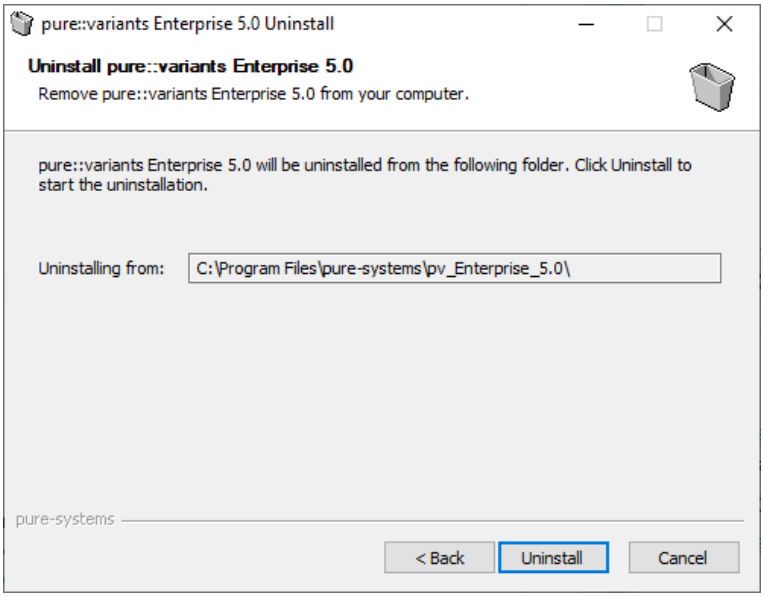
The second way is to navigate to the pure::variants Desktop Client installation folder and start the uninstaller by double clicking it.

Figure 24. pure::variants Desktop Client Uninstaller



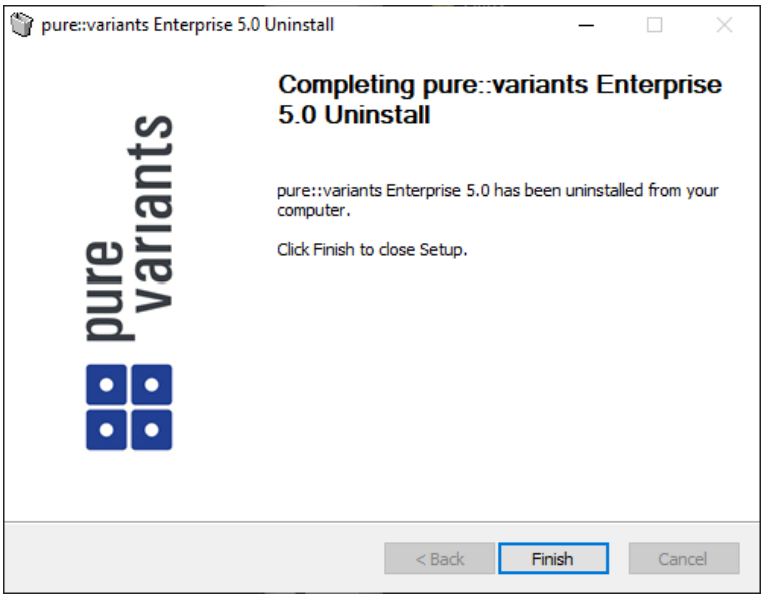
Click *Next*.

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

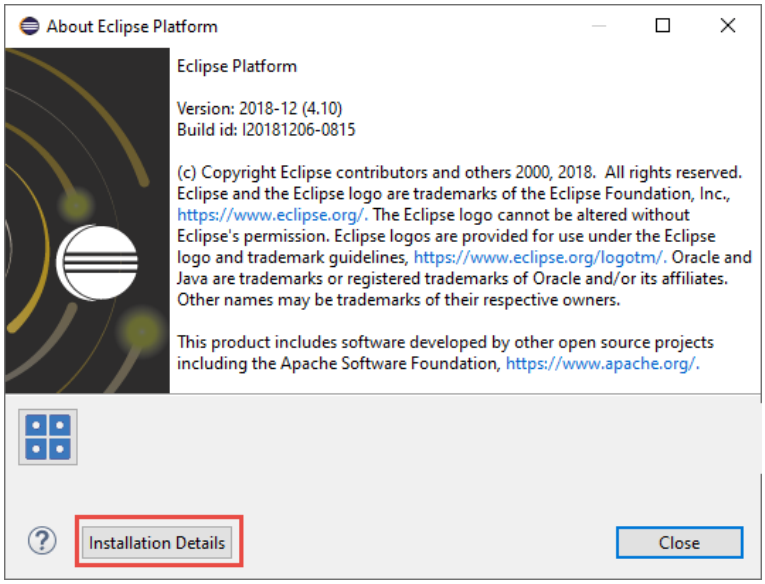
There are two ways to remove pure::variants from an Eclipse instance. You can use the Eclipse command line or remove the pure::variants features one by one in the running Eclipse Instance. Either way a cleanup of the Eclipse instance has to be performed afterwards.

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

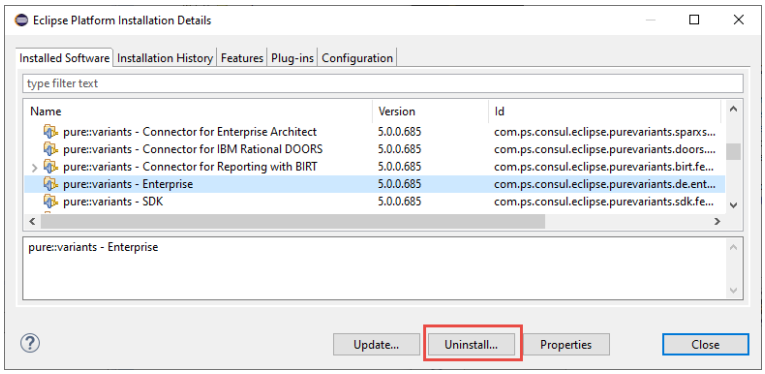
To remove an installed feature from Eclipse using the Eclipse client, open the *About Eclipse Platform* dialog with the *About Eclipse Platform* action in the *Help* menu.

Figure 27. Eclipse About Dialog



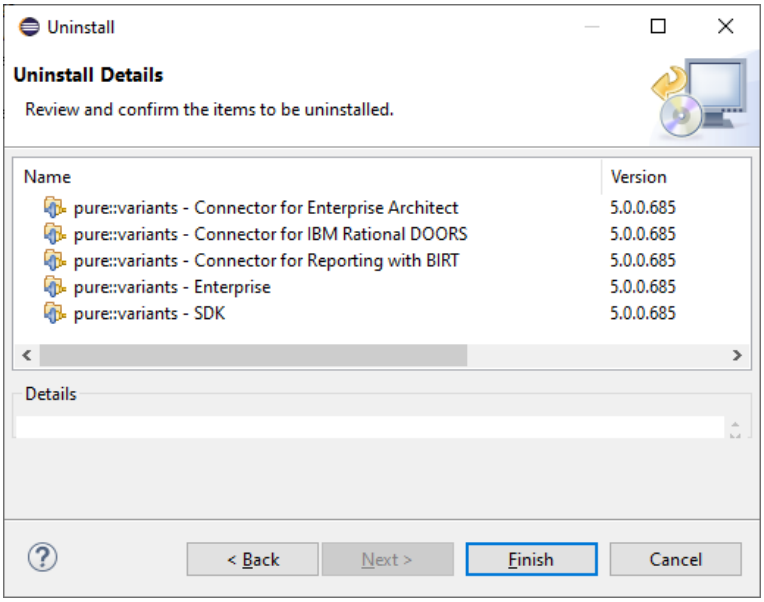
Use the *Installation Details* button to access the installation details.

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

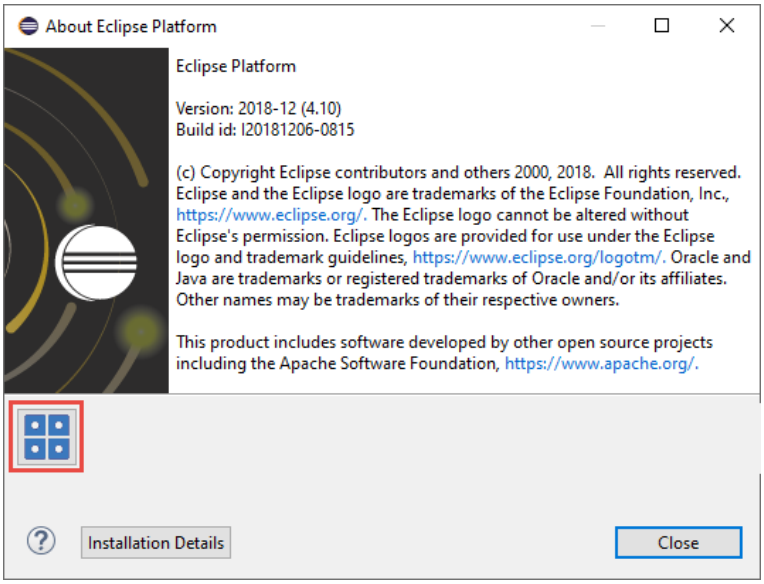


Click *Finish* to start the uninstall process. After it finished, Eclipse will prompt you to restart the application. Click *Restart* to finish the uninstallation.

Uninstall pure::variants using Eclipse uninstall application

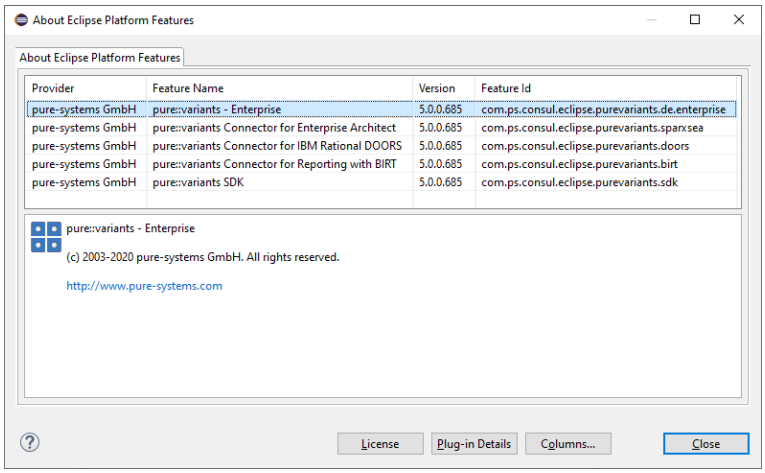
To use the uninstall application you need the feature ids of the features you want to uninstall. The feature ids can be found in the *About Eclipse Plattform* dialog. Open the dialog with the *About Eclipse Platform* action in the *Help* menu.

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.sparxsea.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>\eclipsec.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\<user name>\AppData\Roaming\pure-variants-6, and the second C:\ProgramData\pure-variants-6. On Linux based systems the pure-variants-6 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>\eclipsec.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
```

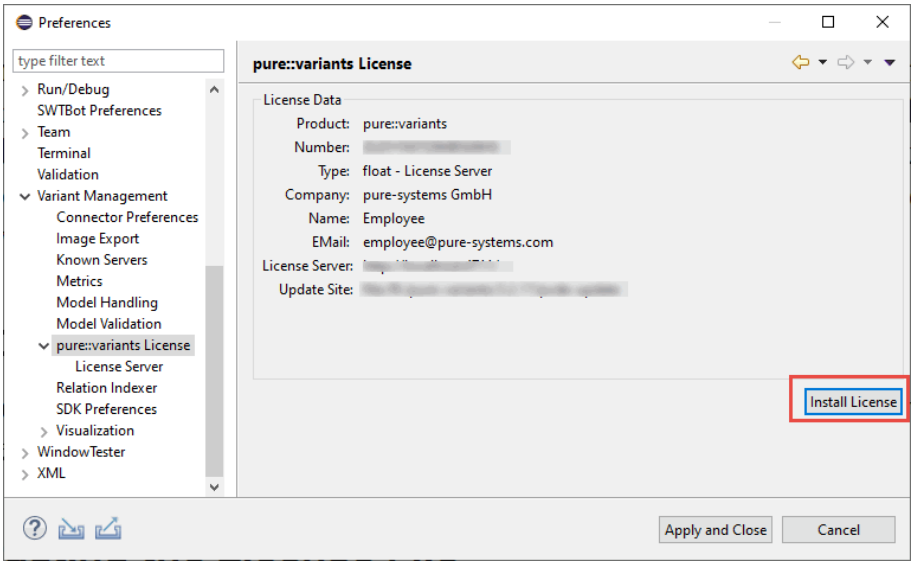
3.4. Basic Setup of the pure::variants Desktop Client

3.4.1. Setup a pure::variants Desktop 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

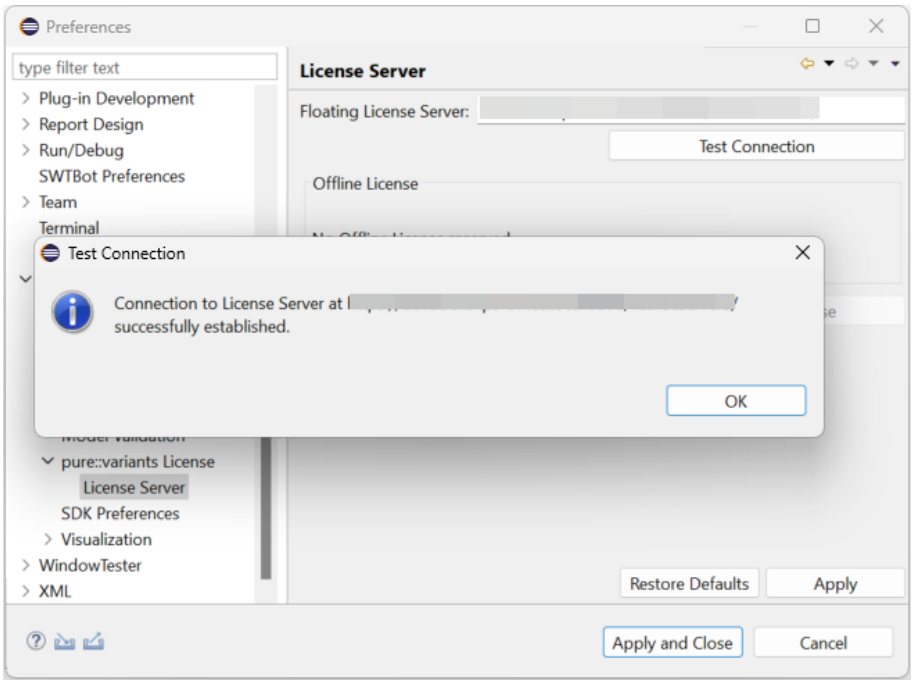


The pure::variants client license can also be defined using the *PVLICENSE* environment variable. This variable has to define the fullpath to a license file. If this variable is defined the given license file is used and the user does not need to define the license for pure::variants client instances.

The next step is necessary only if a floating license with a pure::variants license server is used. The license server URL can be provided with the floating license file, or it has to be set by the user.

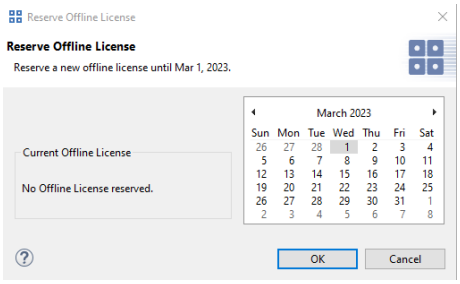
To set the license server URL, open the sub page **Variant Management -> pure::variants License -> License Server** and enter the URL of the license server into the **Floating License Server** text field, click button **Test Connection** to check that the connection is successful.

Figure 33. pure::variants License Server Preferences



Click button **Reserve Offline License** to choose how long the license shall be reserved.

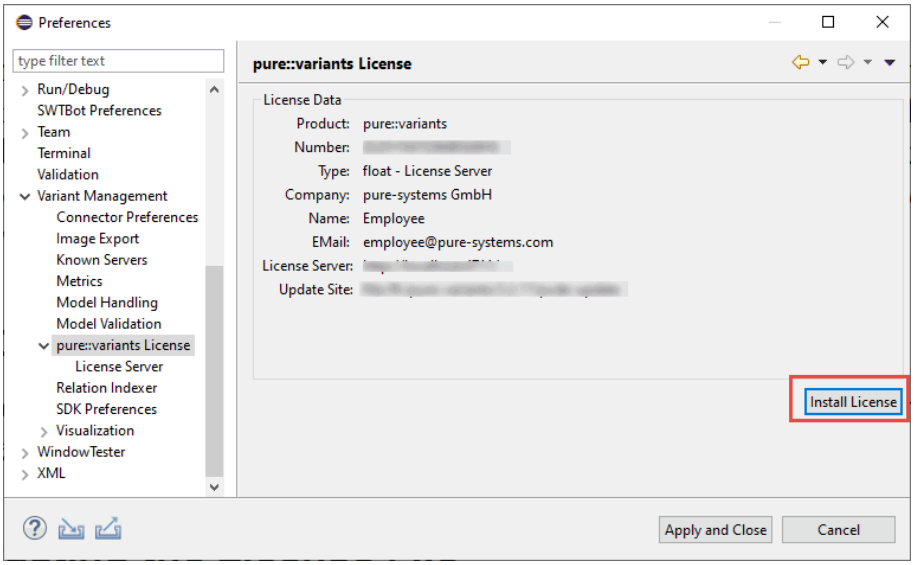
Figure 34. Reserving Offline License



3.4.2. Update a pure::variants Desktop 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 35. pure::variants License Preferences



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

For central or automatic deployed pure::variants Desktop Clients it may be necessary to also automatically deploy or update the pure::variants Desktop 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 License Server

There are two alternative ways to install pure::variants License Server, depending on your operating system. The different installation procedures are described below.

4.1. Installation Requirements

To be able to succesfully install the pure::variants license server you need to following:

- pure::variants server license
- pure::variants license server installer or pure::variants license server archive
- If you want to encrypt the communication with the license server a certificate is necessary
  - Please see Section 7.5.4, “HTTPS Server Options” for further information

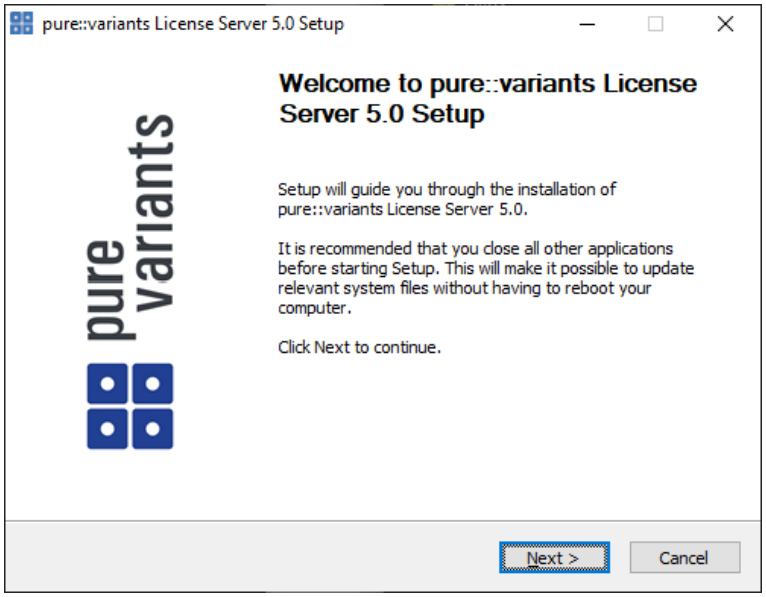
4.2. Install pure::variants License Server

4.2.1. Install with Windows Installer (Windows only)

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.

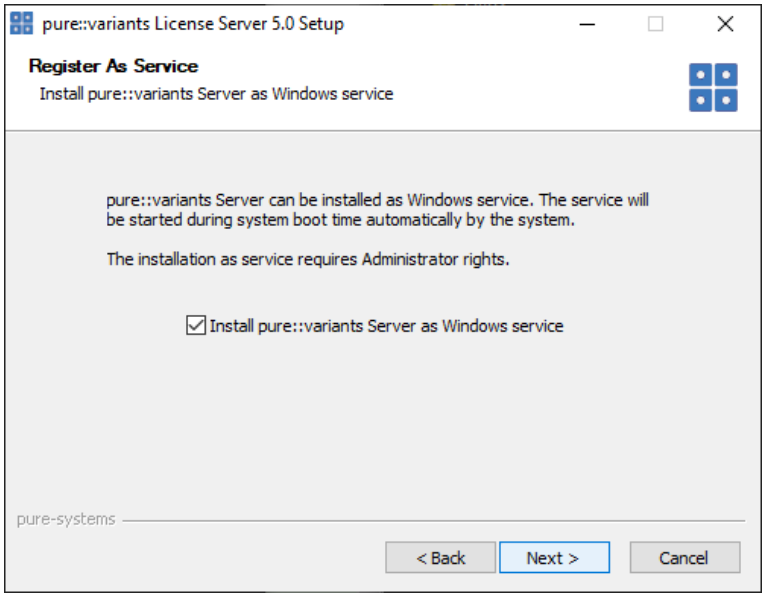
Download the license server installer “Setup License Server X.Y.ZZ.exe” and start the installation by double-clicking the executable. Running the license server installer requires Administrator privileges.

Figure 36. Setup License Server Installer



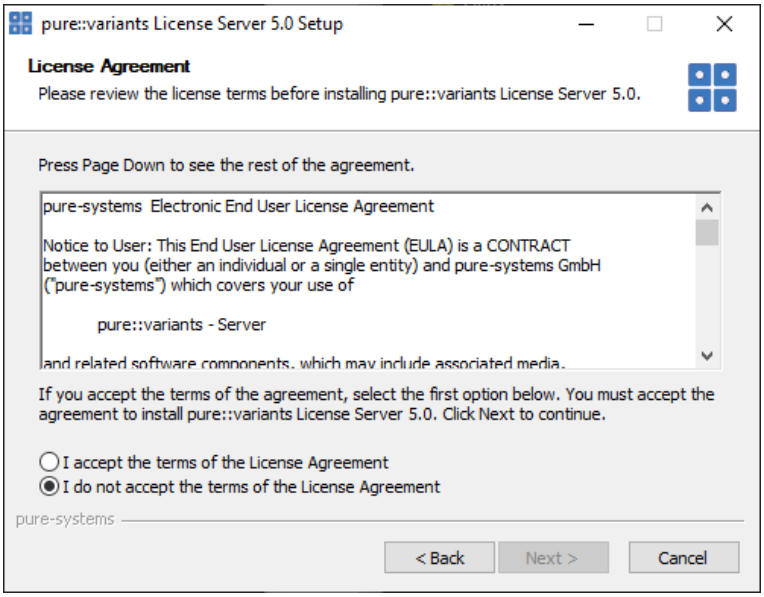
Click *Next*.

Figure 37. Setup License Server Windows Service



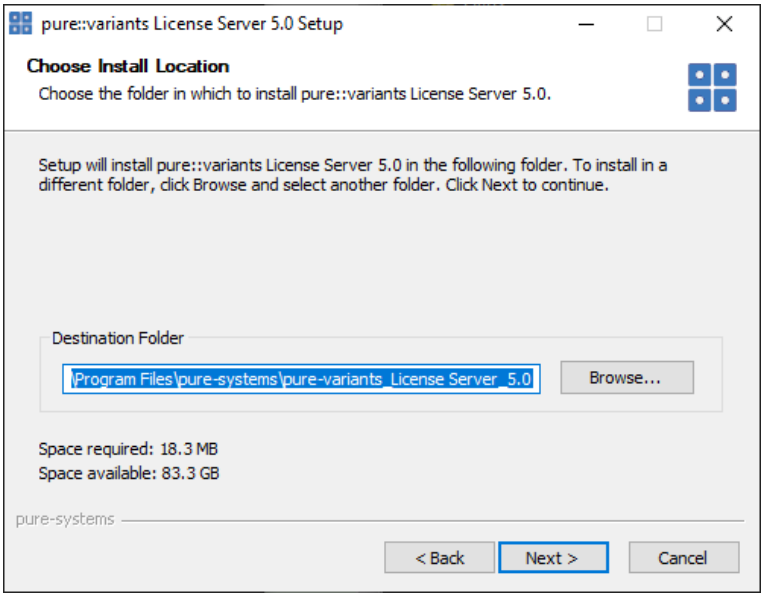
Decide whether the license server should run as Windows service. It is recommended to run the license server as Windows service. This will ensure that the license server will be started automatically during system startup. Click *Next*.

Figure 38. Setup License Server License



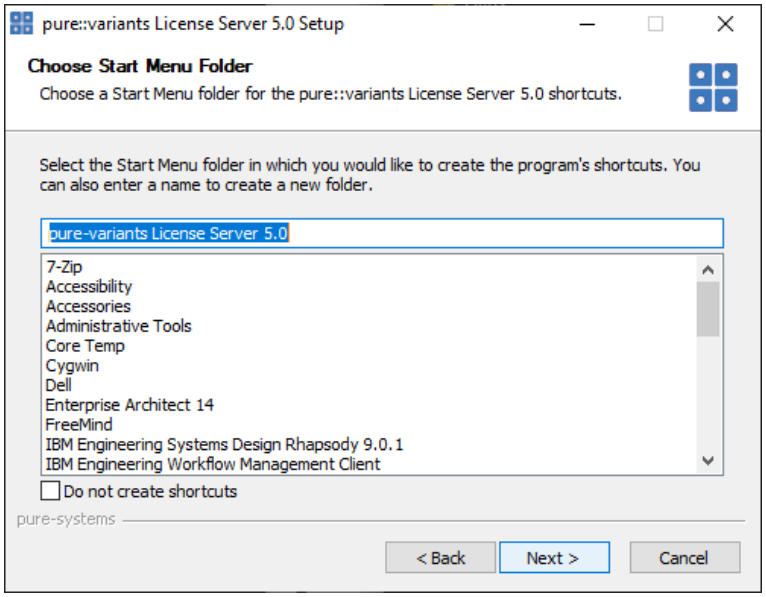
Read the license agreement, and after accepting it click *Next*.

Figure 39. Setup License Server Installation Location



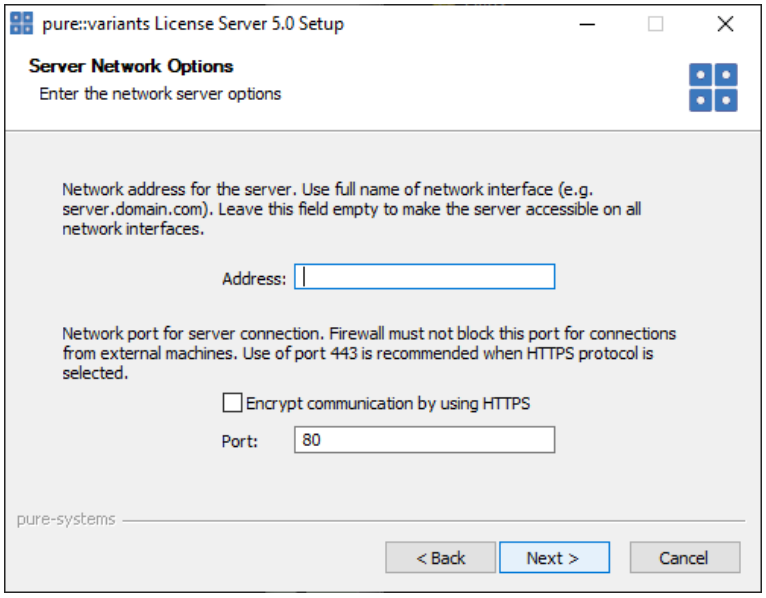
Select the folder where to install the license server files. Click *Next*.

Figure 40. Setup License Server Start Menu



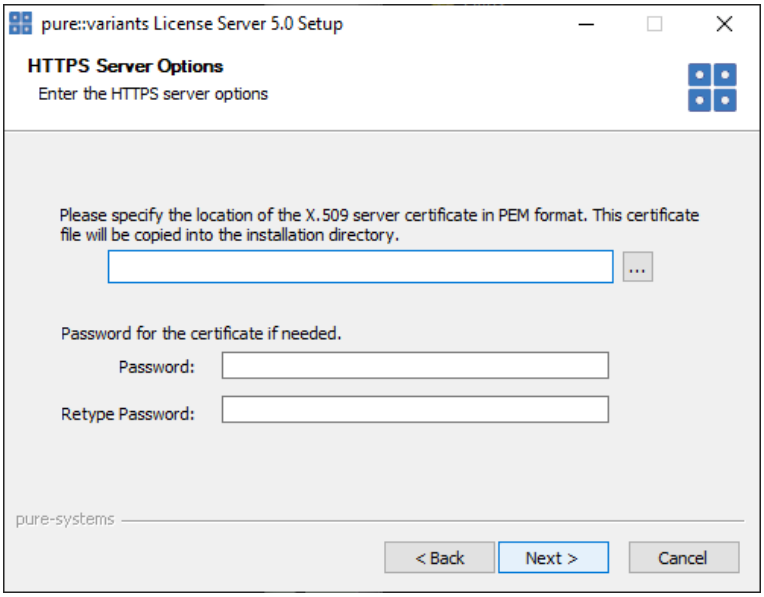
Enter a name for the Windows start menu entry, or disable the creation of the start menu entry. Click *Next*.

Figure 41. Setup License Server Network Options



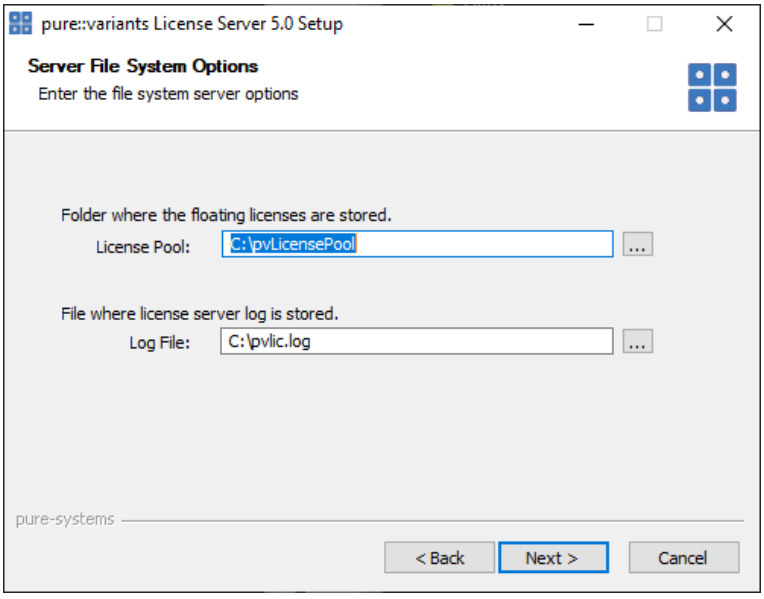
Now configure the network options. The *Address* field specifies the hostname or IP address on which to make the license service available. If you leave this field empty, the license server will automatically be available on all available network interfaces. You can encrypt the communication with the license server by selecting the *Encrypt* option. This will enable HTTPS instead of HTTP. This requires an X.509 certificate for the license server. The certificate setup is done on a separate page in the installer. The port for communication can be specified in the *Port* field. The default values (80 for HTTP and 443 for HTTPS) should work best. Please ensure that the chosen port is not blocked by a firewall or used by another service. Click *Next*.

Figure 42. Setup License Server Encryption Settings



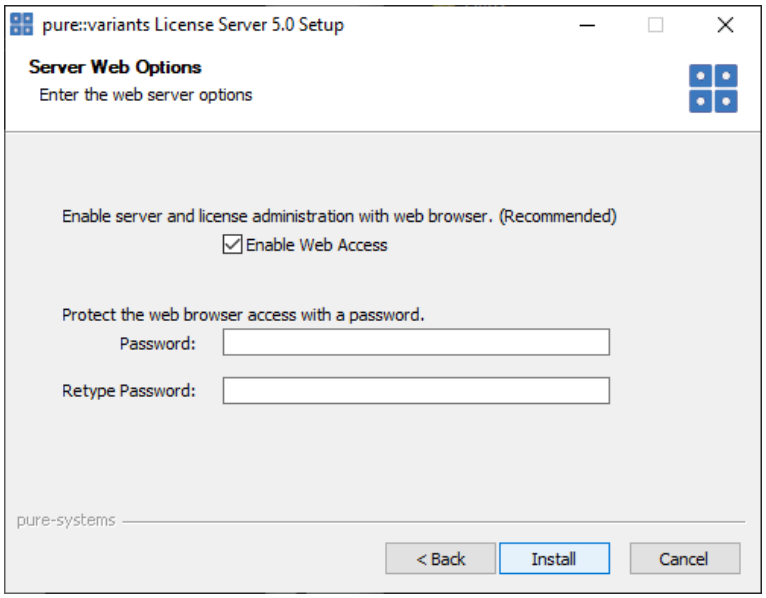
If encryption was enabled, enter the path to the X.509 certificate into the upper file field. The format of the certificate needs to be PEM. No other types are supported. If the certificate is protected by a password, enter the password into the two password fields. Click *Next*.

Figure 43. Setup License Server Licenses Options



Please select a directory where the license files will be stored. You can place your server license file(s) in this folder now, or use the license server's web interface later to install the licenses. Specify also a location for the server log file. Click *Next*.

Figure 44. Setup License Server Web Options



The license server provides a web interface. The web interface can be enabled or disabled on this page. It allows installing and updating licenses as well as the management of currently used licenses. Please secure the web interface with a password. If you do so, then ensure to also have HTTPS enabled. Otherwise the password will be transmitted not encrypted and could be spied on.

Click *Install* to start the installation process.

If the Windows service option was chosen, the license server will start automatically after successful installation.

4.2.2. Install from Archive

If using the installer is not an option, the server is also distributed as compressed archive file.

Download the license server archive 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 registration number from the server license file provided to you by Parametric Technology.

Create a directory for the pure::variants license server on your disk and extract the contents of the archive into that folder. For Linux systems a recommended location is the directory */opt/pure-variants*. On Windows a recommended location is the directory *C:\Program Files\pure-systems\pure-variants_License_Server_6.0*

Create a directory for the license files. A proper location for Linux systems is */opt/pure-variants/licenses* and on Windows *C:\Program Files\pure-systems\pure-variants_License_Server_6.0\licenses*. Copy the provided server license file into that folder.

Open script *start.bat* on Windows resp. *start.sh* on Linux in a text editor. The script is located in the *server* subdirectory. Update the following variables to match your environment:

HOSTNAME: The hostname or IP address of the server machine the license server should bind to.

PORT: The TCP port to bind to on the server machine.

LICENSEDIR: The folder containing the license files.

Clients must be able to connect to the given port on the server machine. Therefore this port must not be blocked by a firewall or used by another service. A port number can range from 0 to 65535. Port 80 is the standard port for the HTTP and 443 for the HTTPS protocol. Using these standard ports can help with firewall problems.

The server can be started by running script *start.bat* on Windows resp. *start.sh* on Linux. Please consult your system documentation how to add this script to the system's start sequence to ensure the server is started automatically after system reboot. You are now able to connect to the pure::variants License Server with your pure::variants Enterprise clients.

See Section 4.5, “Basic Setup with the pure::variants License Server Web Interface” for further installation steps.

On Windows the pure::variants server can be registered as a service. Selecting this option ensures that the pure::variants server always runs after reboot even when no user is logged in. To register the pure::variants server as a service, use the command line option /install. If installing the pure::variants license server as a service, a service name has to be given with command line option /servicename. A service description can be added with command line option /servicedesc. Here is an example how to register the license server as service:

```
<license server install path>/server/variantsd.exe /install
/servicename "pure--variants License Server" /servicedesc "The license server is providing
licenses for the pure::variants Desktop Client"
```

Enable pure::variants License Server Web Interface

The pure::variants license server includes an optional web interface for monitoring and interacting with the server.

To enable the server's web interface on Windows, add the following options to the server command line at the end of the script *start.bat*:

```
/enableweb
/webpwd PASSWORD
```

To enable the server's web interface on Linux, add the following options to the server command line at the end of the script *start.sh*:

```
--enableweb
--webpwd PASSWORD
```

The server's web interface should be always protected by a password. Please use a secure password for this.

If the License Server Web Interface should be accessible with the help of a context path, add the following options to the server command at the end of the *server/start.bat* script. For example if you want the License Server to be accessible with context */pvlicenseserver* please replace *BASEPATH* with *pvlicenseserver* in the command below:

```
/webbasepath BASEPATH
```

To define the base path of the License Server Web Interface in linux, add the following options to the server command at the end of the *server/start.sh* script:

```
--webbasepath BASEPATH
```

4.3. Update pure::variants License Server

4.3.1. Update with Windows Installer (Windows Only)

Updating an existing pure::variants license server works exactly the same as a clean installation of the pure::variants license server. Please see Section 4.2.1, “Install with Windows Installer (Windows only)”. Additional settings performed in the server configuration file will remain in the server configuration file after updating.

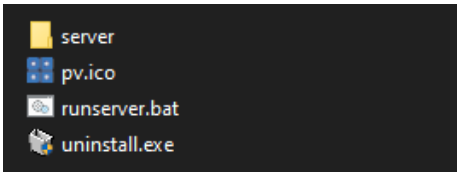
4.3.2. Update with Archive

Updating an existing pure::variants license server works exactly the same as a clean installation of the pure::variants license server. Please see Section 4.2.2, “Install from Archive”.

4.4. Uninstall pure::variants License Server

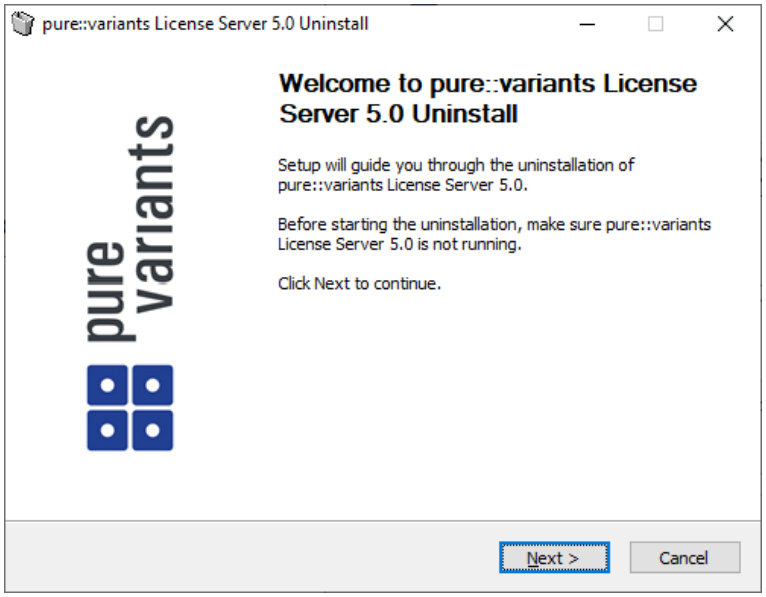
The uninstaller for the pure::variants license server can be started in two different ways. The first one is to go to the Windows *Add or remove programs* application and search for *pure::variants License Server 6.0* and start the uninstaller by using the *Uninstall* action. The uninstaller requires Administrator privileges.

Figure 45. pure::variants License Server Uninstaller



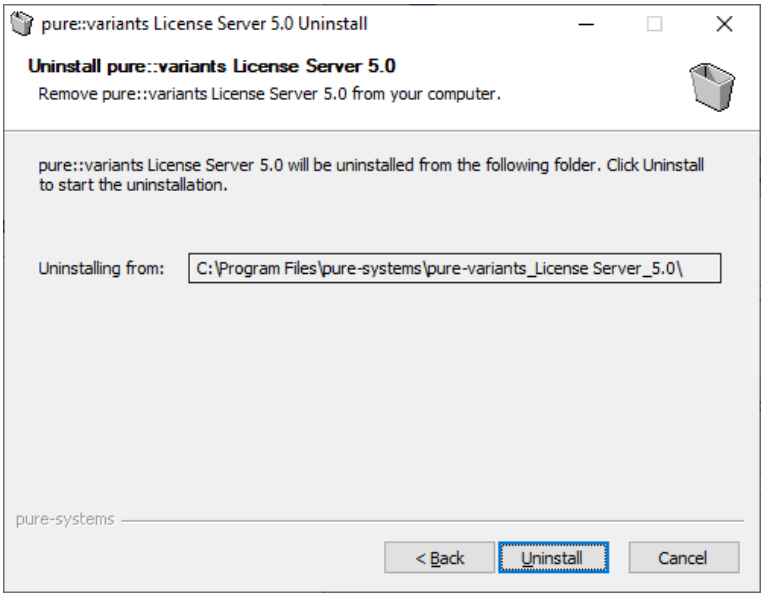
The second possibility is to navigate to the pure::variants license server installation folder and start the uninstaller by double-clicking it.

Figure 46. pure::variants License Server Uninstaller



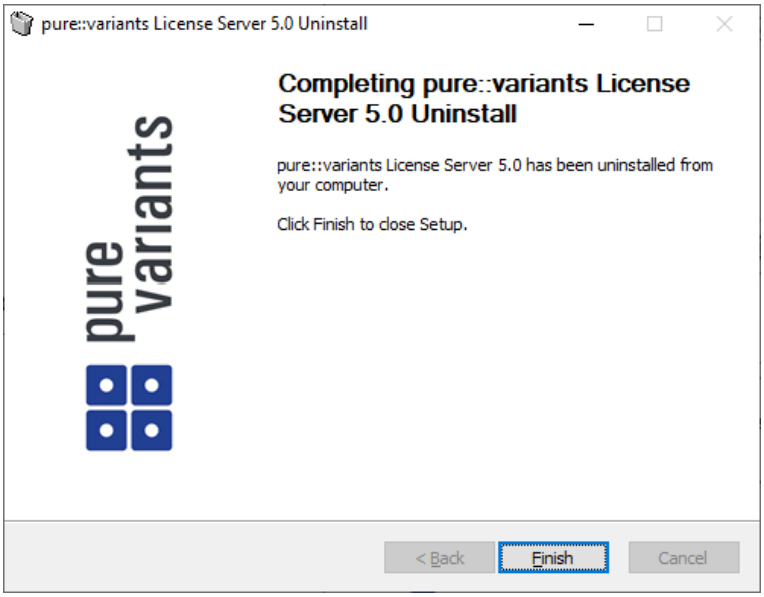
Click *Next*.

Figure 47. Uninstall from



Click *Uninstall* to start the uninstall process.

Figure 48. Completing Uninstall

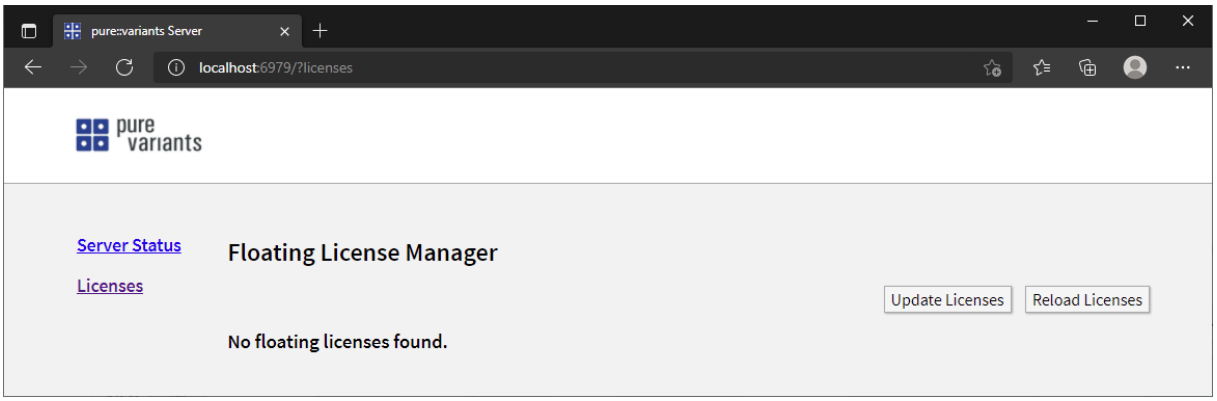


The deinstallation is succesfully finished. Click *Finish* to close the uninstaller.

4.5. Basic Setup with the pure::variants License Server Web Interface

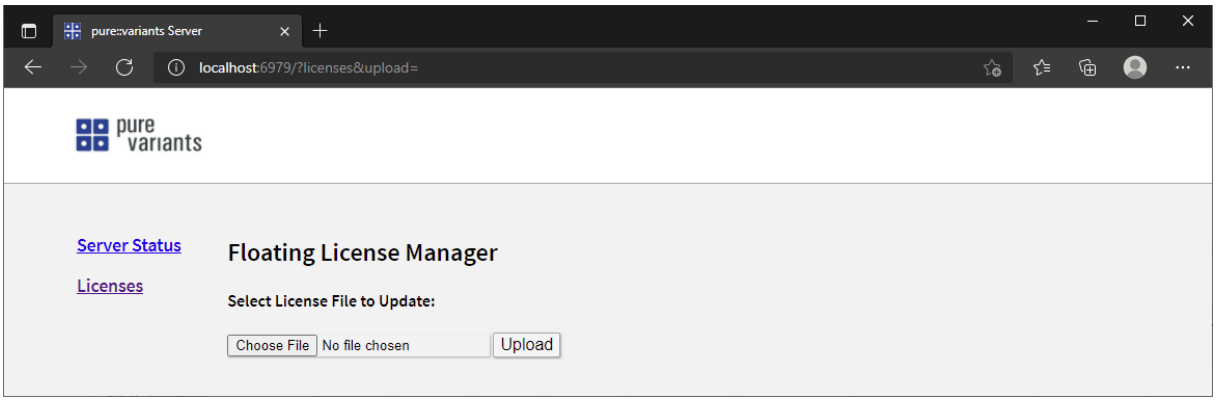
To add your floating licenses to the server, open the web interface. If a password was enabled during installation, the web interface will show a *Login* button. Click the button, enter the password, and leave the user field empty. Go to the *“License”* page.

Figure 49. License Server Web Interface



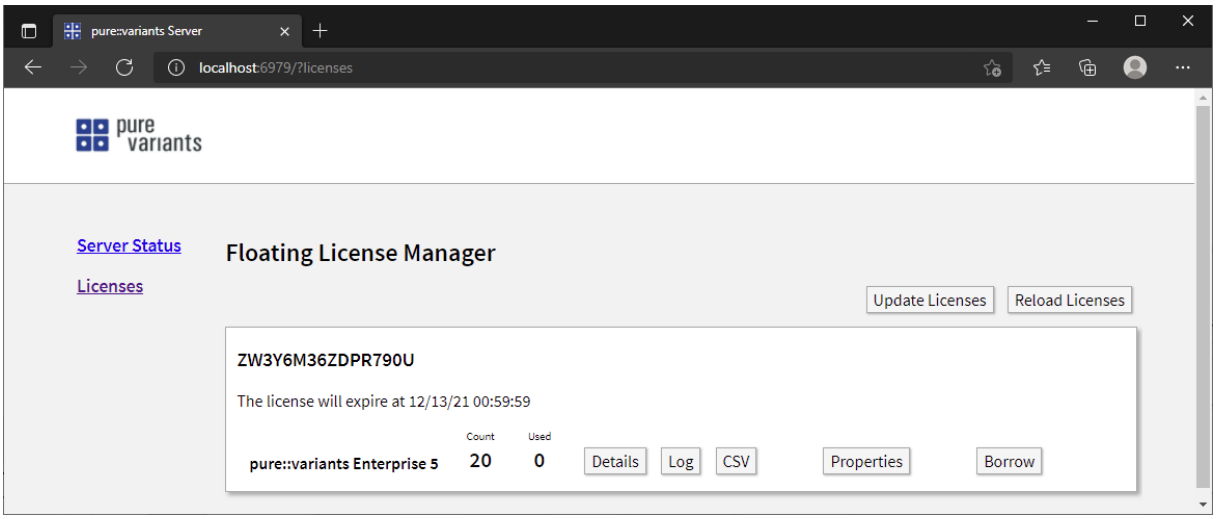
Because there is no license installed, the web page shows a *No floating licenses found* message. Click *Update Licenses*.

Figure 50. Upload License



Select the server license file from your hard disk. Click *Upload*.

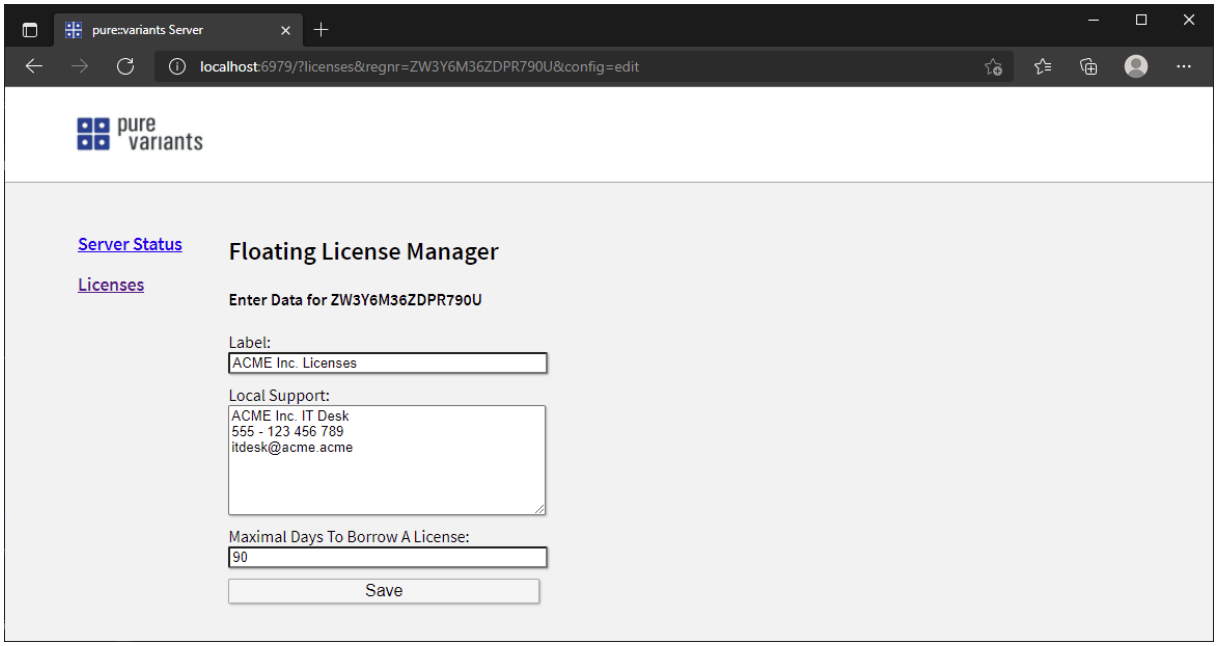
Figure 51. License Overview



After successful license update, the license entry is shown on the web page. The entry shows your registration number and the number of available and used licenses. The server is able to serve multiple licenses. The different licenses will be distinguished by the registration number.

For easier usage every license entry can get a label. Click on *Properties* to enter a label and local support information for the license.

Figure 52. License Server Detailed License Data

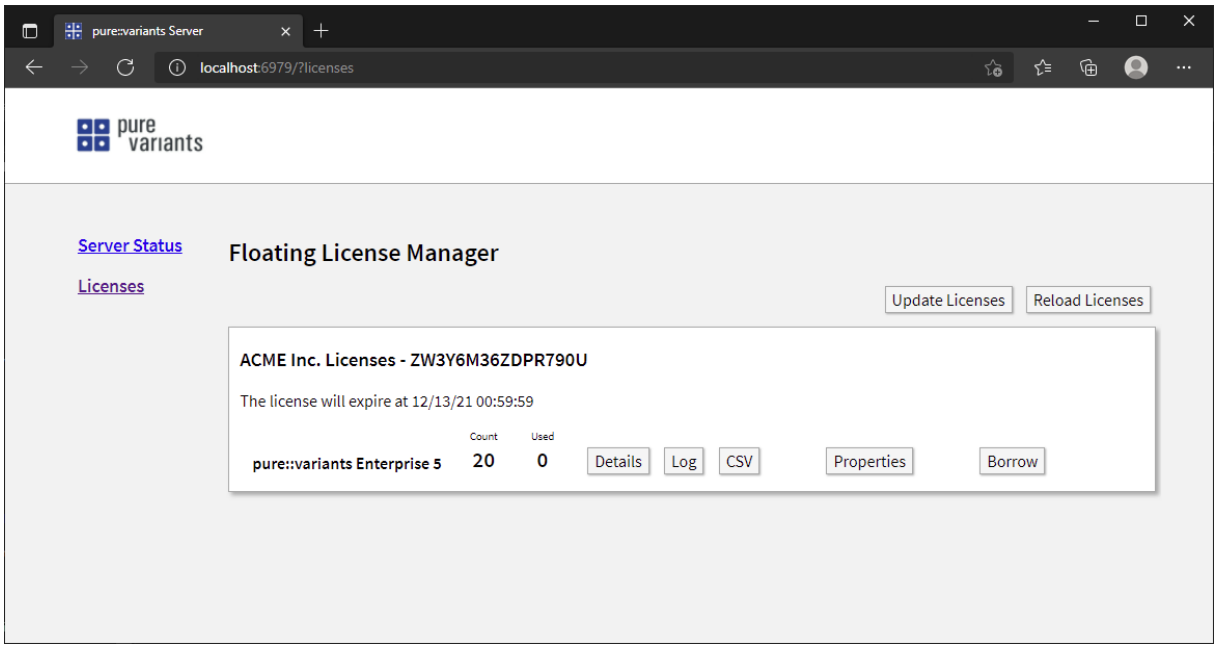


The label is shown on the web page in front of the registration number. The local support data is presented to the clients in case of license access errors. Enter your IT help desk contact data here to inform the user who can be called in case of any problems.

The *Maximal Days To Borrow A License* property defines the maximum amount of days a license can be borrowed by the client or in the web interface. If the value is set to 0, borrowing of licenses is disabled. pure::variants Desktop Clients can use a maximum amount of 90 days for borrowing a license. Longer time intervals can be used in the web interface only.

Click *Save* to store the data.

Figure 53. License Overview



Now the entered label is shown in the license entry.

4.6. License Server Command Line Options

For generic pure::variants server command line options see Section 7.9, “Server Command Line Options”.

Following list describes the license server specific options that can be added to the pure::variants license server command line or configuration file.

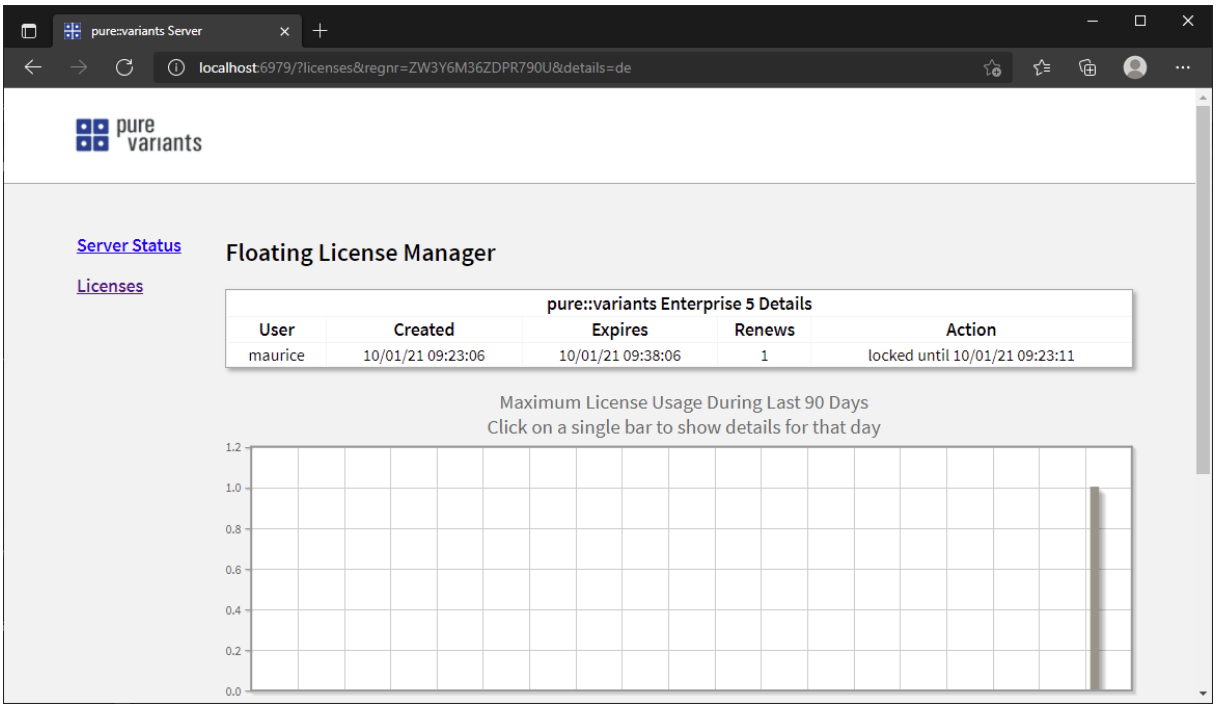
See Section 7.4, “Location of the Server Configuration File” to find the configuration file.

Option	Description
/license	Location of the directory containing the license files, or the location of a single license file. This parameter can be specified multiple times.
/licenselog	Enable license usage logging
/licenseuserlog	Enable license usage logging including user data
/licenseuserlist [PATH]	Location of a user license access control list.
/licenseminclient [VERSION]	Minimal version of a client to get a license - valid value is 5

4.7. Get information about the license usage

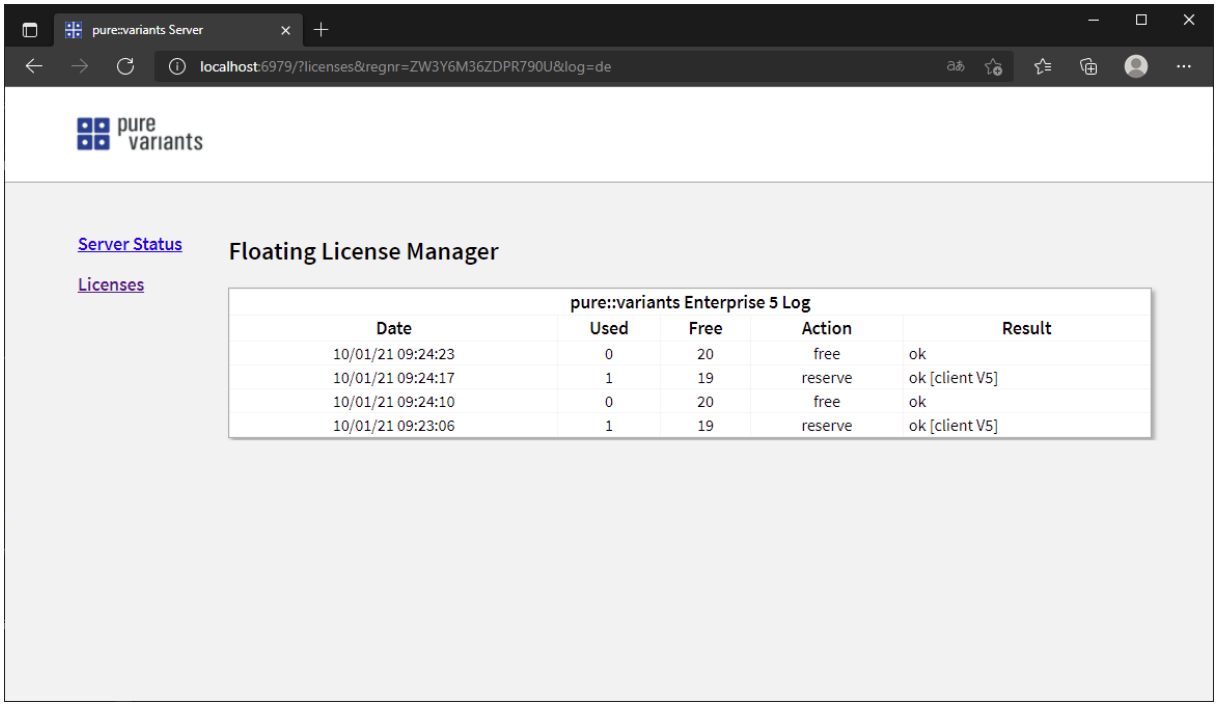
For every license entry you can view the current usage details and a chronological log by clicking the *Details* or *Log* button. After pressing the *CSV* button the license log of the last 90 days is exported into a CSV file.

Figure 54. License Details



Pressing *Details* opens the license details page. The upper part of this page shows who is currently using a license. The lower part shows the maximal license usage during the last 90 days.

Figure 55. License Log



The table on the log page shows all license reserve and free operations. Possible errors are also shown here.

4.8. Borrow a license in the web interface

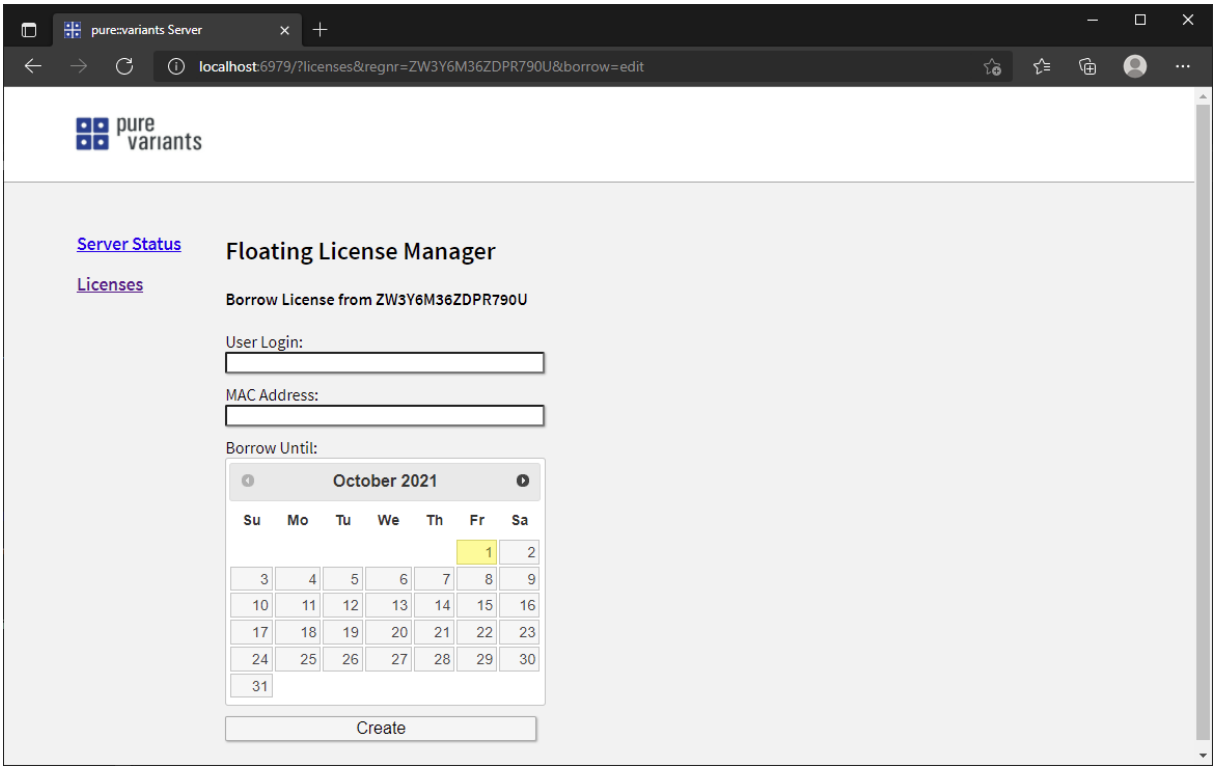
The web interface provides the possibility to reserve an offline license. This offline license can be used like a normal client license.

With the *Borrow* button on one of the license entries it is possible to create a client license for a specified amount of days. On the next page the user login name, the MAC address of the client computer, and the expiration date need to be specified.

Be careful, borrowing a license will block the license until the specified expiration date is reached. This can not be reverted. So check the defined login name and MAC address properly before creating the license.

After using the *Create* button, the client license is created and downloaded automatically.

Figure 56. Borrow a License



4.9. Adding the License Server Information to the Client License

The client license file may be optionally changed to contain the information where the pure::variants license server is located.

The *url* attribute of the *subtype* tag is initially empty and may contain the server URL constructed from the information given during the installation process.

Note

Always use the full domain name including the port number (even for standard ports 80 and 443).

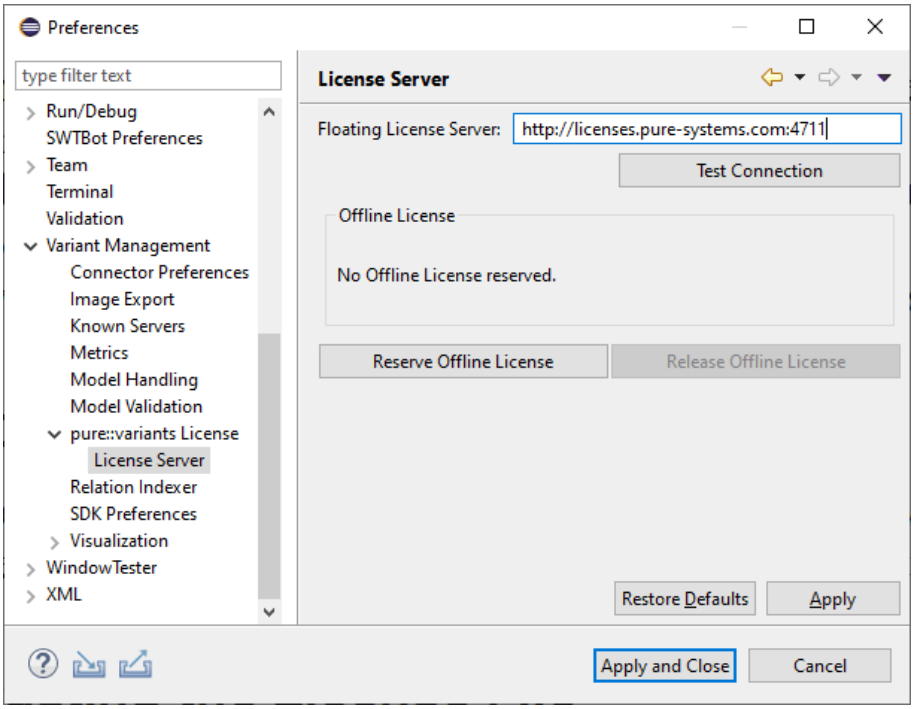
```
<?xml version="1.0" encoding="ISO-8859-1"?>
<licence version="1.1">
  <product>
    <name>pure::variants</name>
    <version>2</version>
    <function>de</function>
    <platform>win32</platform>
    <platform>linux</platform>
    <platform>macos</platform>
  </product>
  <creation>1-1-2000</creation>
  <type>float</type>
  <subtype url="http://pvlicense.example.com:80">licence server</subtype>
  <registration>
    <firma>ACME Inc.</firma>
    <name>Jane Doe</name>
    <email>jane.doe@acme.acme</email>
    <number>1234567890ABCD</number>
  </registration>

<signature>12..EF</signature></licence>
```

4.9.1. Setup License Server Location

This step is necessary only if a floating license with a pure::variants license server is used. The license server URL can be provided with the floating license file, or it has to be set by the user. This setting has to be performed once per workspace in the pure::variants Desktop Client.

To set the license server URL, open the **Preferences** (menu *Window -> Preferences*). Navigate to **Variant Management -> pure::variants License -> License Server** and enter the URL of the license server into the **Floating License Server** text field.

Figure 57. pure::variants License Server Preferences



4.10. Borrow a license in the Eclipse Client

The Desktop Client provides possibility to borrow a license. With such a license the pure::variants Desktop Client works without a connection to the license server for the time the license is borrowed.

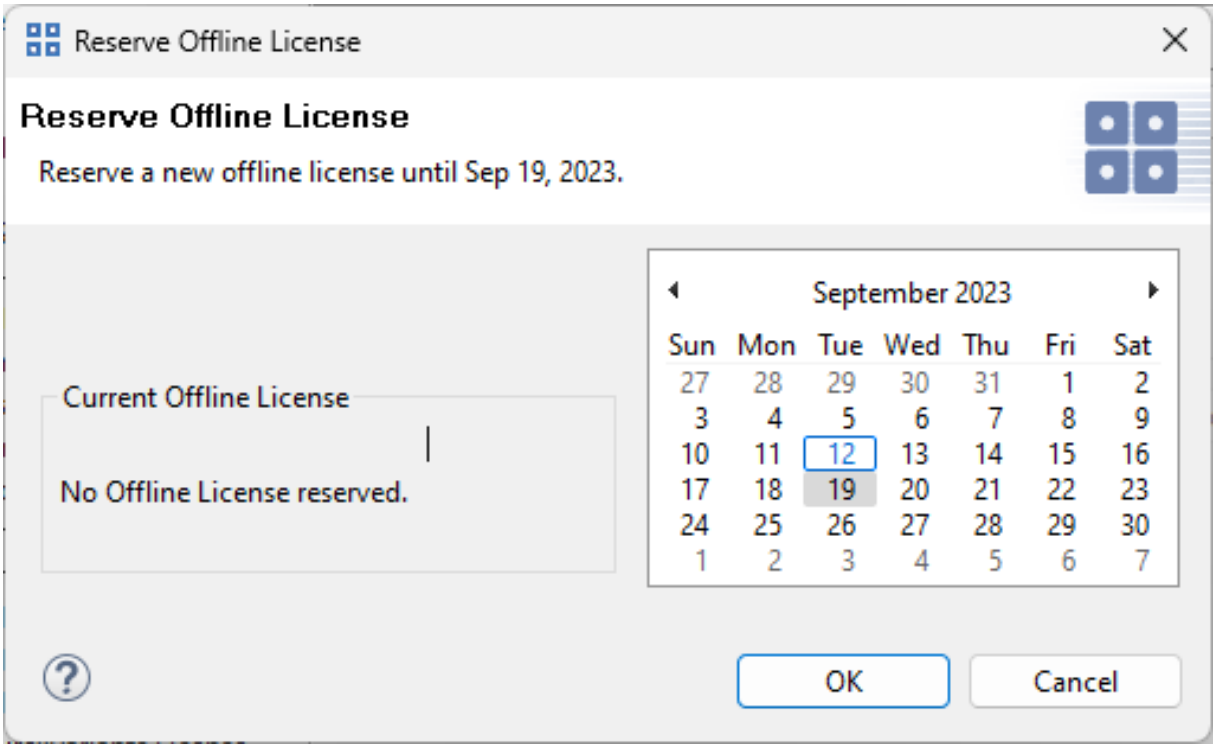
Be careful, borrowing a license will block the license until the specified expiration date is reached. This can not be reverted. So check the expiration date properly before borrowing the license.

To borrow the license click "Reserve Offline License" on the license server preference page (see Figure 57, “pure::variants License Server Preferences”). A dialog opens to specify the expiration date of the offline license.

Note

Clicking the OK button immediately locks the license on the license server and downloads it to the client, so make sure the expiration date is correctly set.

Figure 58. Reserve Offline License Dialog



4.11. Add a license access control list

With a license access control list it can be controlled which users are allowed to get a license from the pure::variants License Server, globally in Single License Mode as well as for specific license registration numbers in Multi License Mode.

Users can be allowed and denied to get a license separately. A license is granted to a user by the pure::variants License Server in Single License Mode if

- no license access control list is configured, or
- a global license access control list is configured, and
  - the user is explicitly allowed, or no user is explicitly allowed, and
  - the user is not explicitly denied

In Multi License Mode a license is granted to a user by the pure::variants License Server if

- no license access control list is configured for the license registration number for which the user requests a license, or
- a license access control list is configured for the license registration number for which the user requests a license, and
  - the user is explicitly allowed, or no user is explicitly allowed, and
  - the user is not explicitly denied

A license access control list is a text file with the following (XML) format.

```
<userlists ignorecase="true">

  <!-- global access control list -->
  <userlist>
```

```
    <allow>username1</allow>
    <allow>username2</allow>
    ...
    <deny>username3</deny>
    <deny>username4</deny>
    ...
  </userlist>

  <!-- for registration number XYZABC123 only -->
  <userlist regnr="XYZABC123">
    <allow>username1</allow>
    <allow>username2</allow>
    ...
    <deny>username3</deny>
    <deny>username4</deny>
    ...
  </userlist>

  ...
</userlists>
```

The listed user names are case sensitive. This can be changed by setting the optional attribute *ignorecase* to the value *true*.

The following example only allows “user1” to get a license for the license registration number “ABC”. No other user can get a license for this registration number.

```
<userlists ignorecase="false">
  <userlist regnr="ABC">
    <allow>user1</allow>
  </userlist>
</userlists>
```

The next example allows all users except of “user1” to get a license for the registration number “ABC”.

```
<userlists>
  <userlist regnr="ABC">
    <deny>user1</deny>
  </userlist>
</userlists>
```

And the last example allows the users “user1” and “user2” to get a license for the registration number “ABC”, and all users except of users “user3” and “user4” to get a license for the registration number “123”.

```
<userlists>
  <userlist regnr="ABC">
    <allow>user1</allow>
    <allow>user2</allow>
  </userlist>
  <userlist regnr="123">
    <deny>user3</deny>
    <deny>user4</deny>
  </userlist>
</userlists>
```

The license access control list file is considered by the pure::variants License Server by adding command line option *licenseuserlist* to the server configuration file or server command line. This option expects the path to the license access control file as argument. Example:

```
/licenseuserlist “C:\pvLicenseACL.xml”
```

See Section 7.4, “Location of the Server Configuration File” to find the configuration file.

Just add the option as new line to the configuration file and either restart the pure::variants License Server, or click the *Reload Licenses* button on the *Licenses* page of the pure::variants License Server Web Interface. This can also be automated by opening the Web interface page with the URL parameter “reload=” added.

Examples for the tools curl and wget:

- curl "http://server:port/?licenses&reload="
- wget -O- "http://server:port/?licenses&reload="

If a password for the web interface is defined, the requests have to provide this password as well. The user has to be specified since the tools are not working without. Just leave “admin” here. The “password” has to be replaced by your password.

- curl "http://server:port/?licenses&reload=" -uadmin:password
- wget -O- "http://server:port/?licenses&reload=" --user=admin --password=password

4.12. Proxy Configuration

The pure::variants license server can be placed behind a reverse proxy. No configuration changes are required on the side of the license server.

Since the license server only uses relative links, e.g. to load image, CSS, and JavaScript files for its web interface from the server, the reverse proxy needs to correct the paths in the HTTP response headers (redirect).

The following code shows a minimal example for an Nginx reverse proxy configuration suitable for the pure::variants license server.

```
events {
  worker_connections 1024;
}

http {
  server {
    listen 80;
    server_name proxy;

    location /pvlic/ {
      proxy_pass     http://backend:8081/;
      proxy_redirect http://backend:8081 /pvlic;
    }
  }
}
```

This configuration passes requests addressed to *http://proxy/pvlic/* to the backend server address *http://backend:8081/*. The server names *proxy* and *backend* only are examples and need to be replaced with your actual server names. The same applies to the ports and the location path in the example.

The following code shows the same example for Apache and IBM HTTP Server (just the proxy part).

```
ProxyPass        /pvlic/ http://backend:8081/
ProxyPassReverse /pvlic  http://backend:8081
```

5. pure::variants Deployment Templates for Docker

The pure::variants Deployment Templates are the recommended way to install and manage the pure::variants Web Components and the pure::variants Web Client. These templates reduce the complexity of installing the various components required for a successful deployment and maintenance of pure::variants. Currently 3 templates are provided, covering the 3 main deployment scenarios typically required.

All templates provide access to the provided pure::variants services through a integrated reverse proxy under a shared base URL. The URL scheme below this shared base URL is fixed. See description of the deployment templates for the provided service URLs.

The following selection guide should be used to select the most appropriate template:

A: All-In-One	This template provides the pure::variants Model Server with PostgreSQL database, the pure::variants Web Components including the pure::variants Web Client and the Transformation Service. User authentication handled by the pure::variants Model Server. Recommended Use Case: First time installations; tool evaluations; no single-sign-on service usage required. It is the least complex installation with a minimum of external dependencies
B: All-In-One with External Single-Sign-On via OpenID Connect	This template provides the pure::variants Model Server with PostgreSQL database, the pure::variants Web Components including the pure::variants Web Client and the Transformation Service. User authentication handled by an external OpenID Connect single-sign-on service. Recommended Use Case: Productive use. This is the recommended template for most deployment scenarios up to 100 active users
C: Web Components Only with External Model Server and Single-Sign-On	This template provides just the pure::variants Web Components including the pure::variants Web Client and the Transformation Service. The pure::variants Model Server is separately installed. User authentication handled by an external OpenID Connect single-sign-on service. Recommended Use Case: Productive use. This is the recommended template if legacy deployments shall be extended with the capability to provide the pure::variants Web Client or if the integrated PostgreSQL is not the desired storage for the pure::variants Model Server.

5.1. Deployment Architecture of the Deployment Templates

All templates share the same architecture. All external access to services is handled by the pure::variants gateway. The gateway acts as a reverse proxy routing the requests to the internal services aswell as an authentication service for the user and is also the single endpoint for TLS termination.

Internally a number of additional containers, orchestrated by Docker Compose are communicating among each other using an internal, preconfigured network structure, which is not exposed externally. Connections of internal services to external services not provided by the respective template, are done using encrypted access using TLS.

The persistent data storage of internal services is done via folder volumes (e.g. for the database service or the transformation service).

5.2. Shared Requirements For All Deployment Templates

A pure::variants Docker deployment requires at least

- For productive use: Linux running Docker version 20.10.0 or higher. For evaluation/test use cases, Windows with Docker version 20.10.0 or higher may be used.
- Docker Compose version 1.27.1 or higher, or Compose plugin for Docker
- the pure::variants Docker Setup package from the pure::variants updatesite
- a running pure::variants License Server (see Section 4.2, “Install pure::variants License Server”)

5.3. Template A: Deployment Instructions

Please follow these steps to deploy template A.

5.3.1. Prerequisites

1. Decide how many concurrent transformations shall be possible. Default in the templates is to have 1 Transformation Runner enabled. It is recommended to start with this and add more Runners later.

2. The URL for the pure::variants License Server. Verify that the designated docker host machine can access this URL.

3. The email address and registration number from your pure::variants license. This can be retrieved from the pure::variants License Servers status page.

4. A X.509 certificate and key for the hostname and port under which the pure::variant services are exposed from the docker host. The key file must not have password protection.

5. Define designated exposed hostname and port for the external access to the provided pure::variants services. This is the hostname users will later use to access the services. To reduce complexity of the setup, it is recommended to at least start with these being the same as in the previous prerequisite.

6. If offering the Transformation Service to Web Client users and the external tool service used during transformation have self-signed certificates or certificates of a non-public certification authority (CA): Get the respective self-signed certificate or certification authority certificates in PEM format with the `.crt` suffix.

7. If the pure::variants License Server is secured by a self-signed certificate or with a certificate of a non-public certification authority (CA) : Get the respective self-signed certificate or certification authority certificates in PEM format with the `.crt` suffix.

8. Have access to an pure::variants Enterprise Desktop Client. This is used to verify correct operation of the deployment. The machine on which the pure::variants Desktop Client is installed must be able to access the docker host and port (see prerequisite 5).

5.3.2. Prepare Installation Directory

First you need to download the pure::variants Docker Setup package 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 your pure::variants license.

Download the package "pure::variants Deployment Templates" and extract it to any directory on the Docker host which will become the installation directory of the pure::variants Docker Deployment.

5.3.3. Select Service Template

To configure the deployed pure::variants services, first copy the provided env.template_a file to the file **`.env`** in the same directory. This file is located in the root of the installation directory and contains all the configuration options for the Docker deployment.

5.3.4. Provide Server Certificate and Key

The pure::variants Docker Deployment always uses secure communication (HTTPS) with external services. This requires to provide a sever certificate for the exposed hostname of the Docker deployment.

Place the server certificate and corresponding private key for the Docker host here in the installation directory:

- **`workspace/certificates/server.crt`** : the server certificate in PEM format
- **`workspace/certificates/server.key`** : the server private key in PEM format

If you don't have a server certificate, you have to request one for your Docker host from your company IT first.

For testing and evaluation purposes **only** you may also use a self-signed certificate. Do **not** use self-signed certificates for production setups! To create a self-signed certificate, you can for instance run the following command line on a Linux terminal in the root of the installation directory.

```
tools/create_self_signed_certificate.sh
```

5.3.5. Setup Certificates for Secure External Service Access

If both prerequisite 6 and 7 were not matching your situation, skip this step.

Otherwise, place all to be trusted certificates / root ca chains as files in PEM format with suffix **`.crt`** in the installation directory:

- **`workspace/rootcertificates`**

Please note, that updating the certificates in this directory at a later point in time always requires a rebuild of the Docker images. See instructions below in Section 5.3.7, “Build Docker Images” when updating after initial build.

5.3.6. Configure Service Template A

To configure the deployed pure::variants services, first copy the provided **`env.template.a`** file in the root of the installation directory to the file **`.env`** in the same directory. This file contains all the configuration options for the Docker deployment.

Open the **`.env`** file in a text editor. Change the following configuration options. If there is a `#` sign in front of one of these options (i.e. this option is commented out), please remove the `#`.

- **`PV_VERSION`**

  Enter the pure::variants version label that you want to build and deploy. Technically this is used as label for the docker images which are build and deployed. It is recommended to use the pure::variants version number here, e.g.. `6.0.0`.

- **`PV_REG_NUMBER`**

  Enter your pure::variants license resp. registration number, which you for instance can find in your pure::variants license file. This number is used to access the floating licenses on the pure::variants License Server.

- **`PV_EXPOSED_HOST`**

  This option defines the base network address used to access the pure::variants services. Enter the full name of the Docker host, e.g. `pv.example.com`

- **`PV_EXPOSED_PORT`**

  Enter the host port on which the pure::variants services are bound. It is recommended to use the standard port `443`. Other ports may need to be allowed in your firewalls first.

- **`PV_TOKEN_SIGNING_KEY`**

  Enter a key used to sign internal JSON Web Tokens exchanged between pure::variants services. The key length must be at least `256` bit.

- **`PV_LICENSE_SERVER`**

  Enter the address of the pure::variants License Server, e.g. `https://pvlicenses.example.com`.

- **`PV_MODEL_SERVER_WEB_PASSWORD`**

  Enter the password with which you want to protect the status web page of the pure::variants Model Server.

- **`PV_MODEL_SERVER_APIKEY`**

  Enter the API key with which you want to protect the API endpoints of the pure::variants Model Server.

- **`PV_MODEL_SERVER_SYSTEMUSER_PASSWORD`**

Enter the password with which you want to protect the pure::variants Model Server superuser `system`.

- **`PV_MODEL_SERVER_DB_PASSWORD`**

  Enter the password with which you want to protect the local pure::variants Model Server PostgreSQL database.

- **`PV_TRANSFORM_RUNNER_1`**

  Replace the default API key `changeit` of the default transformation runner.

- **`TZ`**

  Replace the default time zone `Europe/Berlin` with your time zone. This information is used to let the Docker containers have the correct time according to your location. Please see https://en.wikipedia.org/wiki/List_of_tz_database_time_zones for the available time zones.

If you want to perform tool transformations (e.g. for Codebeamer or Polarion), you may need to specify the corresponding tool credentials as described in Section 5.6.1, “Configuring the Transformation Service Access to 3rd party tools without Single-Sign-On”.

5.3.7. Build Docker Images

The following Docker images are built by the pure::variants Docker evaluation deployment. The image labels depend on the **`PV_VERSION`** configuration option. 6.0.0 is used as an example in the following overview.

- **`pv-gateway:7.0.0`**

  Provides an internal proxy for the pure::variants services included in the deployment setup.

- **`pv-model-server:7.0.0`**

  Provides a pure::variants Model Server.

- **`pv-model-server-db:7.0.0`**

  Provides a local PostgreSQL database for a pure::variants Model Server included in the deployment setup.

- **`pv-web-client:7.0.0`**

  Provides a pure::variants Web Client.

- **`pv-transform-runner:7.0.0`**

  Provides a pure::variants Transformation Runner.

- **`pv-transform-executor:7.0.0`**

  Provides the transformation executor used by a pure::variants Transformation Runner to perform the actual transformations.

To build all Docker images, run the following command line in the root of the installation directory.

```
docker compose --profile build build
```

Please see the instructions in Section 5.6.5, “Custom Docker Registry” if you need to use a custom Docker registry.

5.3.8. Start and Stop Services

To start all pure::variants services of a Docker deployment, run the following command line in the root of the installation directory.

```
docker compose up -d
```

You can list the running services by using the following command line.

```
docker compose ps
```

The log messages produced by the running services can be viewed using the following command line.

```
docker compose logs
```

To stop all running services use the following command line.

```
docker compose down
```

To start, stop, or view the log messages of single services, you can add the service name to the command line. If you for instance want to view the logs of the pure::variants Model Server, you can use this command line:

```
docker compose logs modelserver
```

The pure::variants services are reachable at the following addresses, depending on the host (option **`PV_EXPOSED_HOST`**) and port (option **`PV_EXPOSED_PORT`**) you have configured for the Docker deployment. This overview uses the fictional hostname `pv.example.com` and port `443`.

- **`https://pv.example.com:443/pv/`**

  Address of the pure::variants Model Server.

- **`https://pv.example.com:443/pvweb/`**

  Address of the pure::variants Web Client.

5.3.9. Validate Correct Operation of Deployment

1. Validate Model Server Availability: Use browser to navigate to Model Server URL (make sure to have the slash at the end of the URL), e.g., to https://pv.example.com:443/pv/ . You should now see the basic status page of the Model Server. Try to log in with the password defined in `PV_MODEL_SERVER_WEB_PASSWORD`.

2. Validate Web Client Availability: Use browser to navigate to Web Client URL (make sure to have the slash at the end of the URL), e.g. to https://pv.example.com:443/pvweb/ . You should now see the Login page of the Web Client. Try to log in with the user name `system` and the password defined in `PV_MODEL_SERVER_SYSTEMUSER_PASSWORD`. Validate Number of available Transformation Runners by switching in the Web Client to the Transformation Dashboard and check that the Idle count is equal to the number of configured Transformation Runners (Default is 1).

3. Validate Model Server Access via Desktop Client: Follow the instructions given in the *pure::variants Model Server Administration Manual* using the Model Server URL (make sure to have the slash at the end of the URL), e.g., https://pv.example.com:443/pv/ .

5.3.10. Update Deployment

Configuration Change

If you just want to change the configuration of your pure::variants Docker Deployment, edit file **`.env`** in the root of the Docker deployment installation directory and make the changes.

If you changed the pure::variants version (option **`PV_VERSION`**) you will most probably need to rebuild the Docker images by running the following command line in the root of the installation directory.

```
docker compose --profile build build
```

To finally apply the changes, please run the following command lines.

```
docker compose down
docker compose up -d
```

Full Update

A full update of the pure::variants Docker deployment can be achieved as follows.

First make a copy of file `.env` and the subdirectory `workspace` in your pure::variants Docker deployment installation directory.

Then download a fresh pure::variants Docker Setup package from the pure::variants updatesite. See Section 5.3.2, “Prepare Installation Directory” on how to do this.

Stop the running pure::variants services by executing the following command line in the Docker deployment installation directory.

```
docker compose down
```

Extract the contents of the downloaded ZIP archive into the Docker deployment installation directory, then replace `.env` and `workspace` with the copies you made in the first step.

Rebuild the Docker images by running following command line.

```
docker compose --profile build build
```

Finally restart all services using the following command line.

```
docker compose up -d
```

5.4. Template B: Deployment Instructions

This template is similar to the previous template, but with the addition of using Single Sign-On via OpenID connect for the pure::variants Web Client and pure::variants Model Server access.

The steps are deployment as described in Section 5.3, “Template A: Deployment Instructions”.

Then configure Single Sign-On via OpenID Connect for the pure::variants Web Client.

5.4.1. Additional Prerequisites

1. Test that the URL of the OpenID Connect service can be accessed from the docker host machine.

   If the OpenID Connector service is secured by a self-signed certificate or with a certificate of a non-public certification authority (CA); or if a client certifcate is required by a proxy (see above) : The respective self-signed certificate or certification authority certificates in PEM format with the `.crt` suffix.

2. The **Web Client** registration in the Single Sign-On provider must be conform to the following specification and the hostname and port must be replaced according to the configuration in the template.

   - Grant types: authorization code and refresh token
   - Redirect URL: https://pv.example.com/pvgw/auth/oidc/callback
   - Logout URL: https://pv.example.com/pvgw/ui/
   - Scopes: The default scopes are **openid profile email**.

     Note: When using the webclient in combination with Jazz platform the additional scope **general** is needed.

- For global configuration management introspection must be enabled and the introspecting relying parties must be added with their **Client ID** and the needed permissions to the user management of the pure::variants Model Server. The Client ID can be found by navigating to https://<your-jazz-oidc-domain>/oidc/endpoint/jazzop/clientManagement. Once logged in to your Jazz Authentication Server, you will find an entry with the label /gc which shows the Client ID in the second column. Please see an example of the Global Configuration column in Jazz Authorization Server Client Management.

3. The **Model Server** registration in the Single-Sign-On provider must be conform to the following specification.

   - Grant types: authorization code
   - Redirect URL: https://localhost/pv/openid
   - Scopes: openid

   Further instructions for the Single Sign-On Setup of the Model Server are described in Section 7.6.1, “Open ID Connect Authentication”

5.4.2. Prepare Installation Directory

Follow instructions in Section 5.3.2, “Prepare Installation Directory”.

5.4.3. Select Service Template

To configure the deployed pure::variants services, first copy the provided **`env.template_b`** file in the root of the installation directory to the file **`.env`** in the same directory. This file contains all the configuration options for the Docker deployment.

5.4.4. Execute Steps from Template A

The steps Section 5.3.4, “Provide Server Certificate and Key” , Section 5.3.5, “Setup Certificates for Secure External Service Access” have to executed including the optionally required additional certificates for the OpenID Connectoer service, see prerequisites.

5.4.5. Configure Template B

To configure SSO for the pure::variants Web Client, please open the file **`.env`** in a text editor. This file is located in the root of the Docker deployment installation directory.

All configuration parameters as described in Section 5.3.6, “Configure Service Template A” have to configured according to the instructions before continuing the with parameters given below

Change the following configuration options. If there is a # sign in front of one of these options (i.e. this option is commented out), please remove the #.

- **`PV_WEB_CLIENT_SSO_MODEL_SERVER_USER`**

  Enter the name of the special pure::variants Model Server alias user that will be automatically created and used internally for SSO between the Model Server and the Web Client. The used name shall be long and not overlapping with the username scheme of the OpenID system. It is recommended to use a 64 characters long string of random ASCII letters and numbers as user name.

  `PV_WEB_CLIENT_SSO_MODEL_SERVER_PASSWORD`

  Provide the password of the pure::variants Model Server alias user specified with PV_CLIENT_SSO_SERVER_USER. It must be a strong password, to prevent issues with special characters

it is recommended to use a 64 character long random string of ASCII letters and numbers. This password is automatically assigned to the user when the user is created at first startup of the Deployment.

It can be later changed using the pure::variants Desktop Client. When changed, the change must be reflected in this variable and the affected services will restart after issuing **docker compose up -d**.

- **PV_GATEWAY_SSO_URL**

  Enter the well-known endpoint of the OpenID Connect provider without the trailing `/.well-known/openid-configuration`. Example: `https://jazz.example.com:9643/oidc/endpoint/jazzop`

- **PV_GATEWAY_SSO_CLIENT_ID**

  Enter the OpenID Connect client identifier for the pure::variants Web Client as defined at the OpenID Connect provider.

- **PV_GATEWAY_SSO_CLIENT_SECRET**

  Enter the OpenID Connect client secret for the pure::variants Web Client as defined at the OpenID Connect provider.

- **PV_WEB_CLIENT_SSO_GC_URI**

  Leave this option empty if you don't use Global Configurations. Otherwise enter the address of the Global Configuration service, e.g. `https://jazz.example.com:9443/gc`.

- **PV_MODEL_SERVER_OPENID_CLIENT_ID**

  Client identifier issued to the pure::variants Model Server by the OpenID Connect provider.

- **PV_MODEL_SERVER_OPENID_CLIENT_SECRET**

  Client secret assigned to the pure::variants Model Server by the OpenID Connect provider.

There are two more SSO related options that should be left unchanged if the token exchange with the OpenID Connect provider works fine.

- **PV_GATEWAY_SSO_SCOPES**

  Change this option only if your OpenID Connect provider uses a different set of token scopes. The default scopes requested are: openid, profile, email, and general).

5.4.6. Execute Steps from Template A

Follow all remaining steps from template A from and including Section 5.3.7, “Build Docker Images”

5.4.7. Validate Correct Operation of Deployment

1. Validate Model Server Availability: Use browser to navigate to Model Server URL (make sure to have the slash at the end of the URL), e.g., to https://pv.example.com:443/pv/ . You should now see the basic status page of the Model Server. Try to log in with the password defined in `PV_MODEL_SERVER_WEB_PASSWORD`.

2. Validate Model Server Access via Desktop Client: Follow the instructions given in the *pure::variants Model Server Administration Manual* using the Model Server URL (make sure to have the slash at the end of the URL), e.g., https://pv.example.com:443/pv/ . Create at one pure::variants user with a name matching a valid Single-Sign-On user as described in the manual.

3. Validate Web Client Availability: Use browser to navigate to Web Client URL (make sure to have the slash at the end of the URL), e.g. to https://pv.example.com:443/pvweb/ . You should now see the Login page of the Single-Sign-On service. Try to log in with valid user name and credentials from the previous step. Validate

Number of available Transformation Runners by switching in the Web Client to the Transformation Dashboard and check that the Idle count is equal to the number of configured Transformation Runners (Default is 1).

5.5. Template C: Deployment Instructions

The difference of template C compared to template B is that an extenal pure::variants Model Server is required.

To deploy template C, prepare deployment as described in Section 5.4, “Template B: Deployment Instructions”

5.5.1. Additional Prerequisites

1. Test that the URL of the external pure::variants Model Server can be accessed from the Docker host machine.

2. If the external pure::variants Model Server service is secured by a self-signed certificate or with a certificate of a non-public certification authority (CA); or if a client certifcate is required by a proxy (see above) : The respective self-signed certificate or certification authority certificates in PEM format.

5.5.2. Prepare Installation Directory

Follow instructions in Section 5.3.2, “Prepare Installation Directory”.

5.5.3. Select Service Template

To configure the deployed pure::variants services, first copy the provided **`env.template_c`** file to the file **`.env`** in the same directory. This file is located in the root of the installation directory and contains all the configuration options for the Docker deployment.

5.5.4. Configure Template C

To configure SSO for the pure::variants Web Client, please open the file **`.env`** in a text editor. This file is located in the root of the Docker deployment installation directory.

All configuration parameters as described in Section 5.3.6, “Configure Service Template A” and Section 5.4.5, “Configure Template B” have to be configured according to the instructions before continuing the with parameter given below:

- **`PV_MODEL_SERVER`**

  Enter the address of the pure::variants Model Server to use, e.g. `https://pv.example.com/pv/`.

Please note, that options **`PV_MODEL_SERVER_OPENID_URL`** , **`PV_MODEL_SERVER_OPENID_CLIENT_ID`** , and **`PV_MODEL_SERVER_OPENID_CLIENT_SECRET`** are not relevant in Template C and therefor not part of the template.

5.5.5. Validate Correct Operation of Deployment

1. Validate Web Client Availability: Use browser to navigate to Web Client URL (make sure to have the slash at the end of the URL), e.g. to https://pv.example.com:443/pvweb/ . You should now see the Login page of the Single-Sign-On service. Try to log in with valid user name defined in the pure::variants Model Server and their credentials. Validate Number of available Transformation Runners by switching in the Web Client to the Transformation Dashboard and check that the Idle count is equal to the number of configured Transformation Runners (Default is 1).

5.6. Additional Deployment Configuration Options

5.6.1. Configuring the Transformation Service Access to 3rd party tools without Single-Sign-On

If the pure::variants Web Client users shall be able to to perform pure::variants transformations for supported 3rd party tools (e.g. for PTC Codebeamer, Siemens Polarion or IBM Engineering Requirements Management -

DOORS Next), and these tools OR the pure::variants Deployment is not using Single-Sign-On, credentials for accessing these tools can be provided.

To specific required tool credentials, please open the file **.env** in a text editor. This file is located in the root of the Docker deployment installation directory.

Select and change the only the configuration options for the tools for which non-SSO access is required. If there is a # sign in front of one of these options (i.e. this option is commented out), please remove the #.

- **PV_CODEBEAMER_USER**

  Username for PTC Codebeamer.

- **PV_CODEBEAMER_PASSWORD**

  Password for PTC Codebeamer.

- **PV_DOORSNG_USER**

  Username for IBM Engineering Requirements Management - DOORS Next.

- **PV_DOORSNG_PASSWORD**

  Password for IBM Engineering Requirements Management - DOORS Next.

- **PV_POLARION_USER**

  Username for Siemens Polarion.

- **PV_POLARION_PASSWORD**

  Password for Siemens Polarion.

5.6.2. Multiple Transformation Runners

Per default there is exactly one pure::variants Transformation Runner in a pure::variants Docker deployment. Any number of runners can be added. Keep in mind the resource consumption of a transformation runner is roughly identical to that of a pure::variants Desktop Client. Adding to many runners can degrade performance of the system. Additional, like a normal active user, each runner consumes a floating license from the connected pure::variants License Server while a transformation job is being executed.

To add a second Transformation Runner, edit the configuration file **.env** located in the root of the Docker deployment installation directory.

Find option **PV_TRANSFORM_RUNNER_1** and duplicate it as **PV_TRANSFORM_RUNNER_2** . Change the runner ID to `runner-2` and provide a different API key. Example:

```
PV_TRANSFORM_RUNNER_2='{"id": "runner-2", "apikey": "my secret api key for runner 2"}'
```

Then find option **PV_WEB_CLIENT_TRANSFORM_RUNNERS** and add the second transformation runner like this.

```
PV_WEB_CLIENT_TRANSFORM_RUNNERS="[ $PV_TRANSFORM_RUNNER_1, $PV_TRANSFORM_RUNNER_2 ]"
```

Now edit file **compose.yaml** in the root of the Docker deployment installation directory and add the following at the end of the file. Ensure to use exactly the same indentation as the default Transformation Runner service `transformrunner-1`.

```
  # Example for a second transformation runner
  transformrunner-2:
    <<: *transformrunner
    profiles: ["transformrunner"]
```

```
    environment:
      - PV_RUNNER_CREDENTIALS=${PV_TRANSFORM_RUNNER_2}
```

Restart the pure::variants services to deploy the second runner as follows.

```
docker compose down
docker compose up -d
```

5.6.3. Advanced Logging

The deployed pure::variants services produce minimal logs per default. This can be changed by modifying the configuration as follows. Edit the configuration file **`.env`** located in the root of the Docker deployment installation directory, and adjust the following configuration options.

- **`PV_MODEL_SERVER_LOGLEVEL`**

  Log level of the pure::variants Model Server, from 0 (just errors) to 9 (extensive logging). Defaults to level 1 for minimal logging.

- **`PV_TRANSFORM_LOGLEVEL`**

  Log level of the pure::variants Transformation Runner, from 0 (just errors) to 7 (extensive logging). Defaults to level 1 for minimal logging.

You need to restart the pure::variants services to use the new log level. Execute the following command line in the root of the Docker deployment installation directory.

```
docker compose restart
```

The log messages of the pure::variants services can be viewed using the following command line.

```
docker compose logs
```

If you add `-f` to that command line, then you can watch the logs live.

Additional log messages are collected in log files located in the `pv_logs` Docker volume of the pure::variants Docker deployment (on Linux hosts usually found here: `/var/lib/docker/volumes/pv_logs/_data`).

5.6.4. Resource Limitation

The RAM available to the pure::variants Web Client can be configured using configuration option **`PV_WEB_CLIENT_RAM_LIMIT`** . It defaults to 16384 MB (16 GB) can be adjusted according to the following estimation formula:

```
Number of active user session * average number open models per user * Average number of model
 elements in model * 0.1
```

The result is the amount of RAM in megabytes recommended to be configured for the pure::variants Web Client. If for instance there are 10 models open, each with 1000 elements, and 10 users working with these models in the Web Client, then the recommended amount of RAM is 10000 MB (10 GB). The default setting can serve 16 or more users with a normal usage pattern.

To change the available RAM of the pure::variants Web Client of a Docker deployment, edit the configuration file **`.env`** located in the root of the Docker deployment installation directory.

- **`PV_WEB_CLIENT_RAM_LIMIT`**

  Enter the available RAM for the Web Client in megabytes, e.g. `4096` for 4 GB.

You need to recreate the Docker container of the pure::variants Web Client to apply this change.

```
docker compose up -d --force-recreate webclient
```

5.6.5. Custom Docker Registry

Using a custom Docker registry, other than the official Docker registry https://hub.docker.com/, makes it possible to deploy a pure::variants Docker Setup on a host without Internet access.

To change the Docker registry used by a pure::variants Docker Deployment, edit the configuration file **`.env`** located in the root of your Docker deployment installation directory and change the following option.

- **`DOCKER_REGISTRY`**

  Enter the address of the Docker registry to which you want to push the container images. This address must have a trailing slash, e.g. `https://registry.example.com/`

- **`DOCKER_BASE_REGISTRY`**

  Enter the address of the Docker registry from which you want to pull the base images used to build the pure::variants services container images. This address must have a trailing slash, e.g. `https://registry.example.com/`

This Docker registry is used to download the base images for the pure::variants Docker images during the image build step (see Section 5.3.7, “Build Docker Images”) as well for selecting the images to create the pure::variants services Docker containers during service startup.

5.6.6. Multiple Docker Deployments

If you plan to run multiple pure::variants Docker Deployments on the same Docker host, then each Docker deployment must use a separate prefix for the Docker resources (volumes, networks, etc). This is not an recommended use case for hosts with a production use deployment.

The default prefix for a pure::variants Docker deployment is `pv` and is defined in the **`COMPOSE_PROJECT`** configuration option. To change the prefix of a Docker deployment, edit the configuration file **`.env`** located in the root of this Docker deployment's installation directory.

- **`COMPOSE_PROJECT`**

  Enter a different prefix for this Docker deployment, e.g. `pv2`.

Recreate the Docker resources with the changed prefix by running the following command lines in the root of the Docker deployment's installation directory.

```
docker compose down
docker compose up -d
```

5.6.7. Modification of context paths

To distinguish traffic between the Model Server and the Web Client the deployment templates define context paths for both services. To reduce the amount of configuration parameters these context paths have a default value.

The Web Client's default path is set to `pvweb`

The Model Server's default path is set to `pv`

If your current setup demands or if you simply want to access the pure::variants services by different context paths the deployment templates provide a possibility to configure those.

To change the context path of the Web Client please add the parameter `WEBCLIENT_PATH` to your configuration file `.env` and set it to the desired context path.

```
WEBCLIENT_PATH="webclient"
```

To change the context path of the Model Server please add the parameter `MODELSERVER_PATH` to your configuration file `.env` and set it to the desired context path.

```
MODELSERVER_PATH="modelserver"
```

When modifying a context path please ensure that the context paths of the Web Client and the Model Server are set to **different** values. Furthermore, the deployment templates do not support nested context paths (e.g. `path/to/webclient`).

5.6.8. Modification of default Java Encoding

If your company will use both pure::variants Desktop Client and pure::variants Web Client please make sure to set the encoding of both clients to the same. The default encoding in the pure::variants Web Client is set to `UTF-8` which is the preferred encoding to correctly display all Unicode characters.

To change the default encoding please comment out the parameter `PV_JAVA_DEFAULT_ENCODING` and set it to the encoding matching the pure::variants Desktop Client.

```
PV_JAVA_DEFAULT_ENCODING="UTF-8"
```

5.6.9. Hide inaccessible Projects

As per default the pure::variants Web Client will show all projects existing on the Model Server to the user. The projects the user has permissions to access are fully shown and selectable where as the projects without proper permissions are greyed out. If you want to hide these projects completely, please modify the following parameter in your `.env` file to `true`.

```
PV_WEB_CLIENT_HIDE_INACCESSIBLE_PROJECTS="false"
```

6. pure::variants Deployment for Kubernetes

The pure::variants Deployment for Kubernetes is the recommended way to install and manage the pure::variants Services in Kubernetes. This deployment reduces the complexity of installing the various components required for a successful deployment and maintenance of pure::variants. The Kubernetes deployment is based on the Helm Package Manager and includes the same services and possibilities like the pure::variants Deployment Templates for Docker. The Helm Chart can also be used for a deployment into OpenShift. The pure::variants Helm Chart for Kubernetes is included in the pure::variants Deployment Templates for Docker with the name `pv-chart-<version>.tgz`.

6.1. Deployment Architecture for Kubernetes

The Kubernetes deployment includes the pure::variants Web Client, Transform Service and Model Server. All pure::variants related services are distributed as a helm chart which can be installed with the helm package manager. In contrast to the pure::variants Deployment Templates for Docker there are no prepared template files but a configuration file for the pure::variants helm chart which is called `values.yaml`.

Internally a number of additional containers, orchestrated by Kubernetes are communicating among each other using an internal, preconfigured network structure, which is not exposed externally via ingress routes. Connections of internal services to external services not provided by the respective template, are done using encrypted access using TLS. All externally available services as per default are encrypted using TLS.

6.2. Requirements for the Kubernetes Deployment

In general all prerequisites mentioned for the pure::variants Deployment Templates for Docker are still valid

6.2.1. General Requirements

- The deployment is verified and tested with Kubernetes v1.29.4 and OpenShift 4.13.41

- Helm Package Manager with version 3.14.1 or later
- the pure::variants Docker Setup and the pure::variants Deployment for Kubernetes package from the pure::variants Updatesite
- a running pure::variants License Server and its URL which is accessible to containers running in the Kubernetes Cluster (see Section 4.2, “Install pure::variants License Server”)
- Decide how many concurrent transformations shall be possible. Default in the parameters file is to have 1 Transformation Runner enabled. It is recommended to start with this and add more Runners later if needed.
- The email address and registration number from your pure::variants license. This can be retrieved from the pure::variants License Servers status page.
- A X.509 certificate and key for the hostname and port under which the pure::variant services are exposed from the docker host. The key file must not have password protection. Alternatively you can directly provide a TLS secret in the desired namespace.
- All certificates of a non-public certification authority (CA) in PEM format with `.crt` suffix which will be presented to the pure::variants services (e.g.: License Server, Model Server, Openid Connect Provider, 3rd party tools...)
- Define designated exposed hostname for the pure::variants services. The exposed port is set to 443. With this hostname users will later be able to access the pure::variants services.

6.2.2. Requirements for Single-Sign-On

- The **Web Client** registration in the Single Sign-On provider must be conform to the following specification.
  - Grant types: authorization code and refresh token
  - Redirect URL: https://pv.example.com/pvgw/auth/oidc/callback
  - Logout URL: https://pv.example.com/pvgw/ui/
  - Scopes: The default scopes are **openid profile email**.

    Note: When using the Web Client in combination with Jazz platform the additional scope **general** is needed.
  - For global configuration management introspection must be enabled and the introspecting relying parties must be added with their **Client ID** and the needed permissions to the user management of the pure::variants Model Server.
- The **Model Server** registration in the Single-Sign-On provider must be conform to the following specification.
  - Grant types: authorization code
  - Redirect URL: https://localhost/pv/openid
  - Scopes: openid

  Further instructions for the Single Sign-On Setup of the Model Server are described in Section 7.6.1, “Open ID Connect Authentication”

6.3. Building the Images

The images referenced in the pure::variants Kubernetes Deployment need to be built first. For building the images the pure::variants Deployment Templates for Docker are used. The detailed build steps are mentioned in Section 5.3.7, “Build Docker Images”.

Please provide at least the following prerequisites before running the build process:

- Set a version tag in the .env file for the images with the **PV_VERSION** variable.
- Define the target container registry in the .env file with the variable **DOCKER_REGISTRY**.
- Provide all needed root certificates within the folder `<docker-deployment-templates>/workspace/rootcer\tificates`.
- Add the parameter `PV_USERID=1000` anywhere in your .env file provided in the pure::variants Deployment Templates for Docker. This will enable the creation of an user in the transformation runner image.

Once the build is done please run `docker compose --profile build push` to push the images to your container registry.

6.4. Configuring the Kubernetes Deployment

Opening the extracted pure::variants Helm Chart the contained **`values.yaml`** file will contain all parameters to configure the pure::variants Deployment.

To keep the complexity manageable not all parameters available are directly visible in the parameters file but defaulted to a meaningful value which matches common customer needs.

The parameters file is divided into sections for each pure::variants service and one global section which applies to all services. These sections are represented in the following subchapters.

6.4.1. global

The global parameters are applied to all services installed with the pure::variants Deployment for Kubernetes.

- **PV_VERSION**

  Enter the pure::variants version label that you want to deploy. Technically this is used as tag for the images which are stored in your container registry and deployed to the Kubernetes cluster.

- **PV_NAMESPACE**

  Enter your name of your namespace to which you want to deploy the pure::variants services to.

  Note: This namespace needs to be created manually by running `kubectl create ns NameOfNamespace`

- **PV_SERVICE_ACCOUNT**

  Name of the service account used to access Kubernetes API to create/delete and inspect containers, persistent volume claims and network policies.

- **DOCKER_REGISTRY**

  Enter the address of the Container registry. This address must have a trailing slash, e.g. registry.example.com/

- **IMAGE_PULL_SECRETS**

  If you underlying container runtime cannot directly pull the images from the container registry, you can provide an array of pull secrets to access the container registry.

  Note: The mentioned pull secrets need to be created by you and are not part of the pure::variants Kubernetes deployment.

- **PV_REG_NUMBER**

  Enter your pure::variants license resp. registration number, which you for instance can find in your pure::variants license file. This number is used to access the floating licenses on the pure::variants License Server.

- **PV_LICENSE_SERVER**

Enter the address of the pure::variants License Server, e.g. https://pvlicenses.example.com.

- **PV_EXPOSED_HOST**

  This option defines the base network address used to access the pure::variants services. Enter the full name of the host, e.g. `pv.example.com`

- **RBAC_ENABLED**

  If set to `true` automatically creates a role and role binding for the service account.

- **TZ**

  Replace the default time zone `Europe/Berlin` with your time zone. This information is used to let the containers have the correct time according to your location. Please see https://en.wikipedia.org/wiki/List_of_tz_database_time_zones for the available time zones.

- **PV_INGRESS_CLASSNAME**

  Optional: You can set the name of the ingress class which shall be used for the ingress resources. If no name is provided it defaults to the `default` ingress class.

- **PV_SECRET_NAME**

  Optional: If you prefer to provide a TLS secret by yourself you can specify the name of the secret in the namespace used to encrypt the communication to the pure::variants services.

- **PV_SSL_CERTIFICATE**

  Path to the server certificate used to encrypt the communication to the pure:.variants services. Defaults to `workspace/certificates/server.crt` therefore the server certificate must be placed in this location of the helm chart.

- **PV_SSL_KEY**

  Path to the server certificate key used to encrypt the communication to the pure:.variants services. Defaults to `workspace/certificates/server.key` therefore the server certificate must be placed in this location of the helm chart.

6.4.2. webclient

- **enabled**

  Parameter to enable or turn off the Web Client in this pure::variants Deployment for Kubernetes. Defaults to `true` and therefore the Web Client is part of this deployment.

- **PV_MODEL_SERVER**

  Enter the address of the pure::variants Model Server to use, e.g. `https://pv.example.com/pv/`.

  Note: Can be kept empty if the Model Server is also deployed with this pure::variants Deployment for Kubernetes.

- **PV_WEB_CLIENT_SSO_GC_URI**

  Leave this option empty if you don't use Global Configurations nor OpenID Connect to authenticate. Otherwise enter the address of the Global Configuration service, e.g. https://jazz.example.com:9443/gc.

- **PV_WEB_CLIENT_HIDE_INACCESSIBLE_PROJECTS**

If set to true, all projects which are not accessible for the user are completely hidden instead of greyed out in the projects overview. Defaults to `false`.

- **PV_WEB_CLIENT_LOGLEVEL**

  Change the log level of the web client to increase or lower logging information. Defaults to `INFO`. Available options are: `INFO, DEBUG, TRACE`

- **PV_TRANSFORM_STORAGE_SIZE**

  Define the size of the requested persistent volume claim which stores all transformation job information. Defaults to `25Gi`

- **PV_RAM_MAX**

  Define the maximum heap memory which Java process can acquire inside the web Web Client container. The value is defined as Megabytes and defaults to: `16389`

- **PV_WEB_CLIENT_RESOURCES**

  Please define how much CPU and Memory the POD should request and be limited to. The default values are commented out and therefore no resource definitions are set on POD level.

  Note: If you define resource specifications please make sure to align to our hardware requirements see Section 2.4, “Web Components”

6.4.3. gateway

- **PV_GATEWAY_SSO**

  Enable OIDC as an authentication method for the pure::variants Gateway. Enabling OIDC automatically disables form based authentication method.

- **PV_GATEWAY_SSO_NAME**

  Provide a label for the configured Single-Sign-On provider. This name is used as a label of the OIDC button on the login page.

  Note: Only needed if OpenID Connect login is enabled.

- **PV_GATEWAY_SSO_URL**

  Enter the well-known endpoint of the OpenID Connect provider without the trailing /.well-known/openid-configuration. Example: `https://jazz.example.com:9643/oidc/endpoint/jazzop`

  Note: Only needed if OpenID Connect login is enabled.

- **PV_GATEWAY_SSO_CLIENT_ID**

  Enter the OpenID Connect client identifier for the pure::variants Web Client as defined at the OpenID Connect provider.

  Note: Only needed if OpenID Connect login is enabled.

- **PV_GATEWAY_SSO_CLIENT_SECRET**

  Enter the OpenID Connect client secret for the pure::variants Web Client as defined at the OpenID Connect provider.

  Note: Only needed if OpenID Connect login is enabled.

- **PV_GATEWAY_SSO_USERID_CLAIM**

  Optionally enter the claim which is used for the OpenID Connect token to identify the user id.

  Options are `idtoken:<claim>` or `userinfo:<claim>` where `<claim>` needs to be replaced with the name of the claim

  Note: Only needed if OpenID Connect login is enabled.

- **PV_GATEWAY_SSO_SCOPES**

  Change this option only if a different set of token scopes are needed. The default scopes requested are: `openid profile email`).

  Note: Only needed if OpenID Connect login is enabled.

- **PV_GATEWAY_ADDITIONAL_ANNOTATIONS**

  The pure::variants Gateway is exposed via an ingress route. If your environment or ingress controller needs a specific annotation please add it below this parameter with an array annotation.

6.4.4. runner

- **enabled**

  If set to `true` the transformation runners, which are handling transformations triggered in the Web Client, are also deployed with this pure::variants Deployment for Kubernetes.

- **PV_RUNNER_LOGLEVEL**

  Define the log level of the transformation runner to increase the logging entries. Available options are `1 to 7` and defaults to `1`.

- **PV_RUNNER_RESOURCES**

  Please define how much CPU and Memory the POD should request and be limited to. The default values are commented out and therefore no resource definitions are set on POD level.

6.4.5. runnercredentials

The configured transformation runner is authenticating against the Web Client to ensure no information is passed to non authorized transformation runners.

To add more runner please add more entries to the `values.yaml` parameter file.

```
runnercredentials:
  # to create multiple runners copy the following line like the example below:
  PV_RUNNER_CREDENTIALS_01: '{"id":"runner-01","apikey":"changeit"}'
  PV_RUNNER_CREDENTIALS_02: '{"id":"runner-02","apikey":"changeit"}'
```

6.4.6. executor

The transformation executor is a container with a temporary lifespan. It is created on demand to execute the triggered transformation and is deleted once the transformation is finished.

- **PV_CPU_MIN**

  Define the requested amount of CPUs for the transformation POD.

- **PV_CPU_MAX**

Define the limit of CPUs the transformation POD can obtain.

- **PV_RAM_MIN**

  Define the requested amount of Memory for the transformation POD.

- **PV_RAM_MAX**

  Define the limit of Memory the transformation POD can obtain.

Note: If none of the resource variables are set the transformation executor will not set any resource dependencies to its POD definition. If you choose to define resource dependencies please align them to our requirements see Section 2.1, “pure::variants Desktop Client”

6.4.7. modelserver

- **enabled**

  If set to `true` the pure::variants Model Server is deployed with this pure::variants Deployment for Kubernetes.

- **PV_MODEL_SERVER_WEB_PASSWORD**

  Enter the password with which you want to protect the status web page of the pure::variants Model Server. Defaults to `changeit`.

- **PV_MODEL_SERVER_APIKEY**

  Enter the API key with which you want to protect the API endpoints of the pure::variants Model Server.

- **PV_MODEL_SERVER_SYSTEMUSER_PASSWORD**

  Enter the password with which you want to protect the pure::variants Model Server superuser system.

  Note: This setting has no effect if an external database is connected to the Model Server.

- **PV_MODEL_SERVER_DB_PASSWORD**

  Enter the password with which you want to protect the local pure::variants Model Server PostgreSQL database or to connect to your external database.

- **PV_MODEL_SERVER_LOGLEVEL**

  Log level of the pure::variants Model Server, from 0 (just errors) to 9 (extensive logging). Defaults to level 1 for minimal logging.

- **PV_MODEL_SERVER_OPENID_CLIENT_ID**

  Client identifier issued to the pure::variants Model Server by the OpenID Connect provider.

  Note: Only relevant if the Model Server shall be connected with the Single-Sign-On provider.

- **PV_MODEL_SERVER_OPENID_CLIENT_SECRET**

  Client secret assigned to the pure::variants Model Server by the OpenID Connect provider.

  Note: Only relevant if the Model Server shall be connected with the Single-Sign-On provider.

- **PV_MODELSERVER_RESOURCES**

  Please define how much CPU and Memory the POD should request and be limited to. The default values are commented out and therefore no resource definitions are set on POD level.

Note: If you define resource specifications please make sure to align to our hardware requirements see Section 2.2, “Model Server with Database”

If you want to connect the Model Server to an **external database** please remove the # in front of the following parameters and provide a reasonable value. The Model Server provided with the pure::variants Deployment for Kubernetes shares the same requirements for its database as our standalone Model Server please see Section 2.2, “Model Server with Database”. Please note when using an external database, the internal PostgreSQL database needs to be disabled by changing the value of `database.enabled` to `"false"`.

- **PV_MODEL_SERVER_DB_TYPE**

  Provide the type of database the Model Server shall connect to. Options are: `PostgreSQL`, `MSSQL` and `Oracle`.

- **PV_MODEL_SERVER_DB_HOST**

  Enter the hostname which provides access to the external database.

- **PV_MODEL_SERVER_DB_PORT**

  Enter the port with which the external database can be accessed.

- **PV_MODEL_SERVER_DB_NAME**

  Enter the name of the external database which has been initialized with the init SQL script of the Model Server.

- **PV_MODEL_SERVER_DB_USER**

  Enter the name of the technical user having access to the external database.

6.4.8. database

This section provides parameters to configure the pure::variants Deployment for Kubernetes internal PostgreSQL database for the Model Server.

Note: For productive usage we highly recommend to use an external/managed database.

- **enabled**

  If set to `true` the internal PostgreSQL database is deployed with this pure::variants Deployment for Kubernetes.

- **POSTGRES_SIZE**

  Provide a size which is requested by the persistent volume claim associated with the database statefulset. Defaults to `25Gi`.

- **PV_DATABASE_RESOURCES**

  Please define how much CPU and Memory the POD should request and be limited to. The default values are commented out and therefore no resource definitions are set on POD level.

6.5. Installing the Deployment

Once the configuration is done and your images are available on the container registry you can proceed installing your pure::variants Deployment for Kubernetes.

For an installation from scratch please navigate into your helm chart and run the following command which will also create the desired namespace if it does not exist yet.

Note: Please make sure to replace `<name_of_deployment>` with the name helm should list your pure::variants Deployment for Kubernetes.

Also replace `<namespace>` with the desired namespace in Kubernetes which is defined in the parameter PV_NAMESPACE.

```
helm install <name_of_deployment> -n <namespace> --create-namespace .
```

If the command has been successful your console Output should show the following message:

```
NAME: <name_of_deployment>
LAST DEPLOYED: <Timestamp>
NAMESPACE: <namespace>
STATUS: deployed
REVISION: 1
TEST SUITE: None
```

Once the installation via helm is done you can verify the deployment by running:

```
kubectl get all -n <namespace>
```

You should see your configured pods are pulling their images and starting up. Once the Pod initialization is done all Pods should have the status `Running`.

6.6. Validate Correct Operation of Deployment

6.6.1. Model Server Operation

1. Validate Model Server Availability: Use browser to navigate to Model Server URL (make sure to have the slash at the end of the URL), e.g., to https://pv.example.com:443/pv/ . You should now see the basic status page of the Model Server. Try to log in with the password defined in `PV_MODEL_SERVER_WEB_PASSWORD`.

2. Validate Model Server Access via Desktop Client: Follow the instructions given in the *pure::variants Model Server Administration Manual* using the Model Server URL (make sure to have the slash at the end of the URL), e.g., https://pv.example.com:443/pv/ . Create at one pure::variants user with a name matching a valid Single-Sign-On user as described in the manual.

6.6.2. Web Client Operation

1. Validate Web Client Availability: Use browser to navigate to Web Client URL (make sure to have the slash at the end of the URL), e.g. to https://pv.example.com/pvweb/ . You should now see the Login page of the Single-Sign-On service. Try to log in with valid user name and credentials from the previous step. Validate Number of available Transformation Runners by switching in the Web Client to the Transformation Dashboard and check that the Idle count is equal to the number of configured Transformation Runners (Default is 1).

7. pure::variants Model Server

7.1. Install pure::variants Model Server

7.1.1. Installation Requirements

To be able to succesfully install the pure::variants model server you need to following:

- pure::variants server license
- pure::variants model server installer or pure::variants model server archive
- If you want to encrypt the communication with the model server a certificate is necessary
  - Please see Section 7.5.4, “HTTPS Server Options” for further information
- For the model server with database backend a database instance with sufficient access

- Please see the section called “Setup of the database and the ODBC data source”

7.1.2. pure::variants Database Model Server Installation

The installation of the pure::variants Database Model Server requires 3 steps:

- Setup of the database and the ODBC data source
- Installation of the pure::variants model server on the server host
- Basic setup of the model server

There are two alternative ways to install pure::variants, depending on your operating system. The different installation procedures are described below.

Upgrading from earlier versions prior to 7.0.0

If you already deployed the pure::variants Server and migrate to the pure::variants Server 7.0.0, updating the database is recommended to benefit from an optimized database layout. See the section called “Update database layout and data for databases created before pure::variants 7.0.0”.

Setup of the database and the ODBC data source

Before installing the pure::variants server, a database has to be created. Please read the documentation of your database system how to setup a new database.

For **Oracle**, the new database has to use the **AL32UTF8 character set** to ensure correct handling of multi-lingual content.

As the next step, the fresh database needs to be initialized. For this, execute the provided initialization script for your database system. The script will create the needed table structure, indices, and procedures. The script can be found in the *pure::variants Windows Installer Package*, which can be downloaded from our pure::variants update site. The script is named *Init<DatabaseProductName>.sql*.

After initialization of the database, create a user and grant privileges to the database. The user needs rights to execute SELECT, INSERT, UPDATE, DELETE, and PROCEDURES. Please consult the documentation of your database system how to perform this.

On the host which will run the pure::variants server, the client software for your database has to be installed and an ODBC data source in the Windows Management Console needs to be setup. Please consult the documentation of your database system how to perform this. Ensure to enable the **Query Timeout** option if available.

Setup ODBC Driver for PostgreSQL

The ODBC driver can be downloaded on the official PostgreSQL ODBC pages: https://odbc.postgresql.org/. For Windows go to the download section and the msi sub section to download the ODBC driver installer. The ODBC diver options ducoumentation can be found here: https://odbc.postgresql.org/docs/config.html.

If the database does not use the default schema "public", then the schema name has to be specified. This can be done on Windows by clicking "Datasource" -> "Page 2" and enter the following in the "Connect Settings" field (replace myschema with your schema name): SET SEARCH_PATH to myschema. On other operating systems it has to be defined in the ODBC settings.

Additionally disable the **LF <-> CR/LF conversion** option in the ODBC data source settings and enable the **Parse Statements** option in the data source settings.

Setup ODBC Driver for Oracle

The ODBC driver can be downloaded from https://www.oracle.com/de/database/technologies/instant-client/winx64-64-downloads.html. The **Oracle Instant Client Basic Package** and **ODBC Package** is needed.

The machine running the model server needs to have the **NLS_LANG** environment variable to be set to **.AL32UTF8**. Alternatively the **NLS_LANG** registry entry can be set in the Windows registry below **HKEY_LOCAL_MACHINE\SOFTWARE\Oracle**. Please consult the Oracle NLS FAQ for more details.

Setup ODBC Driver for MSSQL

The Driver can be downloaded from https://docs.microsoft.com/en-us/sql/connect/odbc/download-odbc-driver-for-sql-server.

Setup ODBC driver on Linux.

To use the pure::variants database model server on Linux the machine needs unixODBC installed. Please use the systems package manager to install it. In addition you also need to install and configure the ODBC driver for Linux corresponding to your database. See the following table for further information.

Database System	Information
PostgreSQL	• https://www.postgresql.org/docs/7.2/odbc-install.html • https://www.postgresql.org/docs/7.2/odbc-config.html For some Linux distributions the package repositories offer a postgreSQL ODBC package. For Ubuntu it is called *odbc-postgresql*. After installing this package just the configuration has to be done.
Oracle	https://oracle-base.com/articles/linux/create-an-odbc-dsn-on-linux
MSSQL	https://docs.microsoft.com/en-us/sql/connect/odbc/linux-mac/installing-the-microsoft-odbc-driver-for-sql-server?view=sql-server-ver15

Update database layout and data for databases created before pure::variants 7.0.0

The release of pure::variants 7.0.0 includes an updated database layout to support faster and more efficient storage of versions and branches. To enable these improvements for existing installations, the database table layout and stored procedures need to be updated. This update only needs to be executed once. It creates a database layout that is incompatible with earlier versions of pure::variants Server. Older versions of pure::variants Server (prior to 7.0.0) will not be able to start with the updated database and will cause an error message.

ATTENTION: Before proceeding with the following steps, ensure you have an up-to-date backup of your database. Shut down the pure::variants Server while executing the migration scripts.

Depending on the current database layout version, different scripts need to be executed. The scripts can be found in the “pure::variants Windows Installer Package,” which can be downloaded from our pure::variants update site. The scripts must be executed by the database administrator with sufficient rights. To get the current version, first run the following SQL query inside the database.:

```
SELECT version FROM pvtableinfo WHERE tableid='modeltables';
```

For version 1.0 run the following scripts in the given order:

```
Update_1_<DatabaseProductName>.sql
Migrate_1_<DatabaseProductName>.sql
Update_2_<DatabaseProductName>.sql
```

For version 2.0 run the following script:

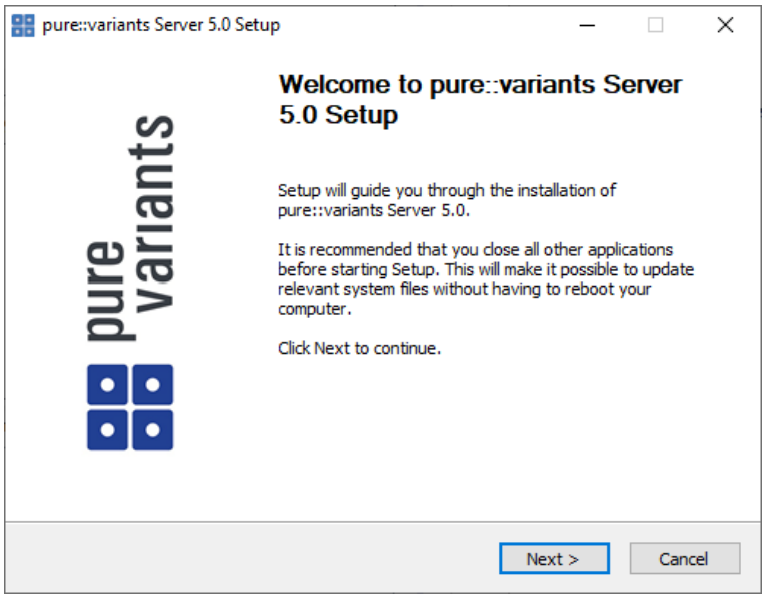
```
Update_2_<DatabaseProductName>.sql
```

Install with Windows Installer (Windows Only)

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.

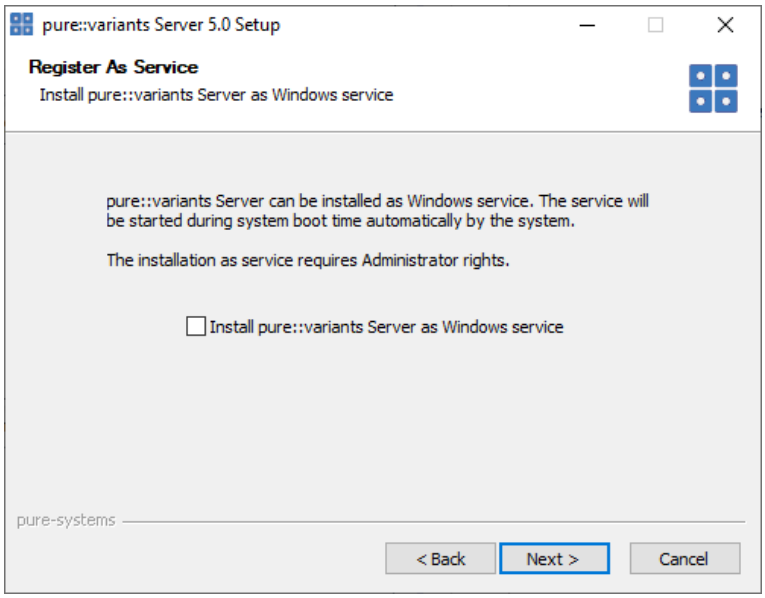
Download the model server installer *Setup Database Server X.Y.ZZ.exe* and start the installation by double-clicking the executable. Running the model server installer requires Administrator privileges.

Figure 59. Setup Model Server Installer



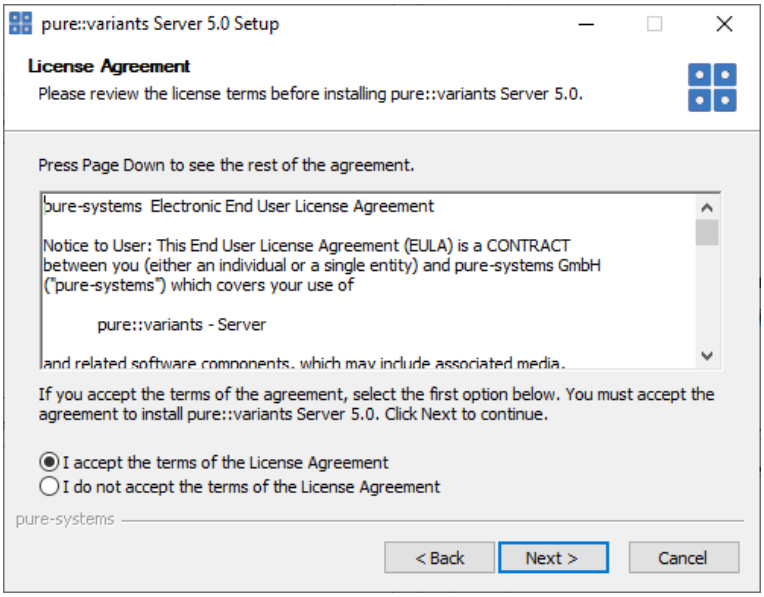
Click *Next*.

Figure 60. Setup Model Server Windows Service



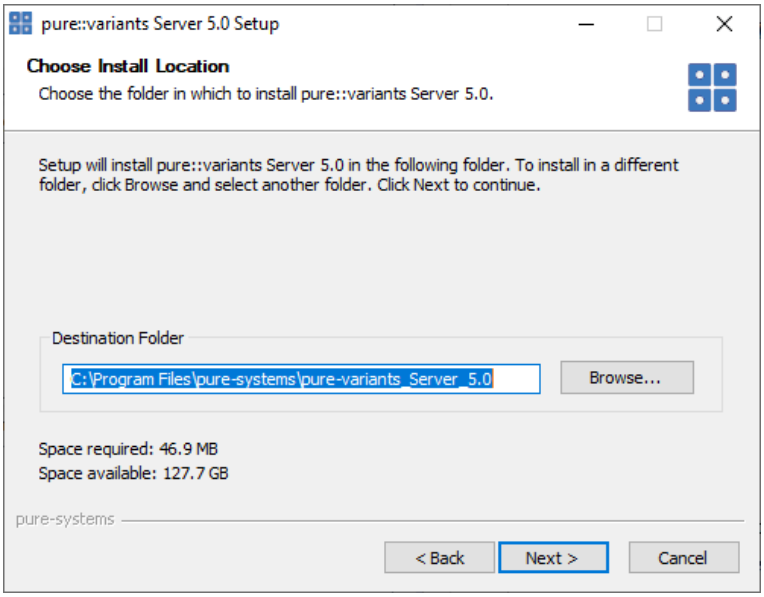
Decide whether the model server should run as Windows service. It is recommended to run the model server as Windows service. This will ensure that the model server will be started automatically during system startup. Click *Next*.

Figure 61. Setup Model Server License



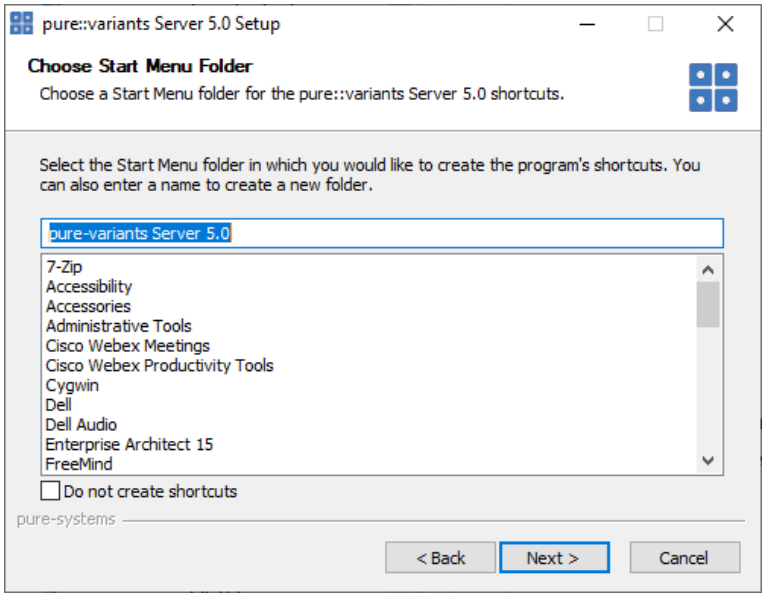
Read the license agreement, and after accepting it click *Next*.

Figure 62. Setup Model Server Installation Location



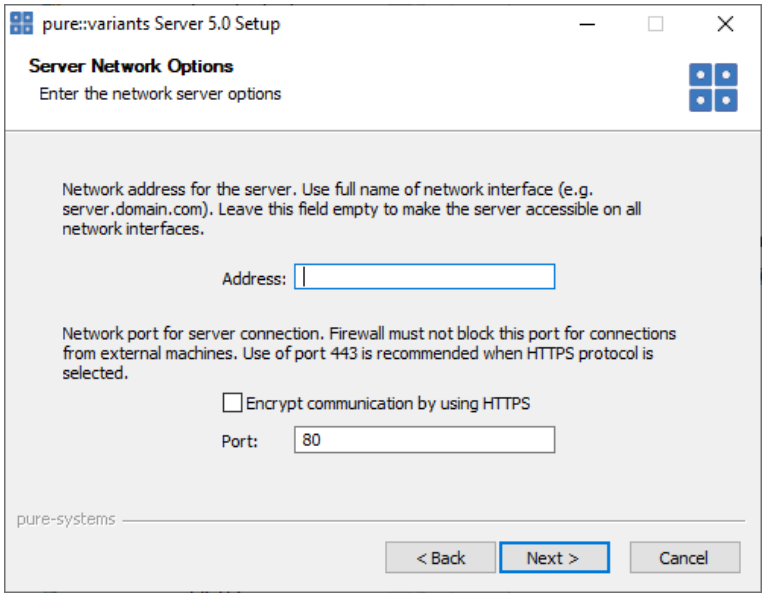
Select the folder where to install the model server files. Click *Next*.

Figure 63. Setup Model Server Start Menu



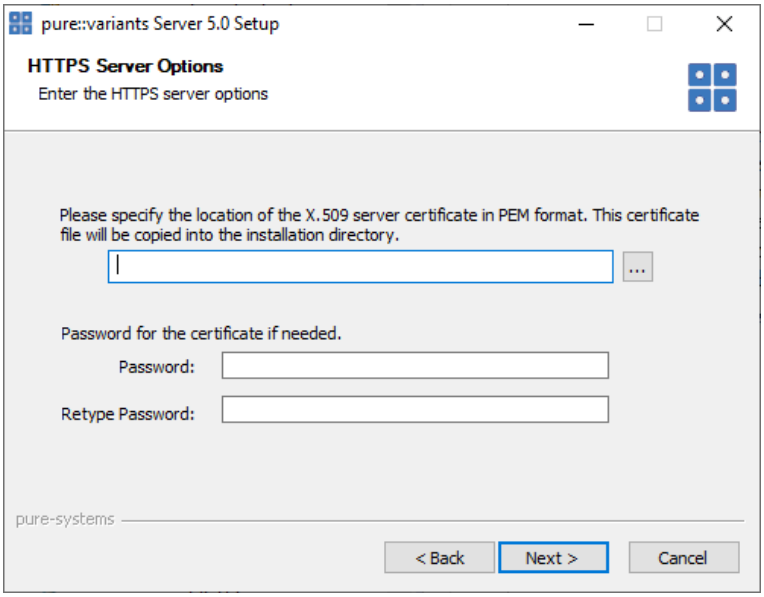
Enter the name of the Windows start menu entry, or disable the creation of the start menu entry. Click *Next*.

Figure 64. Setup Model Server Network Options



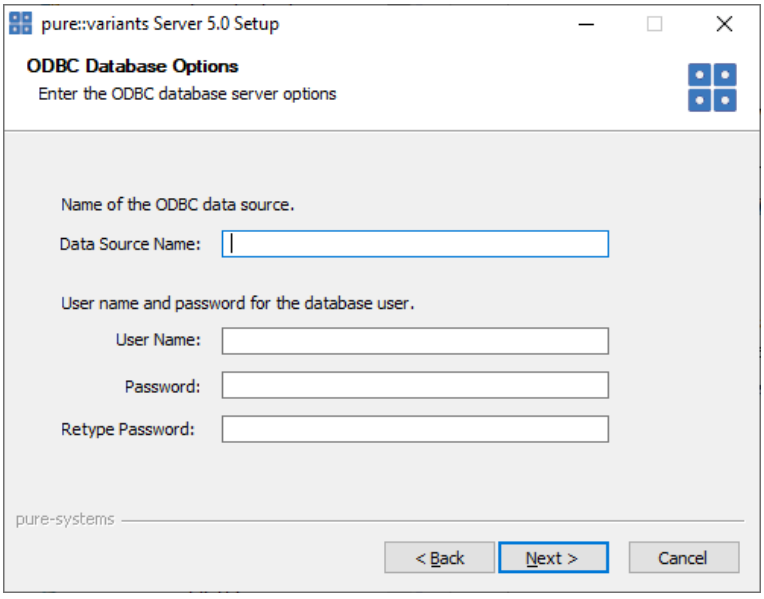
Now configure the network options. The *Address* field specifies the hostname or IP address of the network interface on which the model service should run. If you leave this field empty, the model server will automatically work on all available network interfaces. You can encrypt the whole communication by selecting the *Encrypt communication by using HTTPS* option. This will use HTTPS instead of HTTP. This requires an X.509 certificate for the model server. The certificate setup is done on a separate page in the installer. The port for communication can be specified in the *Port* field. The default values (80 for HTTP and 443 for HTTPS) should work best. Please ensure that this port is not blocked by a firewall or used by another service. Click *Next*.

Figure 65. Setup Model Server Encryption



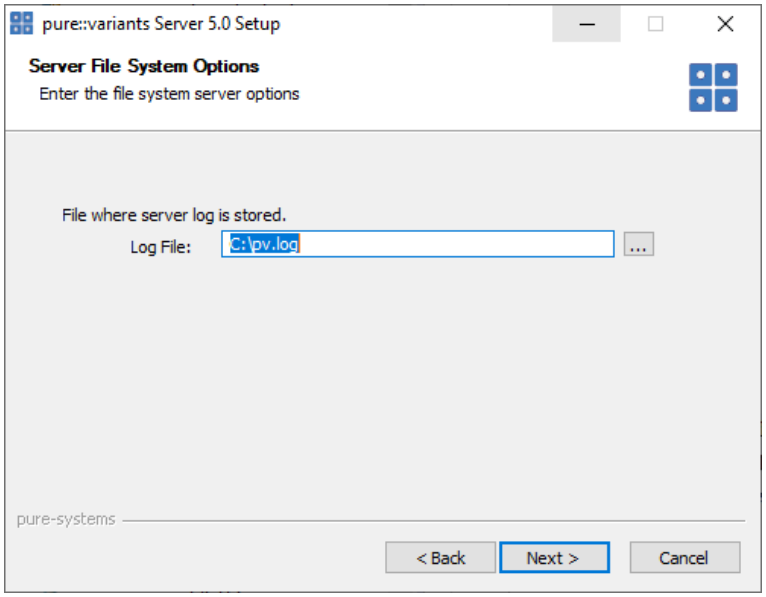
If encryption was enabled, enter the path to the X.509 certificate into the upper file field. The format of the certificate needs to be PEM. No other types are supported. If the certificate is protected by a password, enter the password into the two password fields. Click *Next*.

Figure 66. Setup Model Server ODBC Data Source Settings



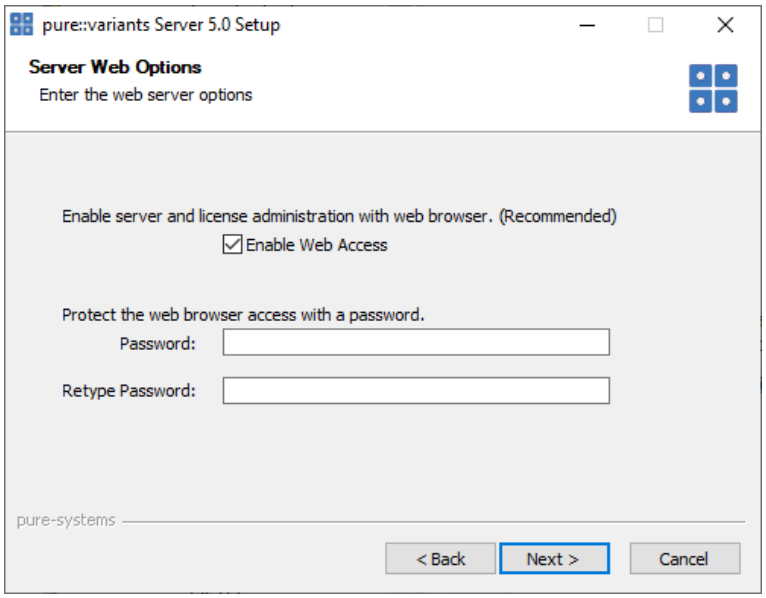
Enter the ODBC data source name which was created for pure::variants to communicate with the database. To access the database, pure::variants needs the credentials of a user with sufficient access to the database. Click *Next*.

Figure 67. Setup Model Server Log



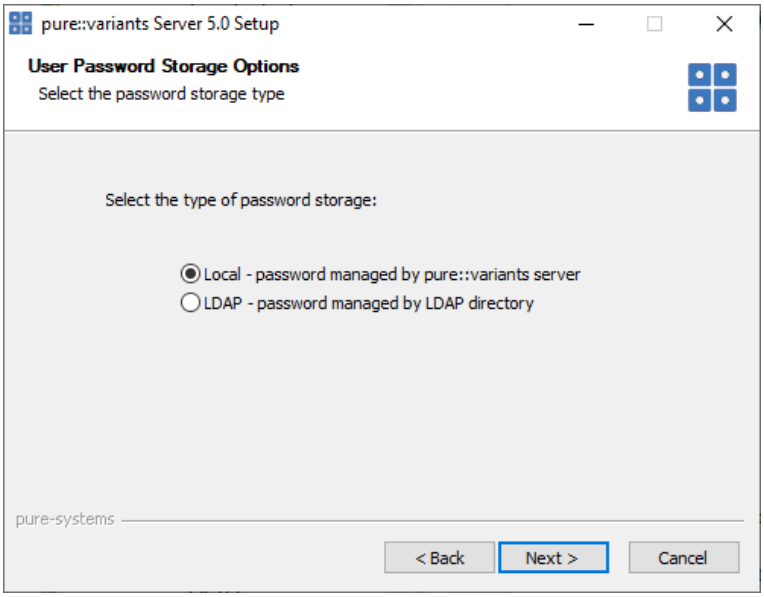
Enter a location for the server log file. The pure::variants server needs to have write access to the specified file. Click *Next*.

Figure 68. Setup Model Server Web Interface



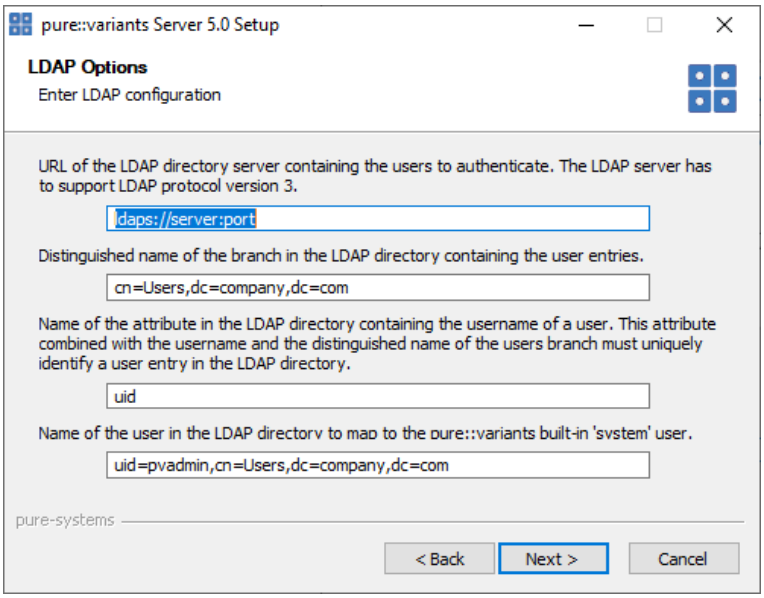
The model server provides a web interface. The web interface can be enabled or disabled on this page. Please secure the web interface with a password. The web interface should be used with connection encryption only, in order to prevent possible risk of password theft. Click *Next*.

Figure 69. Setup Model Server Password Management



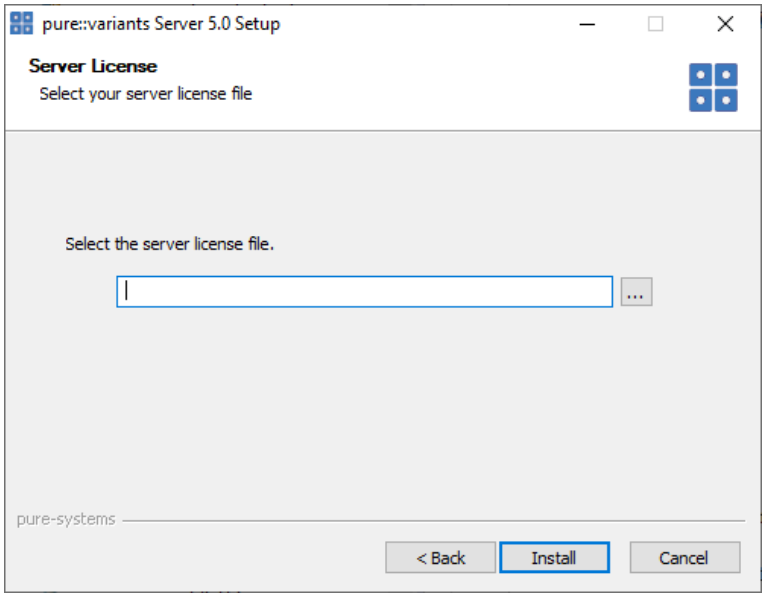
The model server is capable of managing password locally, or can access a LDAP directory for user authentication. Please chose the password management method. Click *Next*.

Figure 70. Setup Model Server LDAP Settings



This page allows you to setup your specific LDAP settings. Click *Next*.

Figure 71. Setup Model Server License File



As the last step, the license file for the pure::variants model server needs to be specified. Clicking *Install* will start the installation process.

If the Windows service option was chosen, the model server will start automatically after successful installation.

Install from Archive

If using the installer is not an option, the server is also distributed as a compressed archive file.

Download the pure::variants Database server archive 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 server license file provided to you by Parametric Technology.

Create a directory for the pure::variants model server on your disk and extract the contents of the archive into that folder. For Linux systems a recommended location is the directory */opt/pure-variants*. On Windows a recommended location is the directory *C:\Program Files\pure-systems\pure-variants_Server_6.0*

Create a directory for the license files. A proper location for Linux systems is */opt/pure-variants/licenses* and on Windows *C:\Program Files\pure-systems\pure-variants_Server_6.0\licenses*. Copy the provided server license file into that folder.

Open the script *start.bat* on Windows resp. *start.sh* on Linux in a text editor. The script is located in the *server* sub-directory. Update the following variables to match your environment:

HOSTNAME: The hostname or IP address of the server machine the server should bind to.

PORT: The TCP port used for communication.

ODBCDSN: The name of the ODBC data source.

ODBCUID: The name of the database user.

ODBCPWD: The password of the database user.

Clients must be able to connect to this port on the server machine. Therefore this port must not be blocked by a firewall or used by another service. A port number can range from 0 to 65535. Port 80 is the standard port for the HTTP and 443 for the HTTPS protocol. Using these standard ports can help with firewall problems.

The server can be now started by running script *start.bat* on Windows resp. *start.sh* on Linux. Please consult your system documentation how to add this script to the systems start sequence to ensure the server is started automatically after system reboot. You are now able to connect to the pure::variants Model Server with your pure::variants Enterprise clients.

On Windows the pure::variants server can be registered as a service. Selecting this option ensures that the pure::variants server always runs after reboot even when no user is logged in. To register the puer::variants server as a service, use the command line option */install*. If installing the pure::variants license server as a service, a service name has to be given with command line option */servicename*. A service description can be added with command line option */servicedesc*.

```
<license server install path>/server/variantsd.exe /install
/servicename "pure::variants License Server" /servicedesc "The license server is providing
licenses for the pure::variants clients"
```

Enable pure::variants Model Server Web Interface

The pure::variants model server includes an optional web interface for monitoring and interacting with the server.

To enable the server's web interface on Windows, add the following options to the server command at the end of the *server/start.bat* script:

```
/enableweb
/webpwd PASSWORD
```

To enable the server's web interface on Linux, add the following options to the server command at the end of the *server/start.sh* script:

```
--enableweb
--webpwd PASSWORD
```

The server's web interface should be always protected by a password. Please use a secure password for this.

If the Model Server Web Interface should be accessible with the help of a context path, add the following options to the server command at the end of the *server/start.bat* script. For example if you want the Model Server to be accessible with context */pvmodelserver* please replace *BASEPATH* with *pvmodelserver* in the command below:

```
/webbasepath BASEPATH
```

To define the base path of the Model Server Web Interface in linux, add the following options to the server command at the end of the *server/start.sh* script:

```
--webbasepath BASEPATH
```

7.1.3. Basic pure::variant Model Server Setup

The initial configuration of the pure::variants server only knows one account, i.e. the special account “system”. If the server is configured to use “local” authentication, which is the default, this account has a default password which is the same for all pure::variants server installations. Thus, it **MUST BE CHANGED** immediately after starting up the server for the first time. Keeping this password renders the server insecure. The password is “sXg9C58JcmPB“ (without the double quotes).

Start the pure::variants server as follows. If installed as service, start the service manually using the Windows management console, or reboot. If not installed as service, start the server from the Start menu (in this case the server terminates upon logout).

The process of changing the password of a user is documented in the online help of the pure::variants Desktop Client (*Help -> Help Contents -> pure::variants Server Administration plug-in manual*). It requires the import of the “ADMIN” project from the server. See the online documentation for information on the import process.

The recommended next step is to setup an administrative user different from the special “system” user. Using the newly created server administrator account, new users and roles may be added.

7.2. Update pure::variants Model Server

7.2.1. Update with Windows Installer (Windows Only)

Updating an existing pure::variants model server works exactly the same as a clean installation of the pure::variants model server. Please see Section 4.2.1, “Install with Windows Installer (Windows only)”. Additional settings performed in the server configuration file will remain in the server configuration file after updating.

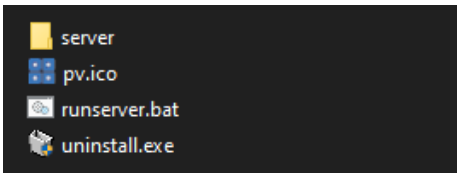
7.2.2. Update with Archive

Updating an existing pure::variants model server instance works exactly the same as a clean installation of the pure::variants model server. Please see Section 4.2.2, “Install from Archive”.

7.3. Uninstall pure::variants Model Server

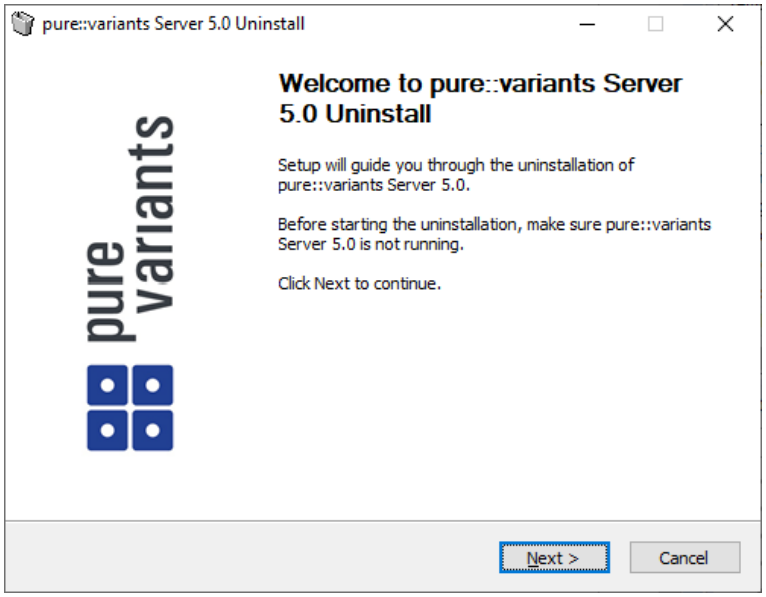
The uninstaller for the pure::variants model server can be started in two different ways. The first one is to go to the Windows *Add or remove programs* application and search for *pure::variants Server 6.0* and start the uninstaller by using the *Uninstall* action. The uninstaller requires Administrator privileges.

Figure 72. pure::variants Model Server Uninstaller



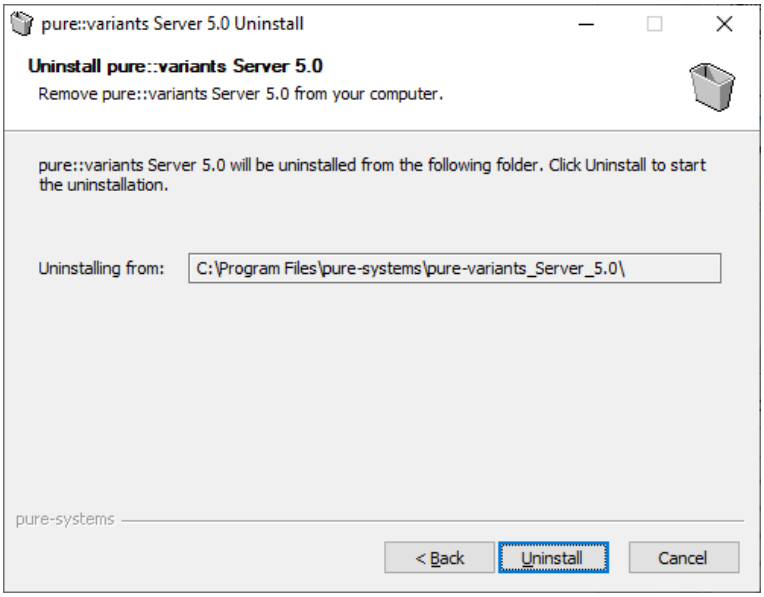
The second possibility is to navigate to the pure::variants model server installation folder and start the uninstaller by double-clicking it.

Figure 73. pure::variants Model Server Uninstaller



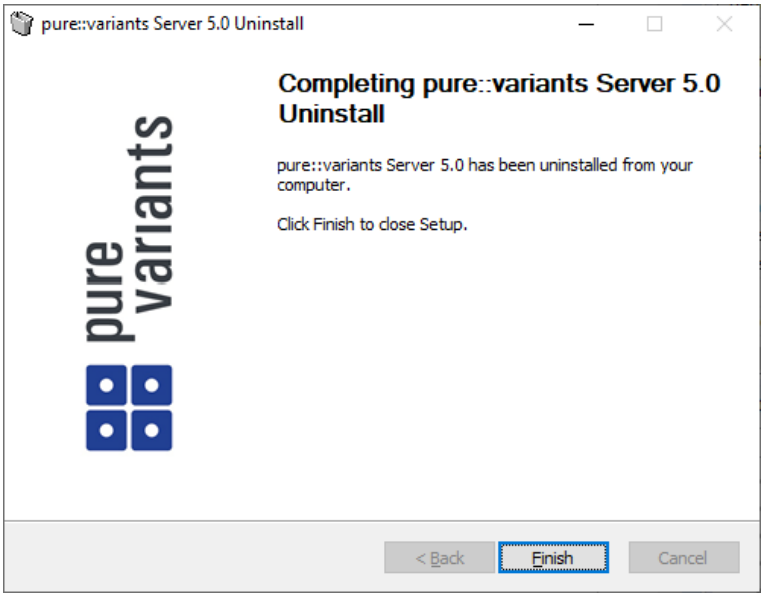
Click *Next*.

Figure 74. Uninstall from



Click *Uninstall* to start the uninstall process.

Figure 75. Completing Uninstall



The deinstallation is succesfully finished. Click *Finish* to close the uninstaller.

7.4. Location of the Server Configuration File

7.4.1. On Windows

You find the server configuration file “server.cfg” in the directory “pure-variants”, which either is located in the sub-directory “server\etc” of the installation directory of the pure::variants Server (e.g., “C:\Program Files\pure-systems\pure-variants_Server_6.0\server\etc”), or in the home directory of the current user.

7.4.2. On Linux

You find the configuration file “server.cfg” in the directory “pure-variants”, which either is located in the sub-directory “server/etc” of the installation directory of the pure::variants Server (e.g., “/opt/pure-systems/pure-variants_Server_6.0/server/etc”), or in directory “/etc”, or in the home directory of the current user.

7.5. Explanation of Installation Options and Parameters

All options are stored as command line settings in the server configuration file “server.cfg”.

7.5.1. Server Network Options

The pure::variants server uses the SOAP over HTTP/HTTPS protocol to communicate with its clients. The Server Network Options define the URL that users have to enter to reach a server installation. The URL for a server looks like this:

<Protocol>//<Address>:<Port>

Address: The hostname of the server machine. This will be the <Address> part of the URL.

Encryption of client/server communication: When the HTTPS protocol is used, all communication between clients and server is encrypted. This requires an X.509 certificate for the server. Certificate setup is done on a separate page in the installer.

Port: The TCP Port used for communication. Clients must be able to connect to this port on the server machine. Therefor this port must not be blocked by a firewall or used by another service. A port number can range from 0 to 65535. Port 80 is the standard port for the HTTP and 443 for the HTTPS protocol. Using these standard ports can help with firewall problems.

7.5.2. Install as Windows Service

The pure::variants server may be started either manually by the user from the Start menu, or automatically as Windows Service when Windows starts. Selecting this option applies the required changes to the Windows registry to register pure::variants as a service. This requires administrative rights during installation. Selecting this option ensures that the pure::variants server always runs after reboot even when no user is logged in.

7.5.3. Automatic reconnect to database after connection is lost

To ensure data consistency the pure::variants model server forces a restart after the connection to the database is lost. The restart ensures that all internally cached data matches the data stored in the database.

For environments with unstable network connections the pure::variants model server is able to automatically re-connect to a database when the connection is re-established. To enable automatic reconnect the option **/odbcautoreconnect** needs to be added to the configuration file. When enabled the server tries to reconnect again to the database when a new database operation is started.

With automated reconnection there is a high risk, that data corruption, when the database is stopped and changes are made on the pure::variants table space like restoring a backup. The operator has to ensure that the pure::variants model server is stopped and restarted when such operations are performed.

7.5.4. HTTPS Server Options

The server includes the option to secure the network communication between the server and clients so that the transmitted data cannot be intercepted by others.

The encrypted communication makes use of an X.509 certificate containing the public and private parts of the certificate in one file. Smartcard-based certificates are currently not supported.

Such a certificate either can be obtained from a (official) certification authority (preferred) or can be a self-signed certificate. If the certificate is not issued by an official certification authority, all pure::variants users have to

register this certificate or the root certificate of the internal certification authority in the pure::variants (Java) key store. Instructions how to do this are given below.

A self-signed certificate should be used only as temporary solution until an official certificate has been provided. It can be created for instance using the *openssl* tool like this:

```
openssl req -newkey rsa:2048 -nodes -keyout key.pem -x509 -days 365 -out certificate.pem
```

Preparation of the PEM file to be used with pure::variants Server

The PEM formatted file containing the public server certificate and private key should look like this.

```
           -----BEGIN ENCRYPTED PRIVATE KEY-----
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
...
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
-----END ENCRYPTED PRIVATE KEY-----
-----BEGIN CERTIFICATE-----
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
...
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
-----END CERTIFICATE-----
```

If you have one file for the private key and one for the public certificate, then copy both together in one file using a text editor.

For security reasons, do not place this file in a path that is exposed to users other than the one running the pure::variants Server.

Configuration of the pure::variants Server for encrypted communication using a server configuration file

To enable encryption for an already installed pure::variants Server, the server configuration needs to be extended.

Open the configuration file in a text editor. If the file does not exist, create it. Add the following lines to the contents of the file.

```
/sslkeyfile "path\to\server-cert.pem"
/sslpassword "password of private key"
```

Replace the path to the PEM file and the password of the private key accordingly.

Configuration of the pure::variants Server for encrypted communication using command line options

To achieve this you have to modify the pure::variants server startup command line to include a reference to the key.

To enable the use of the certificate on Windows, add the following options to the server command at the end of the *start.bat* script:

```
/sslkeyfile <path_to certfile.pem>
/sslpassword certpassword
```

To enable the use of the certificate on Linux, add the following options to the server command at the end of the *start.sh* script:

```
--sslkeyfile <path_to certfile.pem>
--sslpassword certpassword
```

Usage of a self-signed server certificate with the pure::variants Desktop Client

The pure::variants Desktop Client uses the Java services to encrypt the communication with the pure::variants Server. Java requires all server certificates to be signed by a known certification authority, or to be registered as trusted certificate using a separate trust store.

If the pure::variants Server uses a self-signed certificate, all pure::variants Desktop Clients need to import this certificate into the pure::variants Desktop Client's trust store. The following steps show how to do this using the Java tools.

Step 1: Add the public server certificate to a Java Key Store

The tool keytool is part of the Java tools distributed with the JRE or JDK. It is located in the bin directory inside the Java installation.

Change to the home directory of the user running the server (%USERPROFILE%), and execute following command.

```
keytool -import -alias psroot -file certfile.pem
```

If you are asked to trust the certificate, then answer with Yes. This will create the file .keystore in the current directory. You may have to add further trusted certificates to this trust store if you need to access other remote sites using HTTPS from within pure::variants.

Step 2: Use the Java Key Store with the pure::variants Desktop Client

The two most common ways to provide the pure::variants Desktop Client with the server certificate are to extend the command line in the Desktop or Windows Start Menu link used to start the pure::variants Desktop Client, or to extend file *eclipse.ini* in the pure::variants Desktop Client installation directory.

To extend the command line, open the “Properties” dialog of the pure::variants Desktop Client link and append the following after the *-vmargs* option.

```
-Djavax.net.ssl.trustStore="%USERPROFILE%\.keystore"
-Djavax.net.ssl.trustStorePassword=keystorepassword
```

Replace the key store path and password accordingly.

Step 3: Use the Java Key Store with the pure::variants in-tool integration

If any in-tool integration (e.g., pure::variants integration for IBM Rational DOORS, or Microsoft Office) is used, the *trustStore* and *trustStorePassword* properties also need to be set for in-tool integrations. To do this, append

```
javax.net.ssl.trustStore=%USERPROFILE%/.keystore
javax.net.ssl.trustStorePassword=certpassword
```

to file:

```
C:\ProgramData\pure-variants-6\pv.properties
```

If “pv.properties” does not exist, create it and the necessary folders. Please note that slashes in the trustStore path need to be either forward slashes, or escaped slashes (“/” or “\\”). Otherwise the path cannot be read.

```
“C:\\” => “C\:\\” resp. “C:/” => “C\:/”
```

Please remember not to distribute the certificate file with the public and private part to the users! Only the public part of the certificate is required in this step! If an empty password was given in step 1 the second part of the command line can be omitted. The keystore file created in step one can be put on a shared directory and used

by all users. In this case the path in the first part of the command line has to point to the shared location of the key store file.

7.6. Setup Authentication for pure::variants Model Server

7.6.1. Open ID Connect Authentication

The pure::variants Server can be configured to authenticate users using an Open ID Connect provider. This enables single sign-on inside Eclipse between different applications using the same provider.

Open ID Connect authentication is enabled for the pure::variants Server by setting the command line option **/logon** to the value **openid**. Before the first start using Open ID Connect, the server needs to register at the provider. This is done by starting the server on the command line with the **/openidregister** option. The provider URL, the user and the password are set by the corresponding **/openidurl**, **/openiduser**, and **/openidpass** options. The following command line shows an example for a registration.

```
server/bin/variantsd.exe /config server/etc/pure-variants/server.cfg /openidregister
  /openidurl https://openid.company.com/oidc/endpoint /openiduser admin /openidpass password
```

If the connection to the Open ID Connect provider fails due to certificate or other SSL errors, you can use command line option **/openidignoresslerror** to let the pure::variants server ignore such errors. But beware, such errors may indicate that something is wrong with the Open ID Connect provider. Use this option with care. The same is true for the command line option **/openidnorevoke** which can be used to disable the check whether the certificate of the Open ID Connect provider has been revoked.

The registration data will be saved to file **openid.json** in the server configuration directory. After successful registration the server needs to be restarted.

If automatic registration at the Open ID Connect provider is not possible, the **openid.json** file has to be created by hand. This file has the following format:

```
{
    "client_id" : "insert here the client ID",
    "client_secret" : "insert here the client secret",
    "http_flags" : 0,
    "provider_uri" : "insert here the provider URL",
    "redirect_uris" :
    [
        "https://localhost/pv/openid"
    ],
    "scope" : "openid"
}
```

In order to get the client ID and client secret, you have to register the pure::variants server by hand at your Open ID Connect provider. Then copy the client ID and secret and insert both together with the URL of the Open ID Connect provider in the **openid.json** configuration file. If you configured a client URL other than "http://localhost/pv/openid", then you need to replace this URL in the configuration file too.

The **http_flags** setting in the configuration file can have a value between 0 and 3. Value 0 means that all security checks for the communication with the Open ID Connect provider are enabled. Value 1 means that certificate or other SSL errors are ignored, as if option **/openidignoresslerror** is used. Value 2 means that certificate revocation is not checked, as if option **/openidnorevoke** is used. Value 3 is the combination of values 1 and 2, as if both options **/openidignoresslerror** and **/openidnorevoke** are used.

Changes on the **openid.json** configuration file are not recognized by the pure::variants server when it is running. You need to restart the pure::variants server for the new configuration to apply.

The registration is stored until an unregistration is performed. For unregistration the **/openidunregister** option is used together with the **/openiduser** and **/openidpass** option.

The server uses the *userinfo* endpoint to get the name of the user. Is get the value from the "sub" field of the returned data. If the used Open ID Connect provider stores this data into an other field the **/openidloginspec** option can be

used to specify the field. It allows also to get the user name form any field of the id token of the *token* endpoint. The option argument starts with the place followed by a ':' and the field name e.g. *userinfo:sub* or *idtoken:sub*.

7.6.2. LDAP Authentication

The pure::variants Server can be configured to authenticate users against an LDAP directory. Users known to the pure::variants Server can then login to the server using their LDAP password. The server will use the provided username and password to bind to the LDAP directory. If this succeeds, the user is considered to be authenticated.

For this to work the LDAP directory server needs to support version 3 of the LDAP protocol. It is tried at most 3 times to access the LDAP directory for each bind attempt. If the LDAP directory does not respond within 60 seconds,the bind attempt is aborted and the authentication of the user will not succeed.

To use LDAP authentication you either have to enable and configure it during installation of the pure::variants Server, or add corresponding options to the configuration file of the pure::variants Server. You find the configuration file either in the installation directory of the pure::variants Server, e.g. “C:\Program Files\pure-systems\pure-variants_Server_6.0\server\etc\pure-variants\server.cfg”, or in the home directory of the user, i.e. “%APPDATA%\pure-variants\server.cfg”.

Following options control the LDAP authentication performed by the pure::variants Server.

/logon ldap

The default authentication scheme of the pure::variants Server is “local”, which means that the passwords of the users are managed by the pure::variants Server itself in a password file or database. With the logon type “ldap” the pure::variants Server does not manage the passwords by itself but uses an LDAP directory server to authenticate users. Note that in this case it is not possible to change the password of users in pure::variants because the passwords are managed by the LDAP directory and need to be changed there.

/ldapurl [URL]

The URL of the LDAP directory server containing the users to authenticate. The format of this URL is defined by RFC 2255 and basically looks like this:

- ldap://server:port
- ldaps://server:port

Beside protocol “ldap” also protocol “ldaps” for secure LDAP connections is supported by the pure::variants Server. It is strongly recommended to use the “ldaps” protocol if supported by the LDAP directory server. The default port when using the “ldap” protocol is 389, and 636 when using the “ldaps” protocol.

/ldapusersdn [DN]

The sub-tree in the LDAP directory containing the user entries. It has to be given as full distinguished name (DN) according to RFC 4514 and basically looks like this:

- cn=Users,dc=company,dc=com

If no LDAP bind user is given (see option /ldapbinduser), it is expected that all user entries are listed flat in this sub-tree. If an LDAP bind user is given, then nested organization of user entries is supported. The pure::variants Server will then search the whole sub-tree to find a matching user for a given username.

/ldapuidattr [NAME]

The name of the LDAP attribute containing the username of a user in the LDAP directory. Example:

- uid

To authenticate a user against the LDAP directory, the full distinguished name of the user entry is needed. This distinguished name uniquely identifies the user in the LDAP directory and consists of the

users sub-tree distinguished name (see option /ldapusersdn), the username, and this LDAP attribute (e.g. uid=username,cn=Users,dc=company,dc=com), containing further name parts if the users are organized in a nested instead of a flat tree structure (see option /ldapbinduser).

/ldapsysuser [DN]

A user from the LDAP directory to be used whenever a login as the pure::variants built-in “system” user is requested. As a consequence the password of this LDAP user needs to be used instead the documented “system” user's default password. If the user is not located in the users sub-tree of the LDAP directory (see option /ldapusersdn), then the full distinguished name of the user needs to be given. Otherwise the simple username can be used. Examples:

- uid=pvadmin,cn=Users,dc=company,dc=com
- pvadmin

/ldapbinduser [DN]

A user from the LDAP directory with search rights on the username attribute (see option /ldapuidattr) in the users sub-tree of the LDAP directory (see option /ldapusersdn). Example:

- uid=pvadmin,cn=Users,dc=company,dc=com

This option is not needed if the user entries are organized flat in the LDAP directory. But if they are organized in a nested tree structure, then this user is needed by the pure::variants Server to find users by their username.

/ldapbindpass [PASSWORD]

The password of the LDAP user configured with option /ldapbinduser.

Verify LDAP Configuration

It is recommended that a changed LDAP configuration is verified before the pure::variants Server is started. Otherwise it may not be possible for any user to login to the pure::variants Server.

For this purpose the pure::variants Server provides the ldaptest tool, located in the server\bin sub-directory of the pure::variants Server installation directory.

This tool supports the same command line options and configuration file settings as the pure::variants Server. If all settings have been done in the server configuration file, then the tool can be run without further arguments. Just double-click it or open a command prompt in the server\bin directory and enter:

ldaptest

Please follow the instructions on the screen. The tool tries to authenticate a given user that must exist in the LDAP directory. And it tries to authenticate the system user in the same way. If an LDAP bind user is given, then the tool tries to search a given user in the LDAP directory using the bind user before the authentication is tested.

This is an example of running the tool using command line arguments:

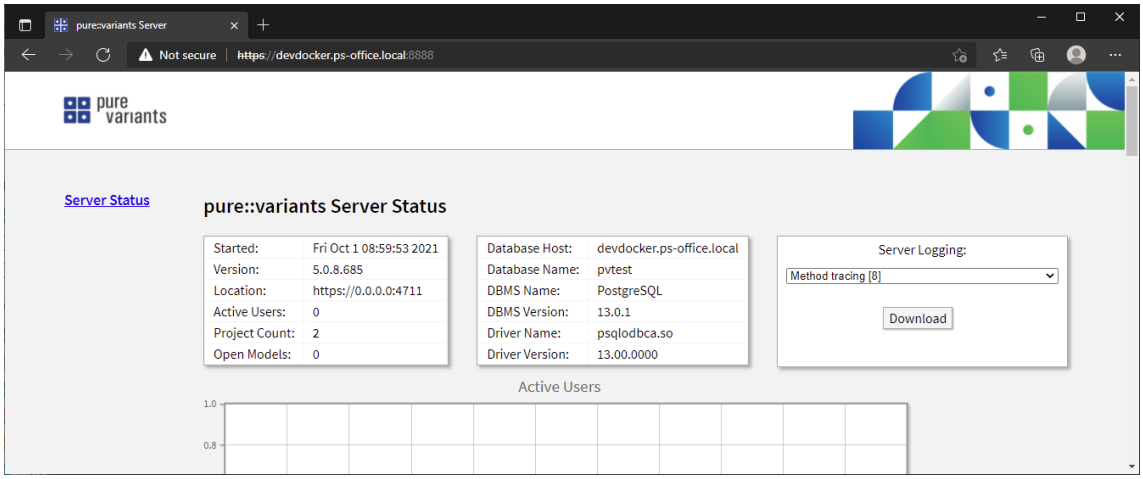
ldaptest /ldapurl "ldap://ldap.company.com:389" /ldapusersdn "cn=Users,dc=company,dc=com" /ldapuidattr "uid" /ldapsysuser "pvadmin"

The LDAP configuration is correct if the tool reports that the test successfully finished.

7.7. Model Server Web Interface

The pure::variants model server provides a simple web interface, which has to be activated during the installation process. Please accure that the access to the web interface is protected by a password during setup.

Figure 76. Server HTTP interface

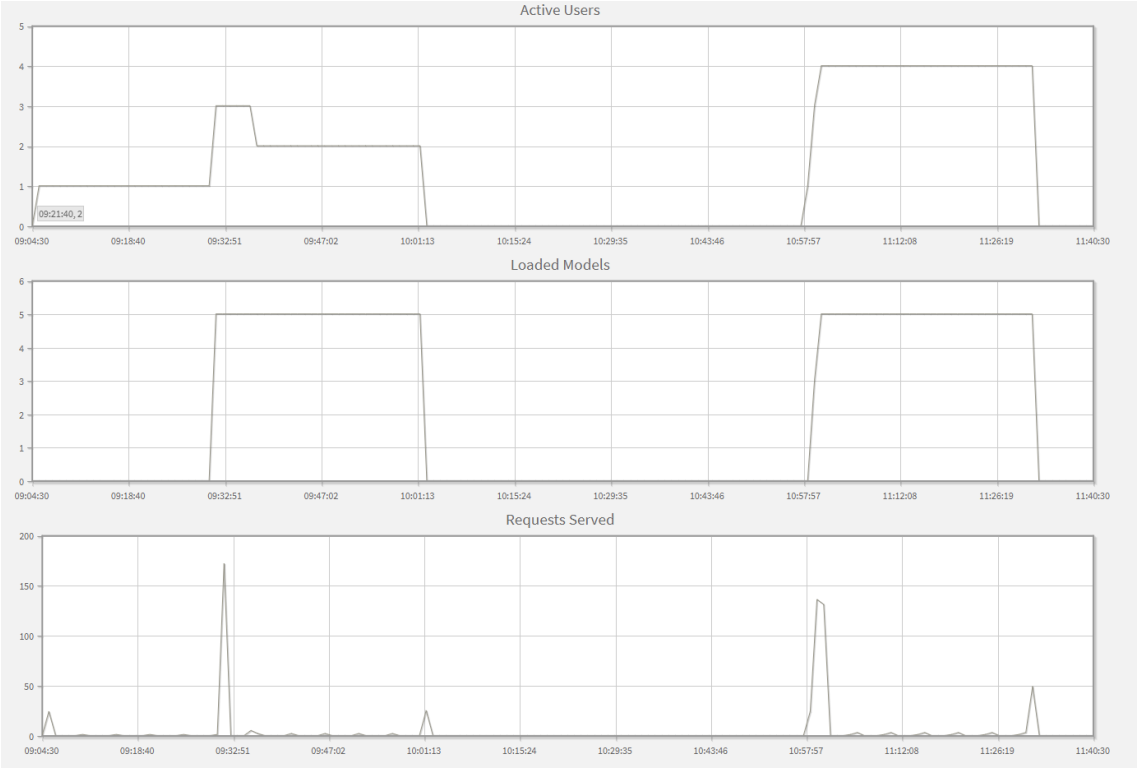


As shown in figure the web interface provides information about the start time, server version and location. In addition information about the connected database and the used ODBC driver are shown. On the right site the logging level of the server can be switch by selection the new level from the drop down box. The current active log file can be downloaded by pressing the download button.

7.8. Resource Monitoring

All pure::variants server provide monitoring of internal resources. The collected values are shown in the servers HTTP interface for the last 24 hours. Depending on the resource, values are absolute counts or maximal/minimal values collected during the monitoring interval of 1 minute.

Figure 77. Server HTTP interface with resource monitor data



The resource monitoring is enabled by default. It can be disabled with the **/disableresourcemonitor** option.

The following list shows the monitored resources.

Resource	Description	Remark
net.requests	Number of client requests served	all server
sessions.active	Maximal count of active users/sessions	model server only
models.loaded	Maximal count of loaded models	model server only
db.read.rows	Total number of database rows read	database server only
db.read.duration	Total time (in ms) needed to read the rows	database server only
db.write.rows	Total number of database rows written	database server only
db.write.duration	Total tin (in ms) needed to write the rows	database server only
license.free.REGNR	Minimal count of free licenses for REGNR	license server only
license.used.REGNR	Maximal count of used licenses for REGNR	license server only

7.8.1. Long term storage of resource monitoring data

For long term storage and analysis of monitoring data the server is able to push all values to an InfluxDB instance. This is enabled by providing the URL to an InfluxDB server in the **/influxurl** option and the name of the database in the **/influxdb** option. All monitored resource values are pushed to the database at the end of a monitoring interval (1 minute). The stored data can easily be used for visualization inside monitoring dashboards (e.g. Grafana).

Figure 78. Resource Monitor Dashboard in Grafana

The following table shows the translation of the resource monitoring values into InfluxDB sensor data.

Resource Name	Sensor Name	Value Name	Tags
net.requests	net	requests	
sessions.active	sessions	active	
models.loaded	models	loaded	
db.read.rows	db.read	rows	
db.read.duration	db.read	duration	
db.write.rows	db.write	rows	

db.write.duration	db.write	duration	
license.free.REGNR	license	free	regnr=REGNR
license.used.REGNR	license	used	regnr=REGNR

For all data the server adds a tag named **host** with the servers host name. Additional tags can be specified by the command line option **/influxtags** as comma separated name value pair (e.g. */influxtags "tag1=value1,tag2=value2"*). To select the resources send to the database the two option **/influxinclude** and **/influxexclude** can be used with a regular expression that matches the resource names (e.g. */influxinclude "(models|db.read|db.write).*"*).

7.9. Server Command Line Options

Following list describes the options that can be added to the pure::variants server command line or configuration file.

See Section 7.4, “Location of the Server Configuration File” to find the configuration file.

All options can also be passed by environment variables with the prefix **PV_SERVER_** (e.g. *PV_SERVER_LOGLEVEL=1*). To enable options without argument the value of environment variable has to be **true** (e.g. *PV_SERVER_ENABLEWEB=true*).

All options are considered in the following order. First the server config file is read, afterwards the environment variables are checked and in the end the command line parameters are checked. That means if one option is defined in the config file and as an environment variable the value of the environment variable is used. If it is additonally defined as a command line parameter the value from the command line parameter is used.

Option	**Description**
-L, /loglevel [LEVEL]	Level for server logging (0-9).
-l, /logfile [FILE]	File for the server logging output.
-r, /rmlog	Remove the old log file on startup.
/maxlogfilesize	Maximum size of the log file (default 500M)
/maxlogfiles	Maximum number of saved log files (default 7)
/config [FILE]	Path to the server configuration file listing command line options one by line.
-p, /port [PORT]	Port for the SOAP server connection (default 80).
-t, /clienttimeout [SECONDS]	Timeout in seconds before killing dead client connections (default 900 = 15min).
-a, /address [ADDRESS]	Address to bind for the SOAP server connection (default 0.0.0.0). Accepts IP address as well as host name.
-i, /info	Print server information (XML) on startup.
-h, /help	Show the command help text.
/printinfo	Print server information (XML) and exit.
/enableweb	Enable the servers HTTP interface.
/webpwd [PASSWORD]	Password for accessing the HTTP interface.
/webbasepath [BASEPATH]	Base path used for accessing the HTTP interface.
/licenselog	Enable Logging for License Server
/licenseuserlog	Enable Logging for License Server with full user data
/licenseuserlist [PATH]	Location of the user license access control list.
/sslkeyfile [PATH]	X.509 certificate for encrypted communication.
/sslpassword [PASSWORD]	Password for the X.509 certificate if needed.

/enableresourcemonitor	Enable the resource monitor (default setting)
/disableresourcemonitor	Disable the resource monitor
/enablecmdresourcemonitor	Enable the resource monitor for commands (default off)
/influxurl [URL]	Url of InfluxDB to record resource monitor data
/influxdb [NAME]	Name of the database in InfluxDB
/influxorg [ORGANIZATION]	Name of the organization in InfluxDB
/influxuser [USER]	Name of InfluxDB user
/influxpass [PASSWORD]	Password for InfluxDB user
/influxtoken [TOKEN]	API access token for InfluxDB
/influxtags [TAG1=VALUE1,TAG2=VALUE2,…]	Tags to append for recording in InfluxDB
/influxignoresslerror	Ignore error during certificate validation for InfluxDB
/influxnorevoke	Disable InfluxDB certificate revocation check
/influxinclude [REGEX]	Regex to define values send to InfluxDB
/influxexclude [REGEX]	Regex to filter values send to InfluxDB

There are some command line options, which are available in Windows only.

Option	**Description**
/service	Start as Windows service.
/install	Install as Windows service.
/remove	Remove Windows service.
/servicename [NAME]	Name for service to install/remove
/servicedesc [DESCRIPTION]	Description for service to install/remove

7.9.1. Model Server Command Line Options

Following list describes the database model server specific options that can be added to the pure::variants license server command line or configuration file.

See Section 7.4, “Location of the Server Configuration File” to find the configuration file.

Option	**Description**
-P, /plugindir [PATH]	Semi-colon separated list of additional server plugin directories.
/prolog [PATH]	Location of the prolog interpreter executable.
-R, /plprog [PATH]	Location of the prolog resource database (prologrc).
-S, /xsltdir [PATH]	Location of the directory containing the server XSLT scripts.
/maxevaluators [NUMBER]	Maximal number of parallel running evaluators (default 25).
/license [PATH]	Location of the license file.
/logon [LOGON,LOGON,...]	Logon types to support (local, ldap, windows, openid).
/allowemptypassword	Allow users with an empty password to logon.
/systemuser [USERNAME]	User name of the user who is mapped to the internal system user
/aliasuser [USERNAME]	User name of the user who is allowed to use aliases
/disablehistory	Disables the model history.

/ldapurl [URL]	LDAP server URL (“ldap://server:port” or “ldaps://server:port”)
/ldapusersdn [DN]	LDAP users branch distinguished name (e.g. “cn=Users,dc=company,dc=com”)
/ldapuidattr [NAME]	LDAP username attribute of users (e.g. “uid”, or “cn”)
/ldapsysuser [DN]	LDAP user mapped to “system” user (e.g. “cn=pvadmin,cn=Users,dc=company,dc=com”)
/ldapbinduser [DN]	LDAP bind user
/ldapbindpass [PASSWORD]	LDAP bind user password
/openidregister	Register at Open ID provider
/openidunregister	Unregister from Open ID provider
/openidurl [URL]	Open ID provider URL (https://server:port/basepath)
/openiduser [USER]	Open ID user for registration
/openidpass [PASSWORD]	Open ID user password for registration
/openidignoresslerror	Ignore certificate validation errors for Open ID provider URL
/openidnorevoke	Disable Open ID provider certificate revocation check
/openidloginspec [SPEC]	Specifies the location from which the user login is taken. Default userinfo:sub
/apikey [APIKEY]	Key for accessing the HTTP API. Requires HTTP interface to be enabled.

Database Model Server Command Line Options

Following list describes the pure::variants database model server specific options that can be added to the server command line or configuration file.

See Section 7.4, “Location of the Server Configuration File” to find the configuration file.

Option	**Description**
/odbcdsn [NAME]	Name of the ODBC data source.
/odbcuid [USERNAME]	Name of the database user.
/odbcpwd [PASSWORD]	Password of the database user.
/odbctimeout [SECONDS]	Timeout for database operations (defaults to 2 minutes).
/odbcautoreconnect	Automatic reconnect to database after connection lost

7.10. Proxy Configuration

The pure::variants model server can be placed behind a reverse proxy. No configuration changes are required on the side of the model server.

The configuration of the reverse proxy for the pure::variants model server is similar to that for the pure::variants license server (see Section 4.12, “Proxy Configuration”). The only difference is that for the connection to the pure::variants model server a higher timeout needs to be configured, since long-running requests are used in the communication. Using the proxy standard settings (usually 60 seconds) can result in "HTTP 502 Bad Gateway" or "HTTP 504 Gateway Timeout" errors on client side. It is recommended to set the timeout to 50% of the configured client timeout (see Section 7.9, “Server Command Line Options”). So for the default client timeout of 900 seconds, a proxy timeout of 450 seconds should be configured.

The following code shows a minimal example for an Nginx reverse proxy configuration suitable for the pure::variants model server.

```
events {
  worker_connections 1024;
}

http {
  server {
    listen 80;
    server_name proxy;
    proxy_connect_timeout 450;
    proxy_send_timeout    450;
    proxy_read_timeout    450;
    send_timeout          450;

    location /pvlic/ {
      proxy_pass     http://backend:8081/;
      proxy_redirect http://backend:8081 /pvlic;
    }
  }
}
```

This configuration passes requests addressed to *http://proxy/pvlic/* to the backend server address *http://backend:8081/*. The server names *proxy* and *backend* only are examples and need to be replaced with your actual server names. The same applies to the ports and the location path in the example.

The following code shows the same example for Apache and IBM HTTP Server (just the proxy part).

```
ProxyPass        /pvlic/ http://backend:8081/ timeout=450
ProxyPassReverse /pvlic  http://backend:8081
```

8. pure::variants Connectors

8.1. Installation of pure::variants Connectors

Installing a connector into an existing pure::variants installation works the exact same way like installing the pure::variants Desktop Client into an exsiting Eclipse instance. You just have to make sure the depemding pure::variants connectors are already installed or they have to be installed together with the new connector. See the section called “Using update site”.

8.2. 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. Depending on the Capella version, a different Eclipse update site needs to be added. To do that, open **Window**->**Preferences**->**Install/Update**->**Available Software Sites** and add the update site.

- For Capella 5.x use https://download.eclipse.org/releases/2020-06/
- For Capella 6.x use https://download.eclipse.org/releases/2021-06/
- For Capella 7.x use https://download.eclipse.org/releases/2023-03/

8.3. 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 79, “A XML configuration for pvRestriction field.”).

Figure 79. 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 80, “A XML configuration for pvRestriction field's control type.”).

Figure 80. A XML configuration for pvRestriction field's control type.

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

8.4. 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.

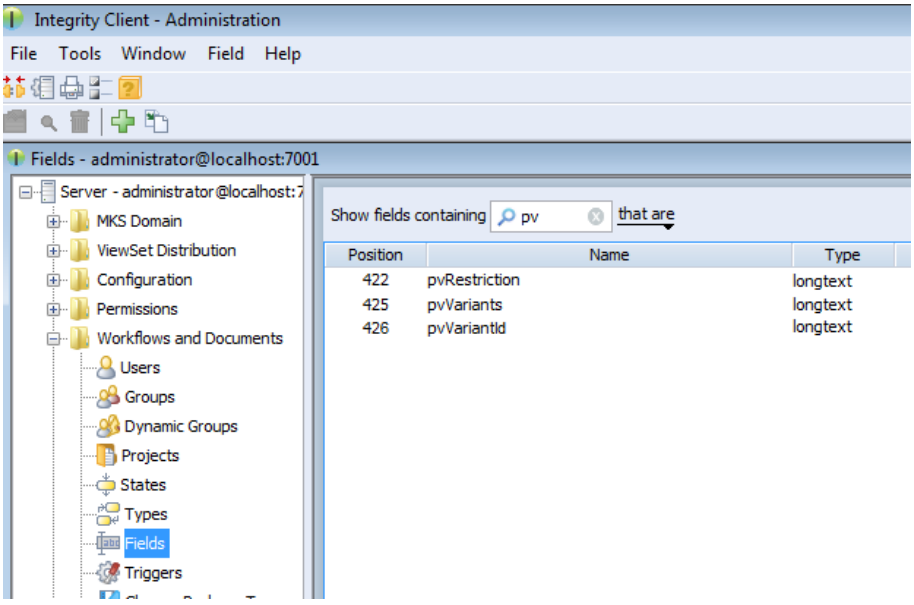
8.4.1. Add additional Fields for pure::variants

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

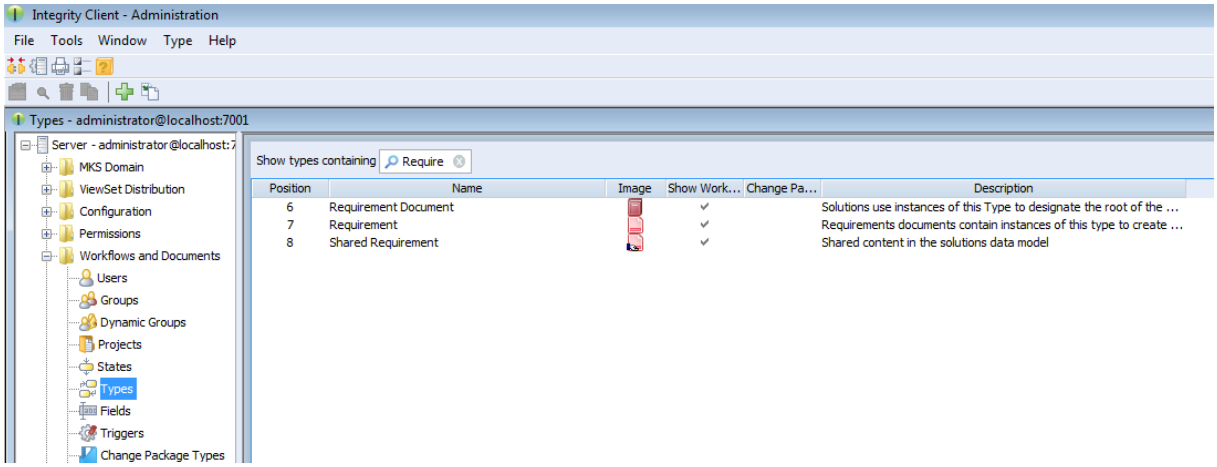
Field	Description
pvRestriction	This field is needed to store the restriction rule on a requirement. Set **longtext** as the type of the field, and ensure that the field is editable.
pvVariants	This field is needed to store the names of variants a requirement is part of. Set **longtext** as the type 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 81. Added fields



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

Figure 82. Solution Types

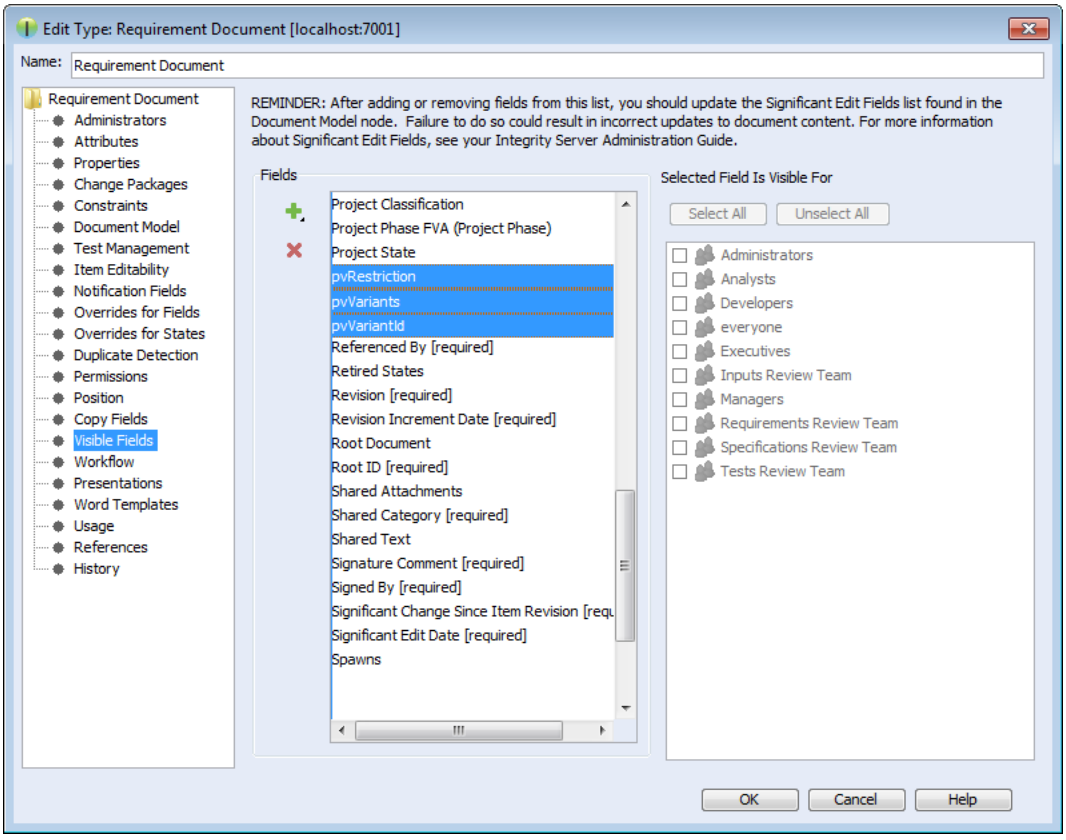


Add the new fields as **Visible Fields** to the following types.

Table 3. Types to add the fields for

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

Figure 83. Fields added to type Requirement Document



8.4.2. 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
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

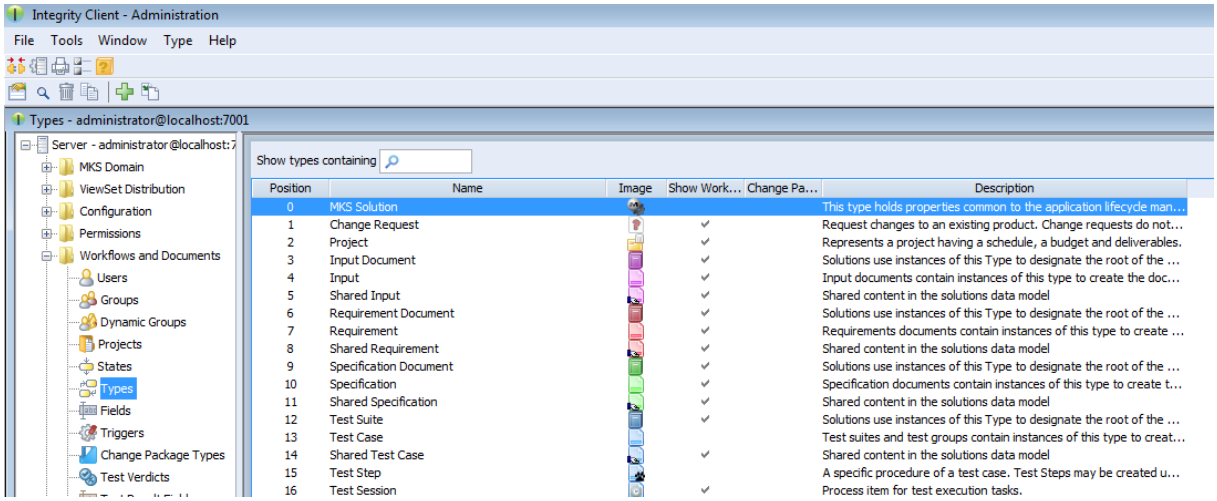
Setting	Config File Property	Solution Type Property
Solution Item	solutionType	
Document Type Field	fieldname.documenttype	PUREVARIANTS.TYPE.FIELD
Project Field	fieldname.documentproject	PUREVARIANTS.PROJECT.FIELD
Document Fields		PUREVARIANTS.VARIANT.FIELDS
Document Title Field	fieldname.documenttitle	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-6**. 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
```

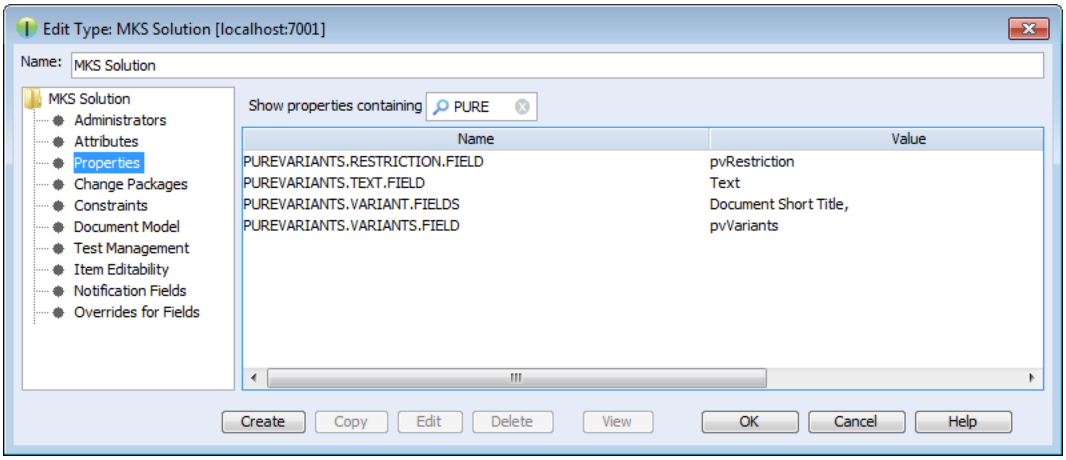
To change these settings on the solution item, open the Administration of PTC Integrity. Switch to **Workflows and Documents** -> **Types** and select the solution type (e.g. **MKS Solution**).

Figure 84. 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 85. Added Properties

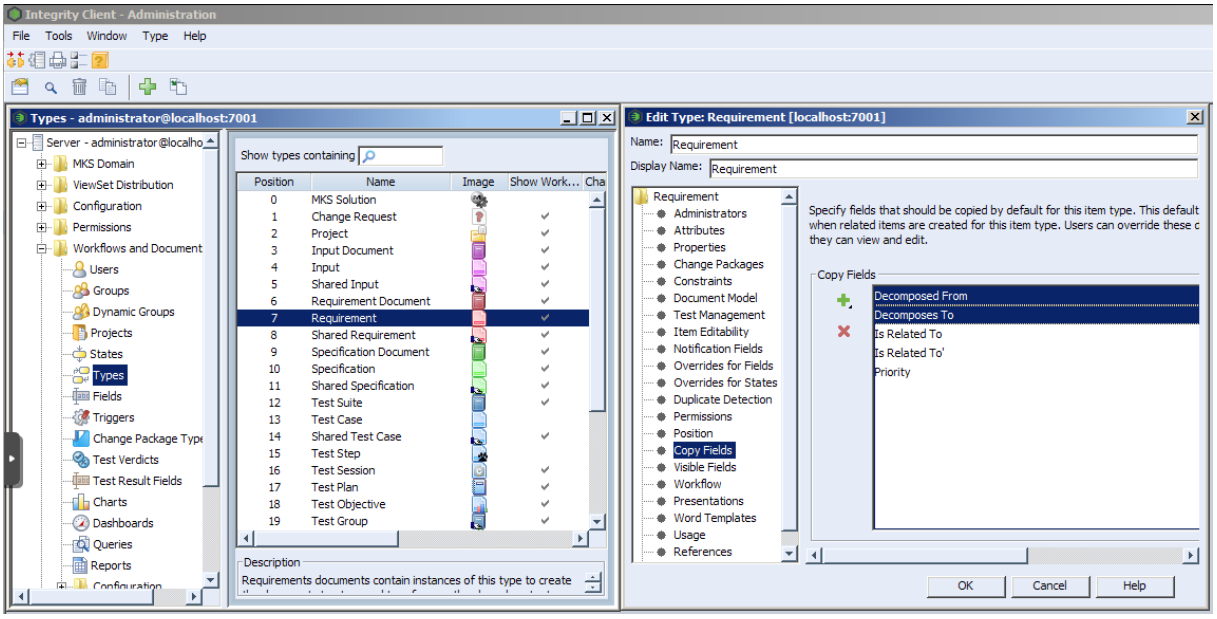


8.4.3. Change Fields Copied for Variant Creation

Several fields of the original document are copied while creating a document variant. This includes the fields mentioned in section Section 8.4.1, “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**.

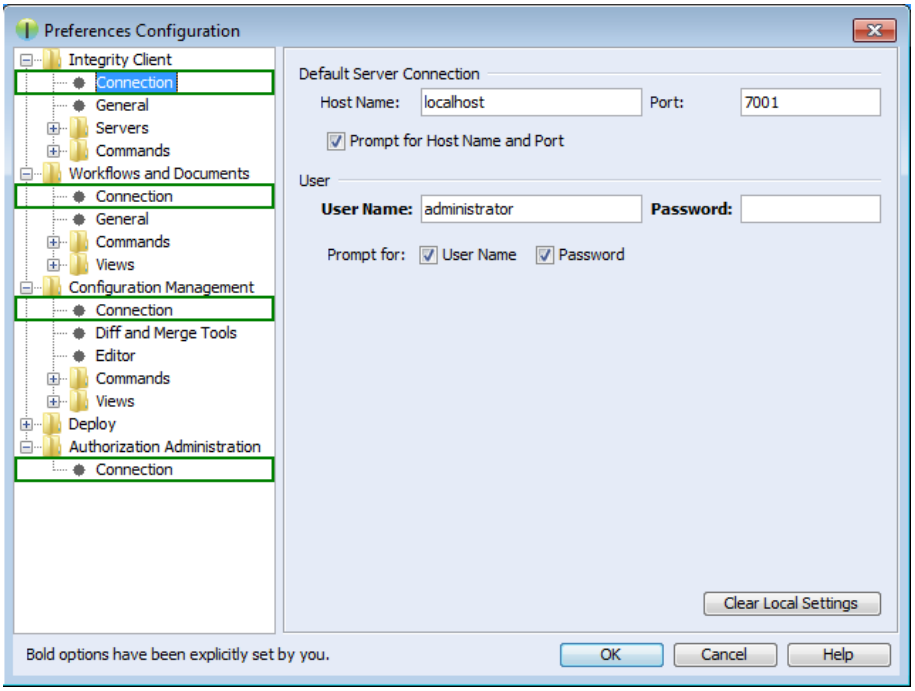
Figure 86. Decompose Fields added



8.4.4. 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 87. 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.

8.5. 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.

8.5.1. 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-sys\tems\pv_Enterprise_6.0\com.ps.consul.eclipse.ui.rhapsody.integration`).

8.5.2. 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 transformatin 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 trasformation will look for it in the default location.

8.6. Connector for Codebeamer

This chapter describes installation instructions specific to the Codebeamer connector.

8.6.1. Installation of pure::variants Desktop Client

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

8.6.2. 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 files are packed in a zip file **com.ps.consul.codebeamer.vel.jar-<*version*>.zip** indicating the compatible pure::variants version for identification.

2. The *'pure::variants Widget to Codebeamer'* that needs to be deployed on the Codebeamer server is packed in a zip archive **com.ps.consul.codebeamer.pvwidget-<*version*>.zip**.

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 Desktop Client side.

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

> **Note**
> The above steps are required only for Codebeamer versions up to 2.2.0.0. More details on how to enable the pure::variants widget in Codebeamer, starting from version 2.2.0.0, can be found in the following link: Enabling pure::variants Widget in Codebeamer.

8.6.3. 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/widget' 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
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"
    }
}
```

> **Note**
> The above steps are required only for Codebeamer versions up to 2.2.0.0. More details on how to enable the pure::variants widget in Codebeamer, starting from version 2.2.0.0, can be found in the following link: Enabling pure::variants Widget in Codebeamer.

8.6.4. 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"
    }
}
```

> **Note**
>
> The above steps are required only for Codebeamer versions up to 2.2.0.0. More details on how to enable the pure::variants widget in Codebeamer, starting from version 2.2.0.0, can be found in the following link: Enabling pure::variants Widget in Codebeamer.

8.6.5. Pre-defined settings for Web Integration in Codebeamer

The Web Integration for Codebeamer allows you to pre-define specific settings to ease up or limitate the setup for end-user.

Please see chapter Section 10.2, “Pre-defined settings for Web Integration” for detailed explanations, which settings are available and how they are configured.

To use these pre-defined settings for Codebeamer Web Integration, you need to write the configuration (in JSON format) into a file called *settings.json* and put this into the deployment directory.

This settings.json must be placed into the existing deployment directory of the Integration.

E.g. <codebeamer>/tomcat/webapps/pv-widget/

```
|-index.html
|-extension.json
|-settings.json
|-... (other files)
```

Configure Widget for SSO setup

The Widget for Codebeamer must be at least configured, if Single-Sign On (SSO) is used in the company's environment. If SSO is used, the pure::variants OIDC proxy has to be used (as described in section Section 8.6.8, “Configuration To Enable Open ID Connect (OIDC) Authentication”). Additionally, the widget must know this proxy's public URL, so the WIdget properly works in all use-cases. Therefore, please add the *settings.json* with the following content:

```
{
```

```
  "SSO_SERVER_URL: "https://codebeamer.server:9443/"
}
```

Please adapt the SSO server url accordingly to your environment setup.

8.6.6. Permissions

The communication of pure::variants Desktop Client and Widget to Codebeamer application uses the Codebeamer REST API. In order to use the REST API end points, the user needs to have *Rest / Remote API - Access* permissions.

To do this, make sure the user group that the users are assigned to in Codebeamer have this permission set via *System Admin->User Groups* menu.

8.6.7. 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'.

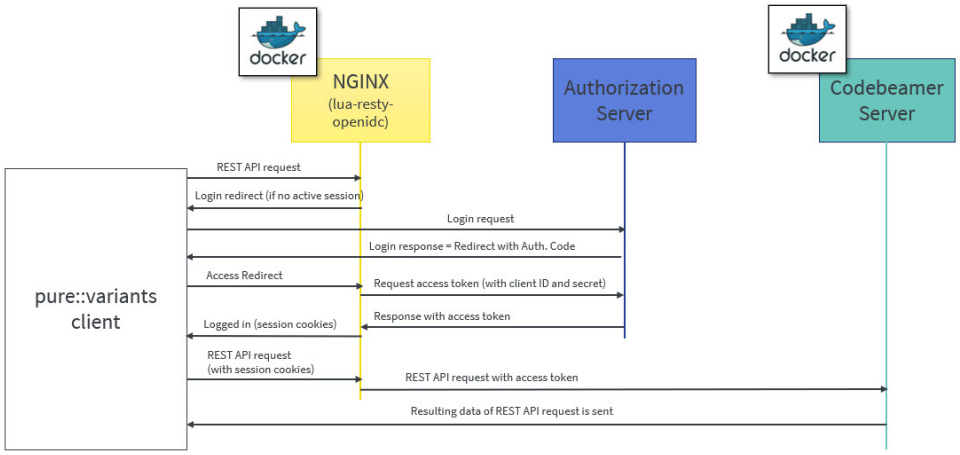
8.6.8. Configuration To Enable Open ID Connect (OIDC) Authentication

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 88. 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.

8.6.9. 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 Desktop 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
```

8.6.10. 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;"]
```

8.6.11. 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;
    client_max_body_size 16m;

    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}
        }
        if ngx.req.get_headers()["Authorization"] == nil or (not
 string.match(ngx.req.get_headers()["Authorization"], "Bearer")) then
          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)
        end
      }
```

```
   ##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;
    }
  }
}
```

8.6.12. 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.

8.7. pure::variants Connector for Siemens Polarion

This chapter describes how to setup the components in order to work properly together with Polarion

8.7.1. Installation of pure::variants Desktop Client

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

8.7.2. Installation of pure::variants server component for Polarion

The pure::variants enterprise ("pure::variants Windows Installer Package") contains the package **com.ps.consul.polarion.api-<*version*>.zip** which needs to be deployed to the Polarion server. Therefore please follow the following steps:

1. Unzip the archive **com.ps.consul.polarion.api-<*version*>.zip** which contains a folder

2. Place the extracted folder in the Polarion server at **<*polarion home*> /polarion/extensions/pure-systems/eclipse/plugins/**

3. Delete the cached Polarion configuration to ensure the pure::variants integration will be loaded properly on startup by deleting the folder **<*polarion home*> /data/workspace/.config**.

   #### Note

   *Please make sure that you only delete the **.config** folder*

4. Restart the Polarion service

The previous steps are only for installing the integration. To configure the the pure::variants plugin please follow the steps in Section 8.7.3, “Configuration of pure::variants server component for Polarion”

8.7.3. Configuration of pure::variants server component for Polarion

There are several steps necessary to prepare the Polarion server for pure::variants.

Configuration steps in Polarion

In order to show the in tool integration in the project's sidebar the pure::variants topic needs to be added in Polarion which can be done in the **Global Administration** or in the **Project Administration**

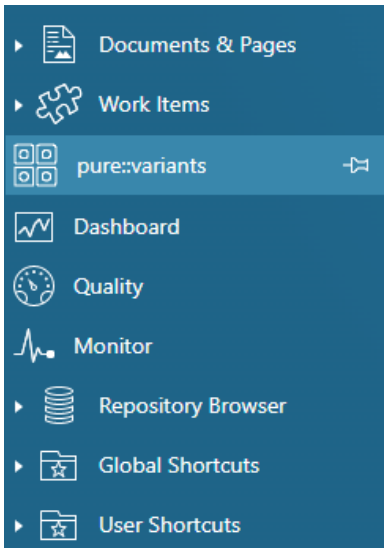
1. Navigate to the **Administration area** -> **Poartal** -> **Topics**
2. Create a new topics configuration or add the pure::variants topic to an existing one
3. Add **<topic id="purevariants" />** to the topic xml

   an example topic definition is given here:

```
<?xml version="1.0" encoding="UTF-8"?>
<topics xmlns="http://polarion.com/schema/Portal/Topics" xmlns:xsi="http://www.w3.org/2001/
XMLSchema-instance" xsi:schemaLocation="http://polarion.com/schema/Portal/Topics">
    <topic id="wiki"/>
    <topic id="workitems"/>
    <topic id="plans"/>
    <topic id="testruns"/>
    <topic id="collections"/>
    <topic id="purevariants"/>
    <topic id="baselines"/>
    <topic id="builds"/>
    <topic id="dashboard"/>
    <topic id="quality"/>
    <topic id="reports"/>
    <topic id="monitor"/>
    <topic id="repository_browser"/>
    <topic id="global_shortcuts"/>
    <topic id="project_shortcuts"/>
    <topic id="user_shortcuts"/>
</topics>
```

4. If all steps from Section 8.7.2, “Installation of pure::variants server component for Polarion” and the above mentioned ones were successful the pure::variants menu should be visible on the sidebar of the project with the configured topics.

Figure 89. Polarion menu on the project's sidebar including pure::variants.



Connection configuration of the pure::variants integration

The connection settings of the pure::variants in tool integration for Polarion can be preconfigured with proper settings in order to ease the process of configuration for the user. Therefore a simple javascript file with the

name **pv.settings.js** is placed in the subfolder **webapp** of the Polarion server extension. The configuration and possibilities of the javascript file are documented here: Section 10.2, “Pre-defined settings for Web Integration”. The default configuration looks lik this:

```
            pvWidgetConnectionSettings = {
  // PV_HUB: Defines either 'webhub' or 'desktophub' as tool to connect with
  "PV_HUB": undefined,
  // PV_HUB_EDITABLE: true or false. Defines if the user is able to change the connection
  "PV_HUB_EDITABLE": true,
  // PV_HUB_URL: provide the url if webhub is defined at PV_HUB
  "PV_HUB_URL": undefined,
  // PV_HUB_URL_EDITABLE: true or false. Defines if the webhub url is modifieable
  "PV_HUB_URL_EDITABLE": true
 };
```

Note

Please make sure not to modify the variable name **pvWidgetConnectionSettings** during the modification of the settings

8.7.4. Preparation of the Polarion project to store variability information

Adding general settings information to Polarion

For managing varaiblity information in Polarion assets pure::variants needs to know where restrictions are stored and which characters are starting a calculation. These settings can be stored either **globally**, on **project level** or even on **document level**. Therefore the settings priority follows this rule: **document settings > project settings > global settings**

The settings are stored in the key **pvSettings** in json format e.g.:

```
            pvSettings={"beginMarker":"[","endMarker":"]","escapeMarker":"%",
"performPartialTextSubstitution":true,"pvRestrictionFieldName":"pvRestriction"}
```

- To store the settings globally navigate to the **Global Administration** and add the pvSettings property to the **Configuration Properties**
- To store the settings for a specific project navigate to the **Project Administration** and add the pvSettings property to the **Configuration Properties**
- To store the settings in a document of a project you need to add a custom field to the document. Therefore please navigate to the **Project Administration** -> **Document & Pages** -> **Document Custom Fields** and add a custom field to the document typs you want to support. The custom field's ID must be **pvSettings** and the type of the custom field must be **String (single line plain text)**

Preparing restrictions for work items

In order to store the restriction of a work item. The work item needs to have a custom field with the type **String (single line plain text)** the ID of the custom field must match the ID which is stored in the pvSettings object from above. Our default ID for the custom field is **pvRestriction**.

To add a custom field to a work item type please navigate to the **Project Settings** -> **Work Items** -> **Custom Fields**. There you need to select the work item type or all types and add a new field with the limitations mentioned.

Preparing the enumeration transformation

During the enumeration transformation of pure::variants the work items will be tagged with their respective variant in which they are contained. To store that information another custom field is necessary. The ID of the custom field must match the ID which is provided in the transformation configuration of the variant which is transformed

via pure::variants. Our default ID for the custom field is **pvVariants**. This custom field must be of type **Rich Text (multi-line)**.

The procedure to add a custom field to a work item is described in the section called “Preparing restrictions for work items”

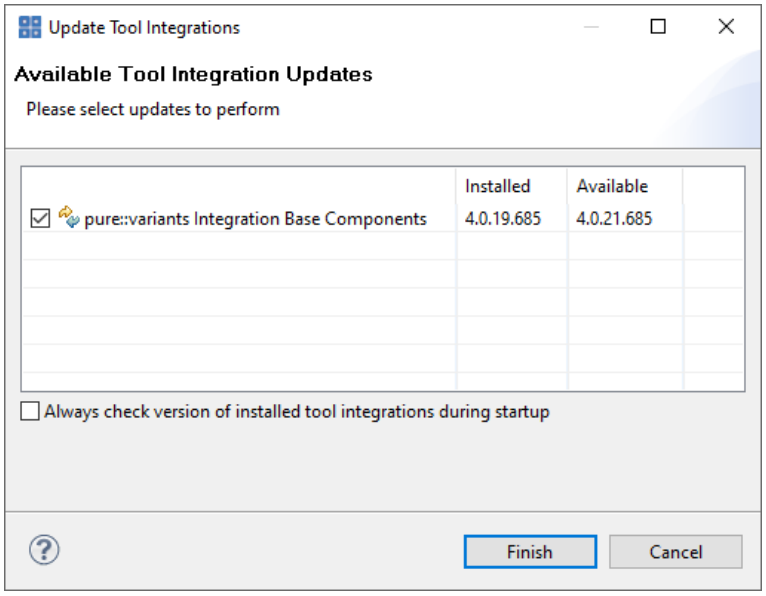
9. pure::variants Tool Integrations

9.1. 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 connector. 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 follwing menu entry in the pure::variants **Help** menu. Go to **Help**->**pure::variants** the item **Tool Integration Updates**.

Figure 90. 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.

9.1.1. Install pure::variants Tool Integrations in silent mode

The pure::variants Tool Integration installers have a silent mode. This mode installs the pure::variants Tool Integrations without user interaction. The different Tool Integration installers can be found in this path <PV_INSTALL_DIR>\<ECLIPSE_DIR>\plugins.

To do this, call the installer with command line option */S*. Note that the target directory must not contain any quotation marks. Here's the generic example commandline for silent install of a Tool Integration:

```
<PV_INSTALL_DIR>\<ECLIPSE_DIR>\plugins\<integration_plugin_folder>\<tool_name>AddIn\setup.exe"
```

```
            /S /D=<PV_INSTALL_DIR>\com.ps.consul.eclipse.ui.<tool_name>.integration
```

For example, to silently install the Office Integration the following commandline can be performed:

```
"C:\Program Files\pure-systems\pv_Enterprise_6.0\eclipse\plugins
\com.ps.consul.eclipse.ui.ms.office.integration_5.x.x.xxx\msOfficeAddIn\setup.exe"
            /S /D=C:\Program Files\pure-systems
\pv_Enterprise_6.0\com.ps.consul.eclipse.ui.ms.office.integration
```

9.1.2. Update pure::variants Tool Integrations in silent mode

It is also possible to run the update in the 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.

9.1.3. pure::variants Desktop Hub

If the pure::variants Desktop Client was installed with the installer the pure::variants Desktop Hub is already installed along with the pure::variants Desktop 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 Desktop Client.

Please see Section 9.1, “Install pure::variants Tool Integrations” for installing the **pure::variants Integration Base Components**.

9.1.4. pure::variants Integration for Doors

Please see Section 9.1, “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 adaptions of the pure::variants Integration menu DXL scripts.

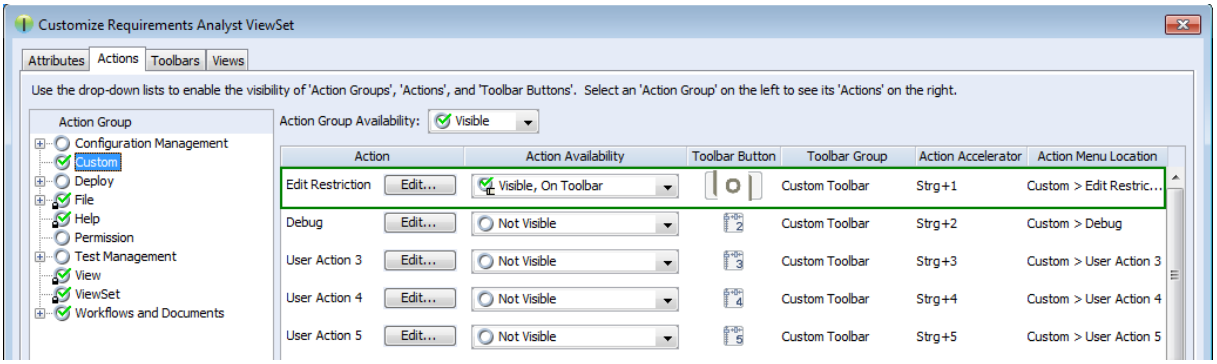
Now the Integration should be available in Doors. To verify if the installation was successfull, open a Doors module and select from the menu **pure::variants** the item **Open pure::variants Integration**. If the pure::variants Integration window opens, the Integration was installed correctly.

9.1.5. pure::variants Integration for PTC Integrity

Please see Section 9.1, “Install pure::variants Tool Integrations” for installing the integration executable.

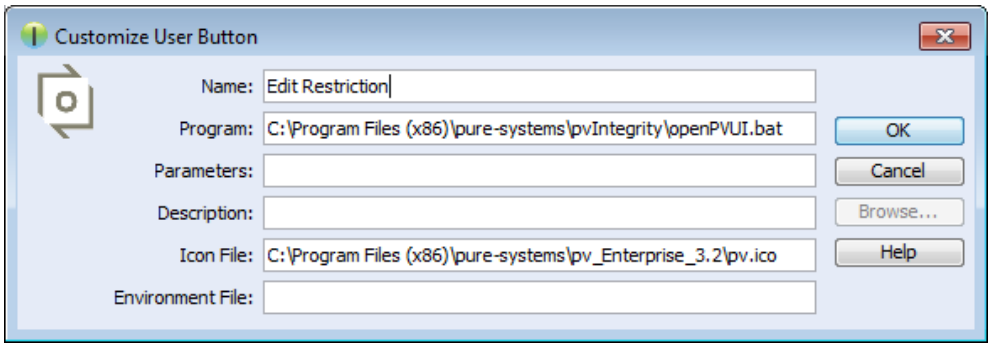
For starting the integration within PTC Integrity some additional steps have to be performed. Start the PTC Integrity client application. Open menu **ViewSet** -> **Customize** and switch to page **Actions**. Click on action group **Custom** to view the custom user actions.

Figure 91. 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 92. Custom Button "Edit Restriction"



9.1.6. pure::variants Integration for IBM Rational Rhapsody

Please see Section 9.1, “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**.

9.1.7. pure::variants Integration for Enterprise Architect

Please see Section 9.1, “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**).

9.1.8. pure::variants Integration for Microsoft Office

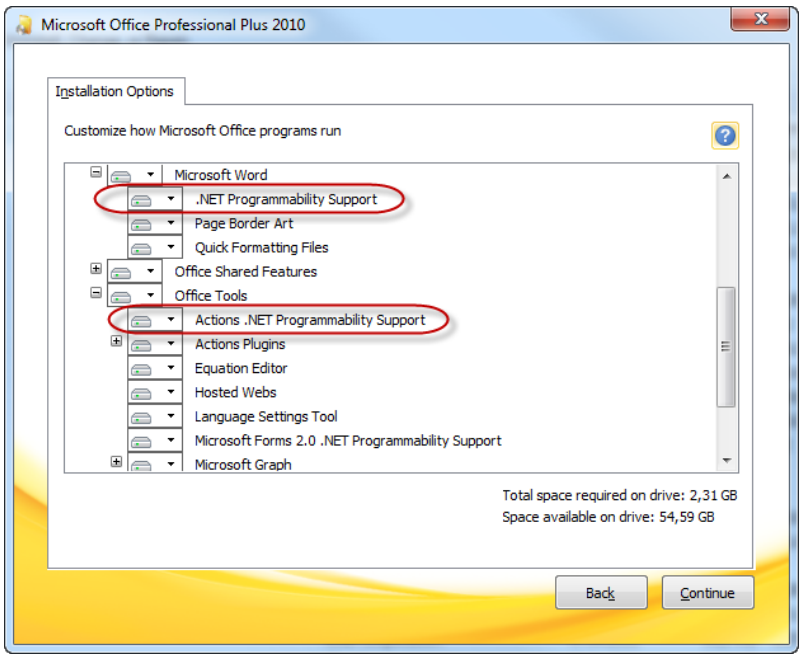
The Integration will not work if the following features are not installed with Microsoft Office:

- 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 93, “Adding Missing Features to the Office Installation” to your Office Installation.
5. Press **Continue** and close the Dialog.

Figure 93. Adding Missing Features to the Office Installation



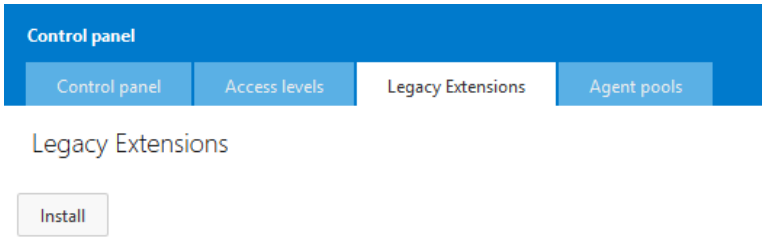
Now the installation of the pure::variants Integration for Microsoft Office should run successfully.

9.1.9. 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 94. Install pure::variants TFS Integration



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

9.1.10. 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-6/pv.properties
```

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

```
%APPDATA%/pure-variants-6/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
http.proxyHost	The hostname, or address, of the proxy server
http.proxyPort	The port number of the proxy server
https.proxyHost	The hostname, or address, of the proxy server in case HTTPS is used
https.proxyPort	The port number of the proxy server in case HTTPS is used
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 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 an 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.

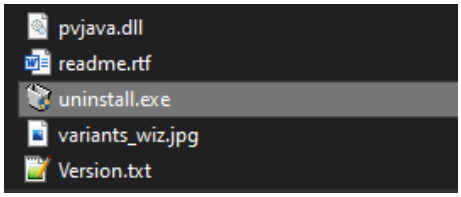
9.2. Update pure::variants Tool Integrations

To update an pure::variants tool integration the same mechanism as for installing a pure::variants tool integration is used. Please consult section Section 9.1, “Install pure::variants Tool Integrations” for a detailed description.

9.3. Uninstall pure::variants Tool Integrations

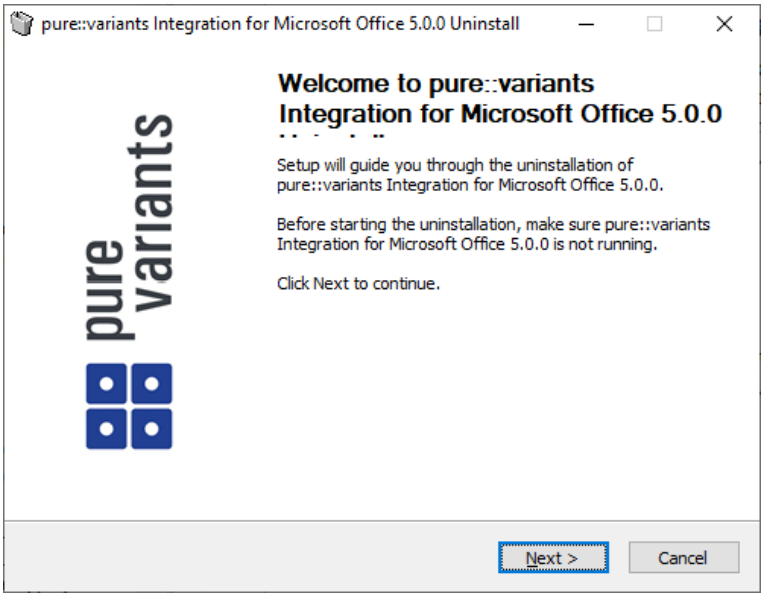
The uninstaller for the pure::variants integration can be started in two different ways. The first one is to go to the Windows *Add or remove programs* application and search for the pure::variants integration and start the uninstaller by using the *Uninstall* action. The uninstaller requires Administrator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

Figure 95. pure::variants Integration Uninstaller



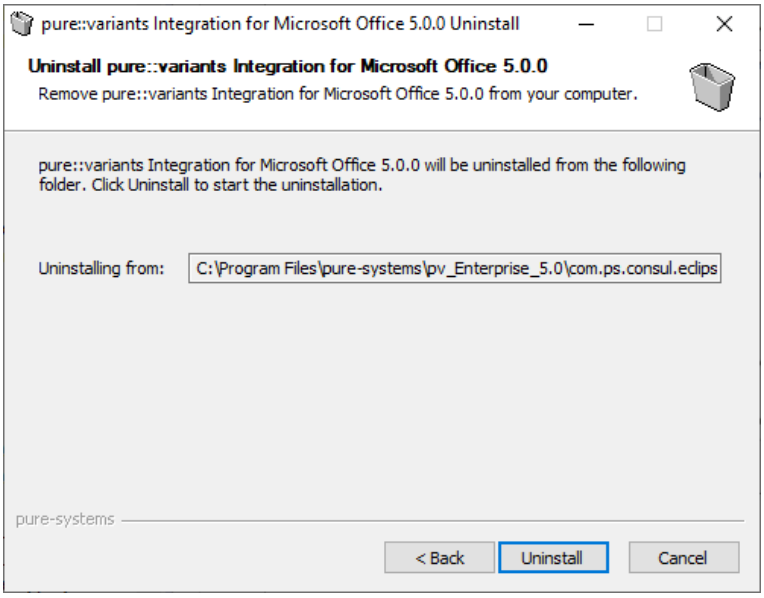
The second possibility is to navigate to the pure::variants Desktop Client installation folder and start the uninstaller by double clicking it.

Figure 96. pure::variants Integration Uninstaller



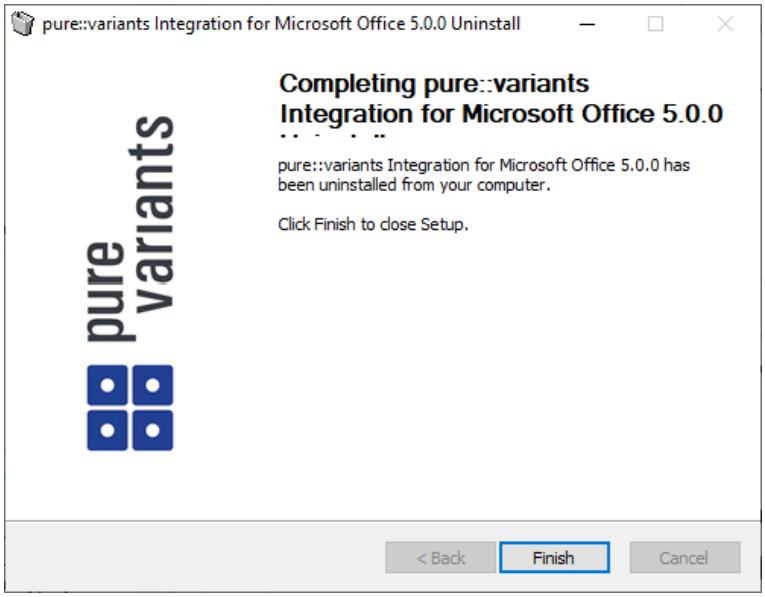
Click *Next*.

Figure 97. Uninstall from



Click *Uninstall* to start the uninstall process.

Figure 98. Completing Uninstall



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

9.4. Basic Setup of pure::variants Tool Integrations

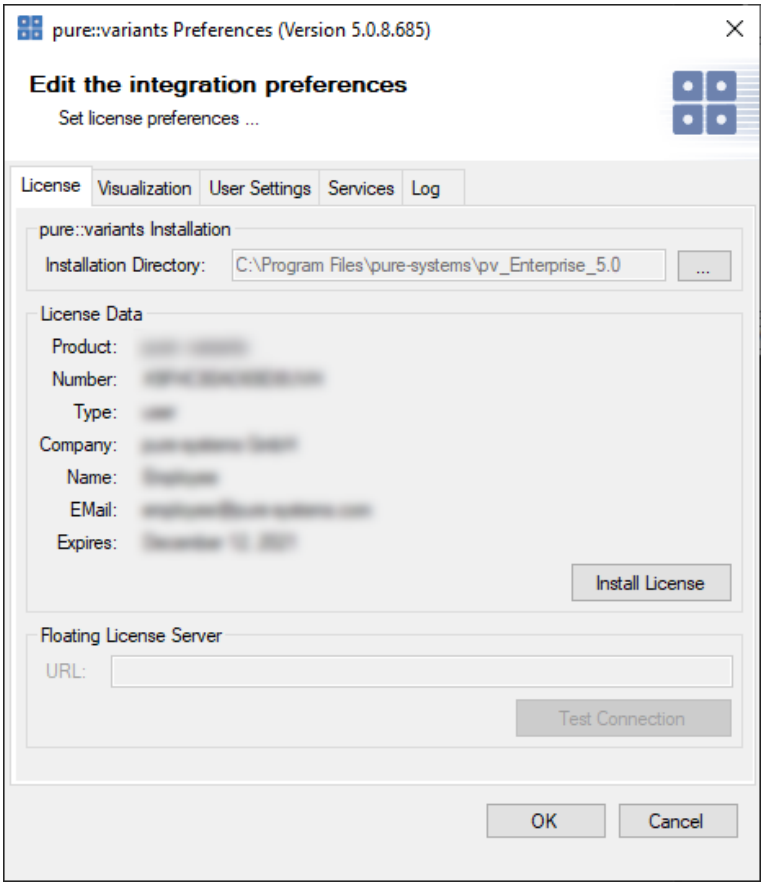
When you first use the Desktop Hub after installation, it is necessary to check whether the license preferences are correct. To this end, open the preferences dialog via the button in the Desktop Hub window or by selecting **Hub Configuration** from the pure::variants tray menu.

A dialog opens that shows the path to your pure::variants installation and your license information (see Figure 99, “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 99. Preferences Dialog



9.4.1. 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 9.1.4, “pure::variants Integration for Doors”
- pure::variants Integration for Microsoft Office
- Section 9.1.7, “pure::variants Integration for Enterprise Architect”
- Section 9.1.9, “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-6/pv.properties
```

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

```
%APPDATA%/pure-variants-6/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
http.proxyHost	The hostname, or address, of the proxy server
http.proxyPort	The port number of the proxy server
https.proxyHost	The hostname, or address, of the proxy server in case HTTPS is used
https.proxyPort	The port number of the proxy server in case HTTPS is used
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-6/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-6/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-6/selfsigned.jks
# the password to the keystore
keystorePassword=password
```

10. pure::variants Web Integration

10.1. 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)

10.1.1. Requirements for pure::variants Integration Deployment

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

- Extenstion 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 Envritonment (JRE) or Java Development Kit (JDK) version 1.6 or later.

10.1.2. 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 indi-

vidual 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.

10.1.3. 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 your Web Application Server (like Tomcat or WebSphere Liberty. Replace the **pvwidget.war** in the **server.config.dir/apps** directory. **Start** the web server again.

If you have used the pre-defined settings, as described in the section Section 10.1.8, “Pre-defined settings for Web Integration in DOORS NG”, please re-add the settings.json file accordingly, as done before.

Once finished, please start your Web Application Server again.

10.1.4. 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.

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

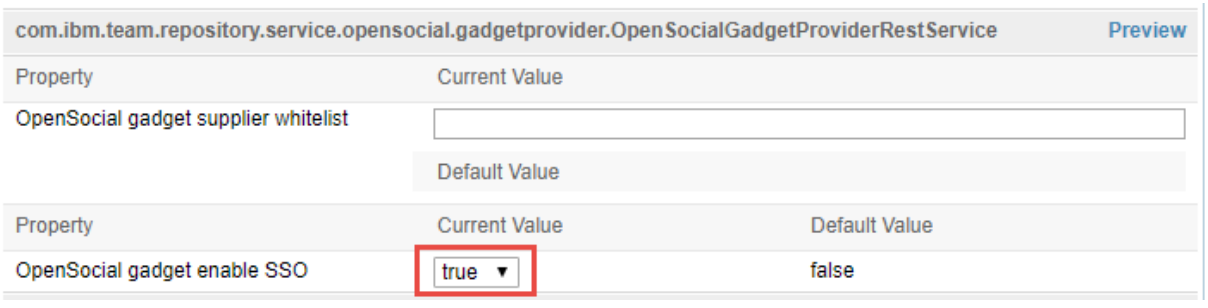
In order to visualize the variability, folloing settings need to be configured on the Jazz side.

JTS Advanced Settings

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

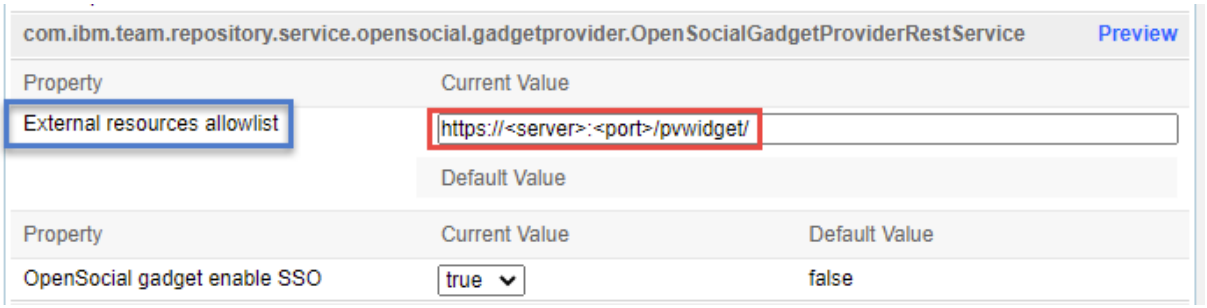
Scroll down to **OpenSocial gadget enable SSO** and set its value to **true** as shown in figure Figure 100, “OpenSocial gadget enable SSO Setting”.

Figure 100. OpenSocial gadget enable SSO Setting



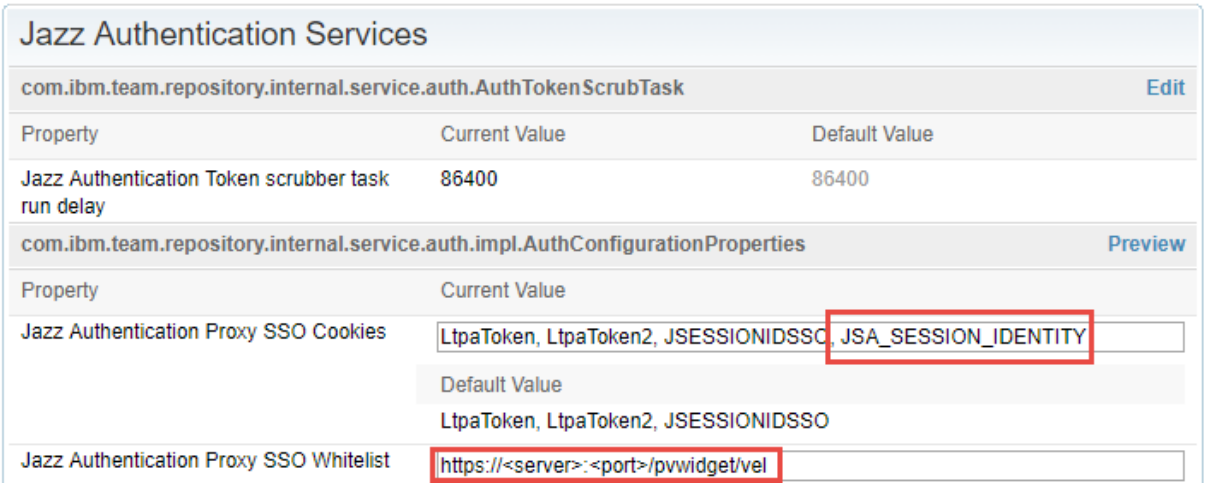
For the versions ELM 7.0.2 iFix004, ELM 7.0.1 iFix009, CLM 6.0.6.1 iFix018, CLM 6.0.6 iFix022 and onwards, an extra configuration is required. In **External resources allowlist**, please enter the URL of the widget. For example if pure::variants Integration for DOORS NG is accessible at https://jazz.server.net:8443/pvwidget/catalog.xml then enter https://jazz.server.net:8443/pvwidget/ in the allowlist as shown in the figure Figure 101, “External resources allowlist”.

Figure 101. External resources allowlist



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 102, “Jazz Authentication Proxy SSO Settings”.

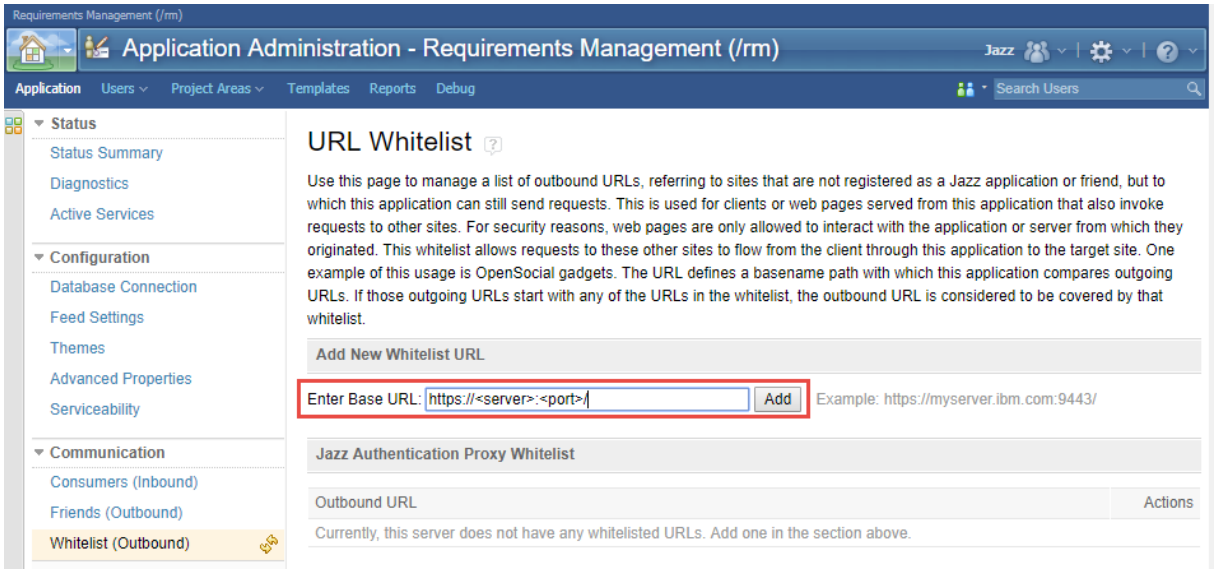
Figure 102. Jazz Authentication Proxy SSO Settings



Jazz RM Whitelist

As shown in Figure 103, “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 103. 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**.

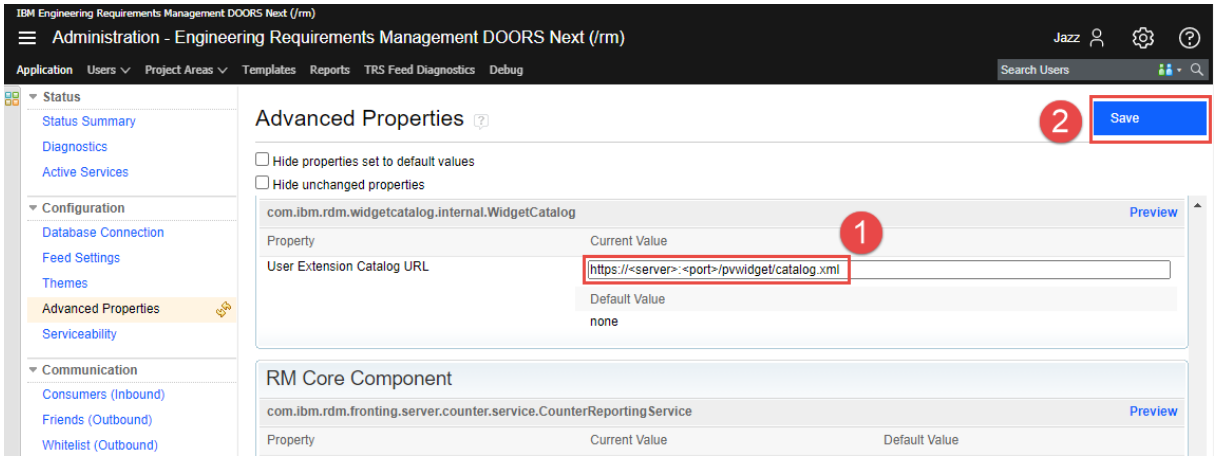
User Extension Catalog URL

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 104, “RM User Catalog Configuration”.

Figure 104. RM User Catalog Configuration



Considering that the context path of the pure::variants Integration is **pvwidget**, enter the Catalog URL as `https://<server>:<port>/pvwidget/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 accessbile 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>
```

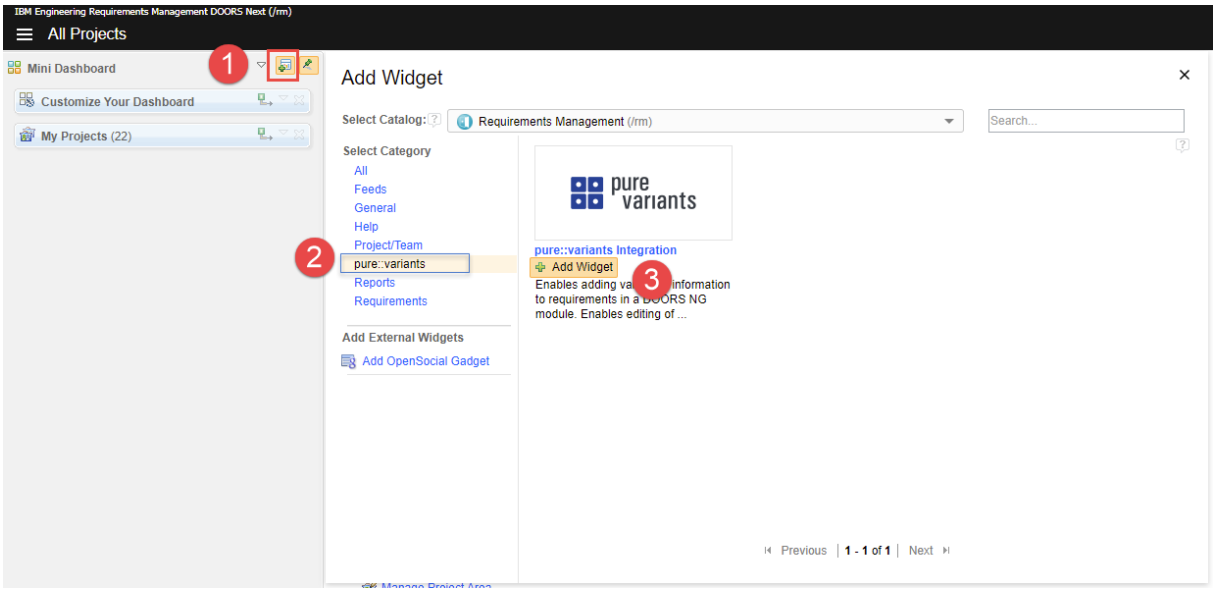
Note

Please note that the starting URL in the ju:gadeget, ju:icon, ju:preview and ju:thumbnail depends on the deployement 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.

10.1.6. 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 105, “Adding Integration inside Mini Dashboard”.

Figure 105. 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 attributation for Cookies. For this, please see the optional step in our Tomcat resp. WebSphere configuration guide.

10.1.7. 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}
```

10.1.8. Pre-defined settings for Web Integration in DOORS NG

The Web Integration for DOORS NG allows you to pre-define specific settings to simplify or limit the setup for end-user.

Please see chapter Section 10.2, “Pre-defined settings for Web Integration” for detailed explanations, which settings are available and how they are configured.

To use these pre-defined settings for DOORS NG Integration, you need to write the configuration (in JSON format) into a file called *settings.json* and put this into the deployment directory.

If your Web Application Server (like Tomcat or WebSphere Libterty) extracts the war-archive automatically, please add the settings.json file into the war-archive (called "com.ps.consul.web.ui.doorsng.*.war"

If your Web Application Server (like Tomcat or WebSphere Libterty) does not extract the war-archive automatically and you extract it manually, please add the *settings.json* into the deployment directory of the Integration.

Finally, the deployment directory should have located the *settings.json* next to the *pvscl.xml*, as follows:

```
|-META-INF
|-WEB-INF
|-pvscl.xml
|-settings.json
|-... (other files)
```

10.2. Pre-defined settings for Web Integration

As administrator, one can pre-define the settings for the end-user, e.g. to connect to the WebHub immediately. Thus, the user does not need to be aware of the connection configuration and can use the Widget out-of-the-box.

Therefore, a ***JSON*** structure must be created as described in the following for WebHub and DesktopHub, respectively, and deployed:

```
{
     "PV_HUB": "webhub",
     "PV_HUB_URL": "https://webhub.server/pv",
     "PV_HUB_EDITABLE": true,
     "PV_HUB_URL_EDITABLE": true
}
```

```
{
     "PV_HUB": "desktophub",
     "PV_HUB_PORT": "4773",
     "PV_HUB_EDITABLE": true
}
```

The *PV_HUB* property defines the hub to be used, which can be either the WebHub (*webhub*) or DesktopHub (*desktophub*).

The *PV_HUB_PORT* property defines the port number for the DesktopHub. It can be configured as either a string or a number

The *PV_HUB_URL* property defines the URL for the WebHub.

The *PV_HUB_EDITABLE* property defines, if the hub should be editable by the user. If it is undefined, the hub is editable by the user in the widget's settings.

The *PV_HUB_URL_EDITABLE* property defines, if the webhub URL should be editable by the user. If it is undefined, the webhub URL is editable by the user in the widget's settings.

Note

Regarding the deployment of the JSON structure, see the previous sections.

10.3. Friend Setup

You need to add **pure::variants Web Components** as a **Friend** to **Jazz Team Server (JTS)** in order to enable JTS to access the pure::variants web components data. Jazz Official Documentation can be visited for details.

Note

After finishing the pure::variants web components setup and before starting the friend setup in JTS, it is a must that the web server hosting the pure::variants web components must be restarted.

10.3.1. SSO mode

Procedure

1. Browse to the **JTS Admin page**. Then click **Manage Server** and then click **Friends (Outbound)**. This page can also be directly accessed by pointing your web browser to **https://[jazz-server-FQDN]:9443/jts/admin#action=com.ibm.team.repository.admin.friends**.

2. On the Friends Page, in the **Friends List** section, click **Add**.

3. In the **Add Friend** dialog, provide the following information.

   - Provide the URI for root services of the pure::variants Web Components in this form: **https://[pv-server-FQDN]:[port]/pv/rootservices**.

   - Enter a name to identify the pure::variants web components e.g. VM.

   - Click Next to continue.

4. As both jazz and pure::variants web components are configured in SSO mode that's why this dialog does not ask for any credentials. Click **Create Friend** and click **Finish** to close the wizard.

Note

On step 4 of the above procedure, if you are asked to enter OAuth credentials then it is a problem. It means that JTS is unable to recognize that the pure::variants web components is also SSO enabled and registered with the same Jazz Authorization Server as JTS. In this case you have to close the friend setup dialog and wait for 10 to 15 minutes and start the friend setup again. If the problem persists then you have to restart the jazz server and then do the friend setup again.