# pure::variants Setup Guide

pure-systems GmbH

Version 5.0.11.221 for pure::variants 5.0

Copyright © 2003-2022 pure-systems GmbH

2022

## Table of Contents



## 1. Introduction

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

components in certain deployment scenarios. In the following sections each of the orange boxes are explained from the perspective of administration and setup.

**Figure 1. The Big Picture**


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

# 2. System Requirements

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

## 2.1. Eclipse based Client

- Operating System
  - Windows 7, 8, 10
  - Windows Server 2003, 2008, 2012, 2016, 2019
  - Linux 64 Bit with X11 Window System installed
  - Mac OS
- Software
  - Oracle Java SE or OpenJDK

    Supported Java versions:
    - Java 8, 9, 10, 11, 12, 13, 14, 15, 16 ,17
  - The Java compatibility is tested with the official Java Standard Edition provided by Oracle (https://www.java.com/en/download/) and the OpenJDK provided by Oracle (https://jdk.java.net/archive/).
  - Eclipse 3.8.0 – 4.23
- Memory
  - Min: 2 GB
  - Recommended: 8 GB
- CPU
  - Min: Dual Core CPU
  - Recommended: 4 CPU cores

- HDD
  - Min: 10 GB free disk space

# 3. pure::variants Client

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

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

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

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

## 3.1. Install pure::variants Client

### 3.1.1. Install with pure::variants Installer

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

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

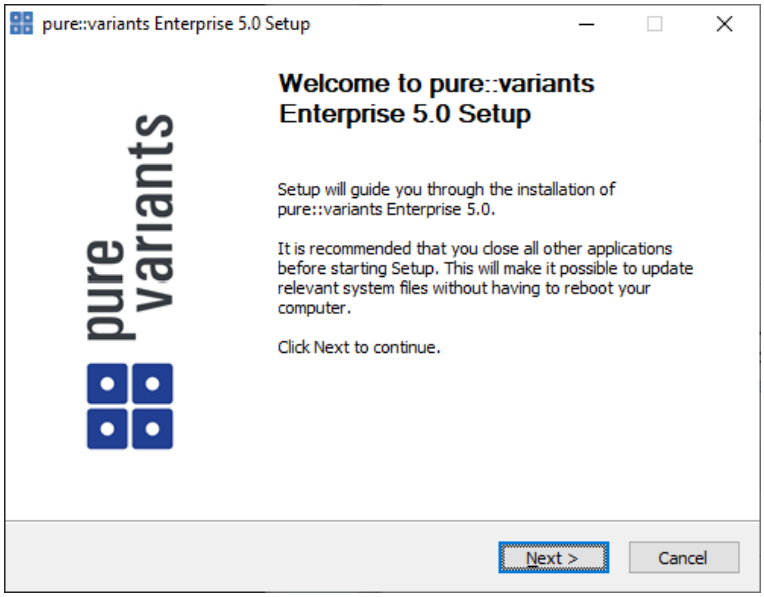
- pure::variants client license
  - If a floating license is used additionally the license server URL is needed to be able to connect and obtain a license form the license server.
- pure::variants client installer or pure::variants update site
- supported 64-Bit Java version installed

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

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

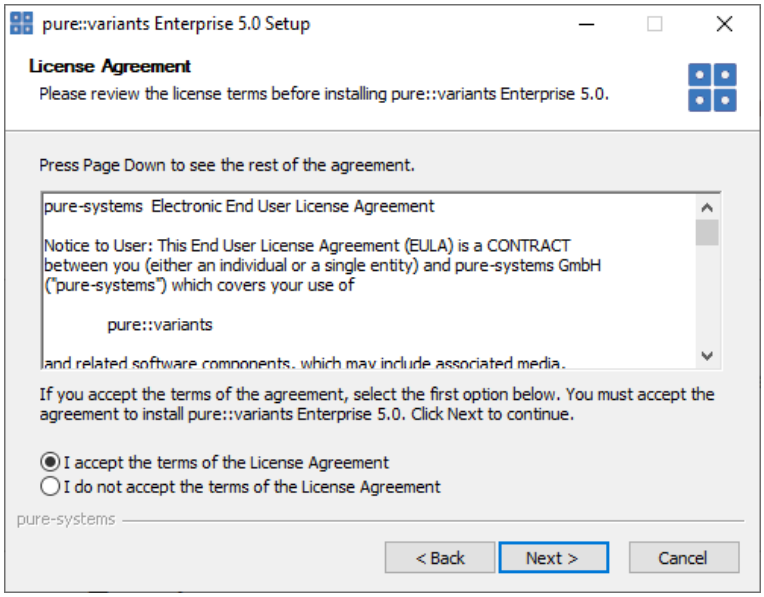
All pure::variants extensions available for the account are automatically included in the Windows Installer download. However, some may not be enabled by default in Installer. Make sure to select the desired extensions during the installation process. Later updates to the extension selection can be done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

**Figure 2. pure::variants Client Installer**



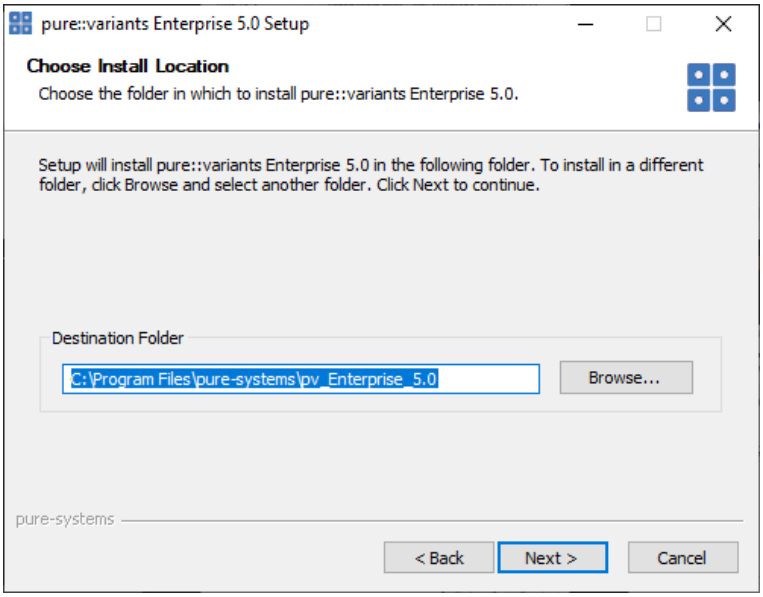

Click *Next*.

**Figure 3. Setup pure::variants Client License**



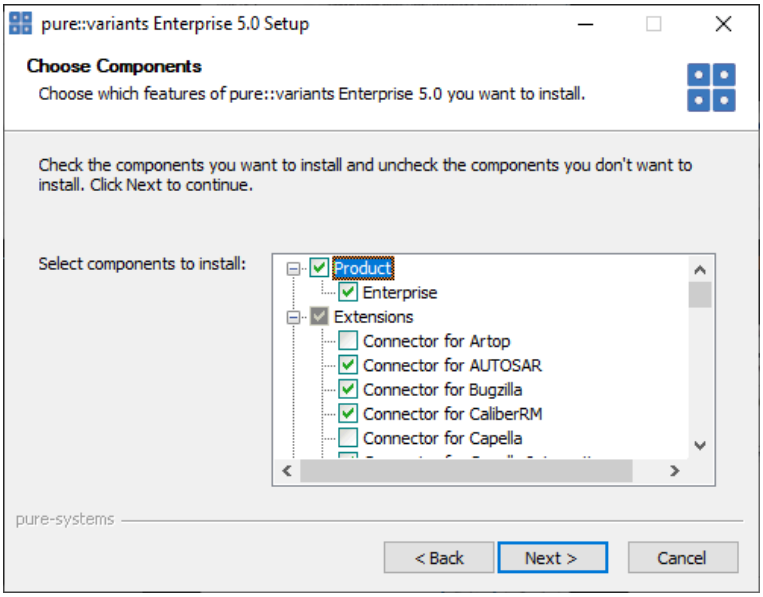

Read the license agreement and after accepting it click *Next*.

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



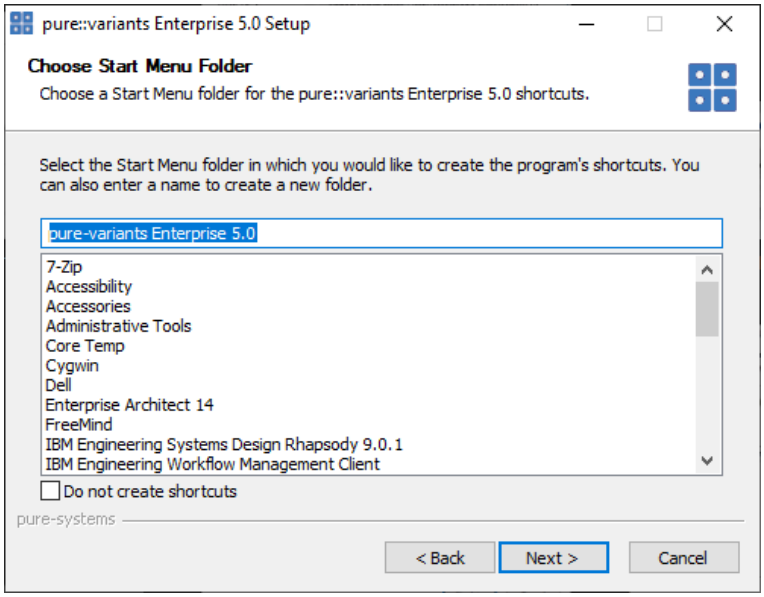

Select the folder where to install the pure::variants client files. Click *Next*.

**Figure 5. pure::variants Client Feature Selection**




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

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

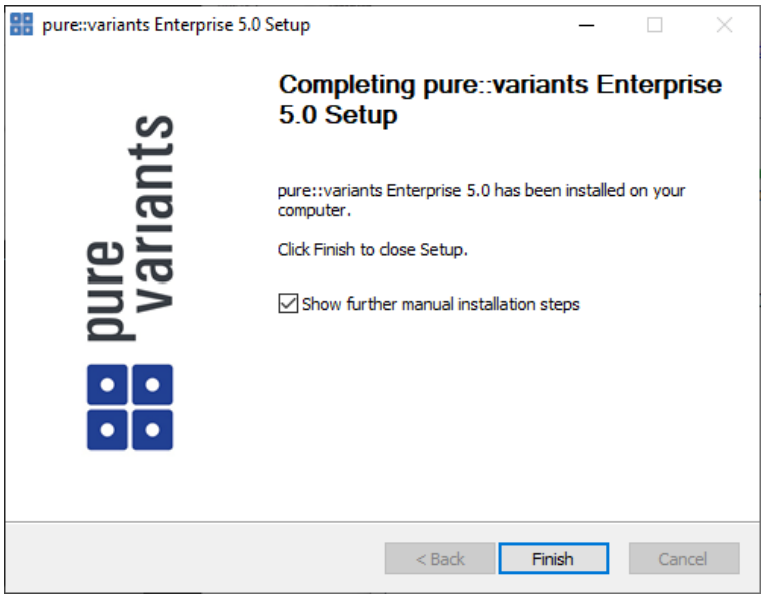



Enter the name for the Windows start menu entry, or disable the creation of the start menu entry. Click *Next*

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

Click *Install* to start the installation process.

**Figure 7. Setup pure::variants Client Finish Page**




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

## pure::variants Enterprise Installer Command Line Options

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

**Table 1. pure::variants Installer Command Line Options**

| Option | Description |
|---|---|
| /S | Run the installation in silent mode. No installation dialog is opened.<br><br>Automatically installs the default selected software packages, or all if used together with option /ALL. |
| /UPDATE /S | To update an exsiting installation in silent mode. Same as silent installtion, no dialog is opened. |
| start "" /WAIT "Setup.exe" /UPDATE /S | To update an exsiting installation in silent mode. Same as silent installtion, 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.<br><br>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.<br><br>This directory must contain the file eclipsec.exe.<br><br>Example: /ECLIPSE="C:\Program Files\Eclipse 3.8\eclipse" |
| /D | Path to the directory where to install the pure::variants stand-alone application.<br><br>**Must be the last option on the command line and must not contain any quotes, even if the path contains spaces.**<br><br>Example: /D=C:\Program Files\pure-variants |

Example commandline with JAVA path:

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

## Install pure::variants in silent mode

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

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

## Update pure::variants in silent mode

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

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

### 3.1.2. Install into an existing Eclipse

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

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

#### Installation Requirements

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

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

#### Using pure::variants Installer

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

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

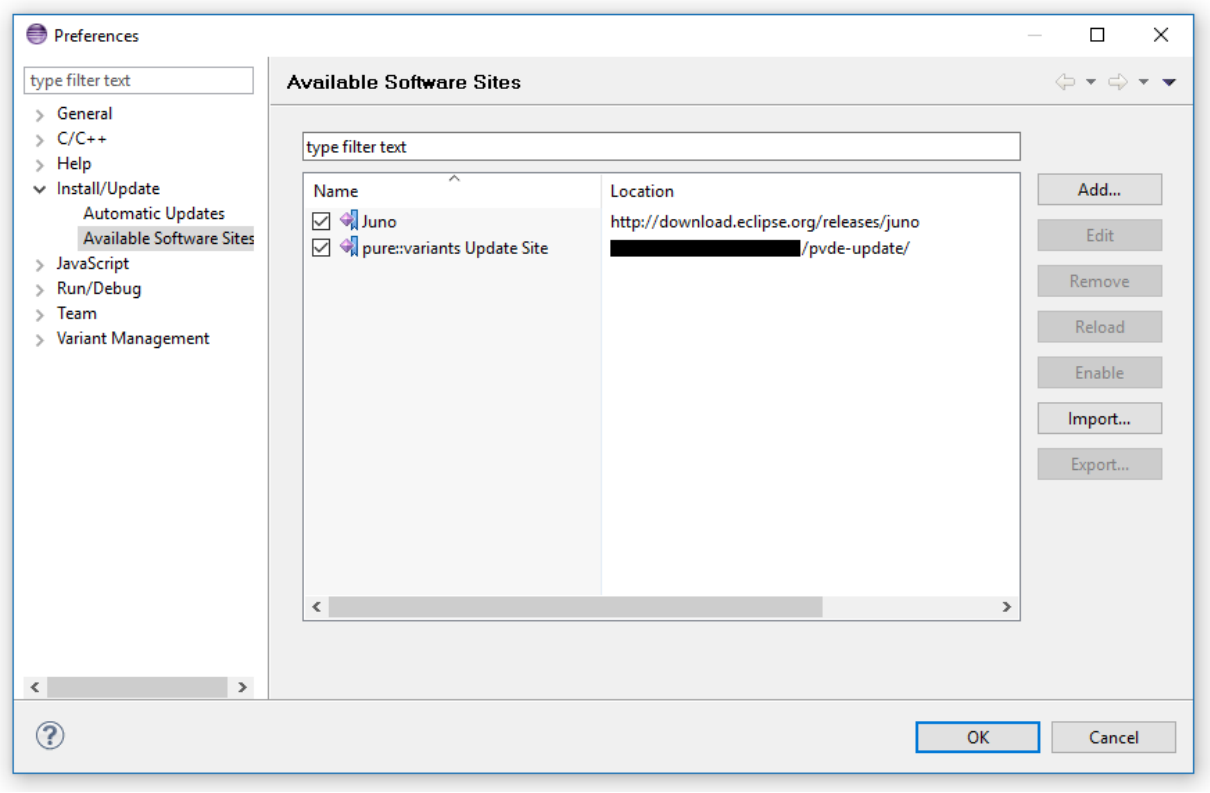
#### Using update site

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

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

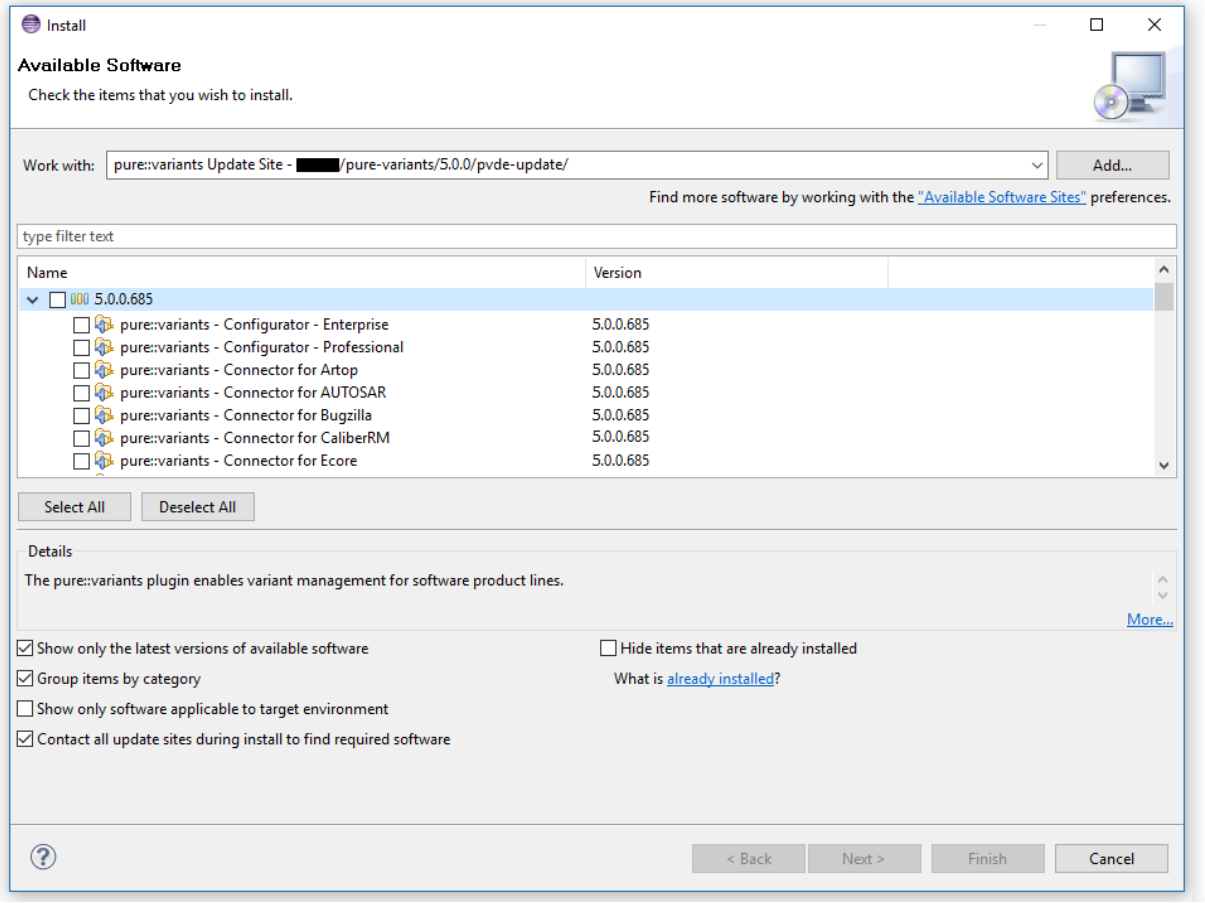
  The location of the site depends on the pure::variants product variant. Visit the pure-systems web site (https://www.pure-systems.com) or read your registration email to find out which site is relevant for the version of the software you are using.

**Figure 8. Update Site Selection**



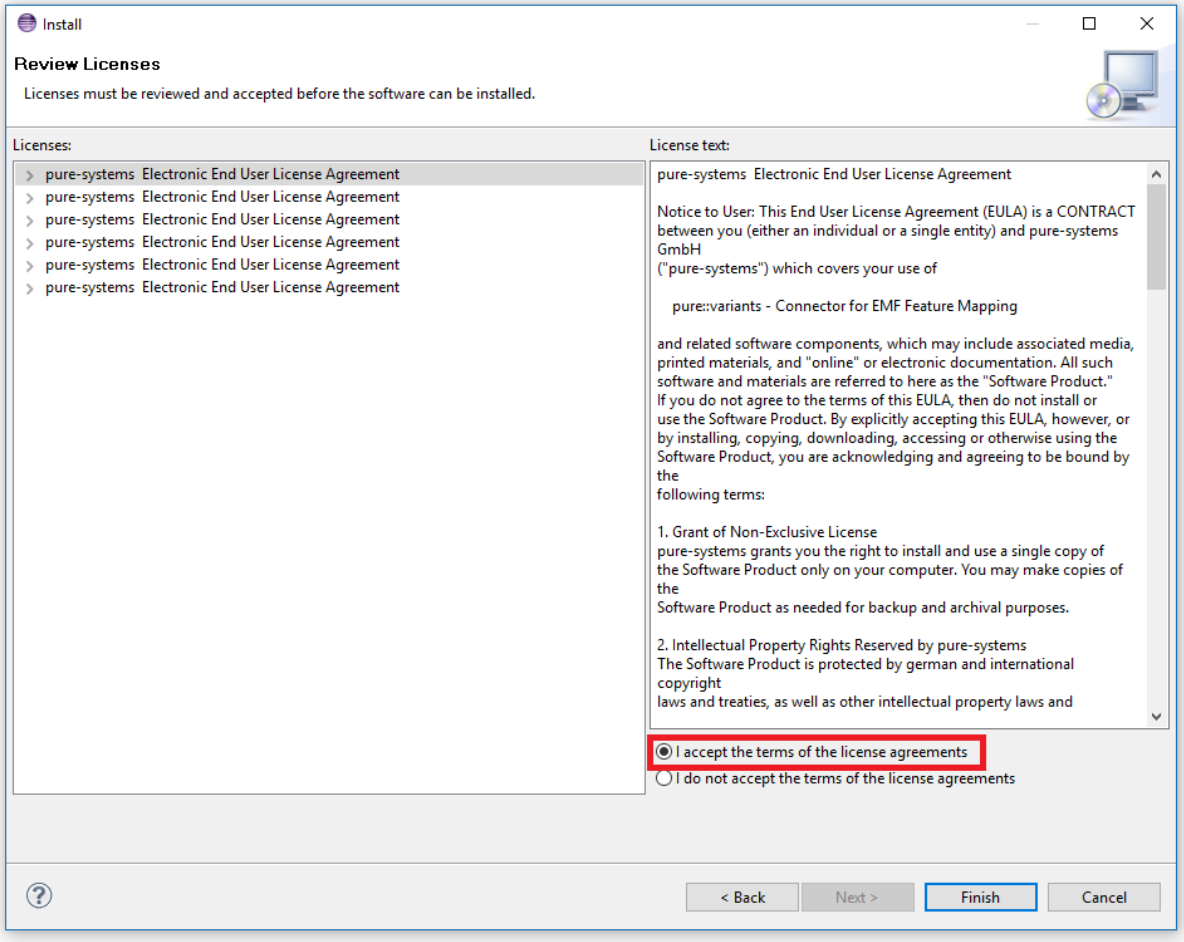

- Unfold the pure::variants update site and select all features to be updated. Select "Next".

**Figure 9. Pure::variants Plugin Selection**




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

**Figure 10. Licence Agreement**




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

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

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

and download the "Complete Updatesite" archive:

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

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

## 3.2. Update pure::variants Client

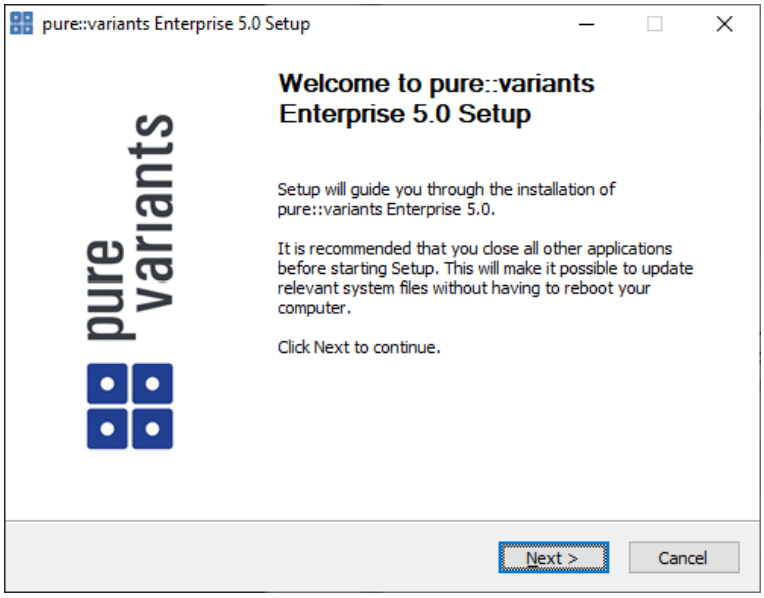
### 3.2.1. Update with pure::variants Installer

This update method is available for Windows only. If you do not use Windows please see Section 3.2.2, “Update with Update Action” or Section 3.2.3, “Update with Eclipse package manager”.

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

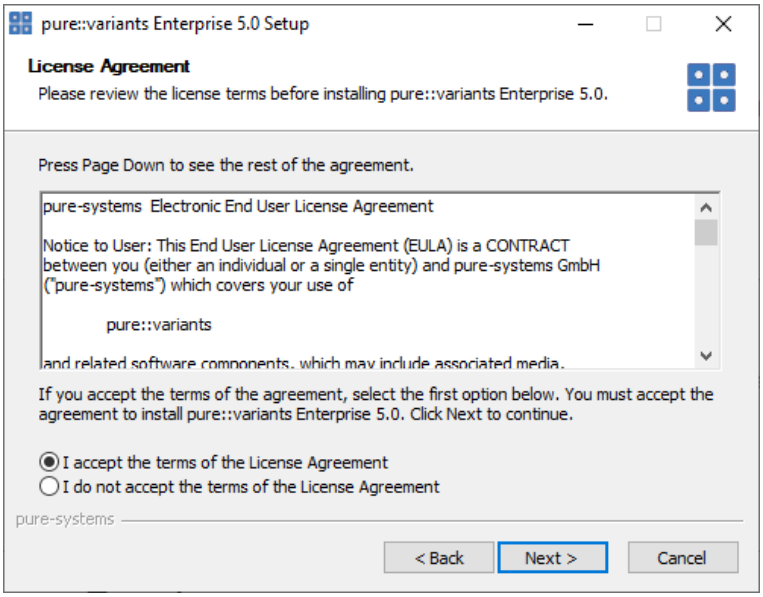
Download the installer package ("pure::variants Windows Installer Package") and extract it. The installer will check for an existing pure::variants client installation and start in update mode if it finds one. Start the update by double-clicking "Setup Enterprise X.Y.ZZ.exe". Running the pure::variants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

**Figure 11. pure::variants Client Installer**



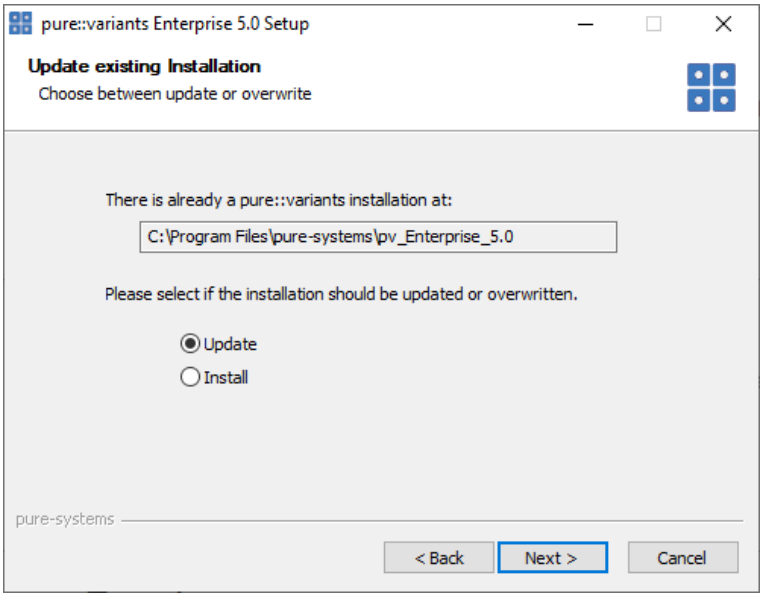

Click *Next*.

**Figure 12. Setup pure::variants Client License**




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

**Figure 13. Choose Update Mode**

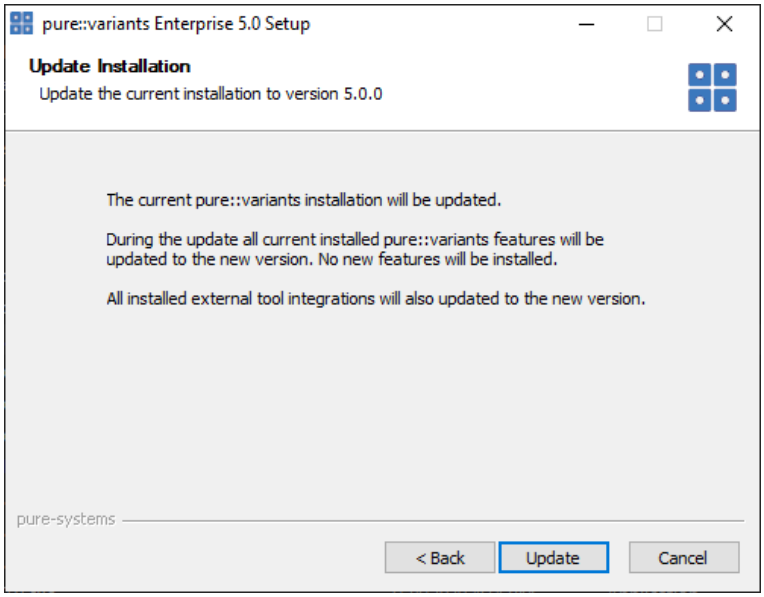



Choose *Update* if the current pure::variants client installation shall just be updated with the same installed feature and settings. The installed pure::variants integrations will also be updated. The installed components cannot be changed. If a change of the installed components is wanted, choose *Install*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

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

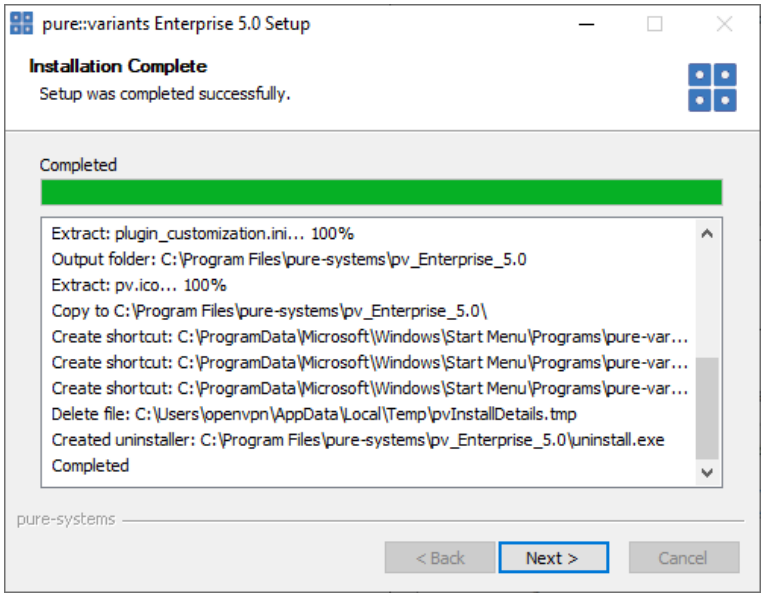
Click *Next*.

**Figure 14. pure::variants Start Update**



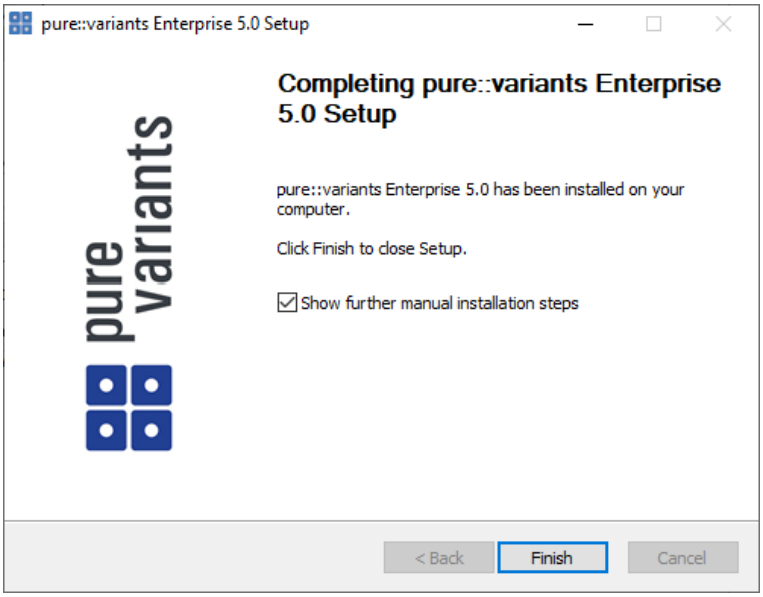

Click *Update* to start the update process.

**Figure 15. pure::variants Installation Progress**




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

**Figure 16. Update pure::variants Client Finish Page**




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

## 3.2.2. Update with Update Action

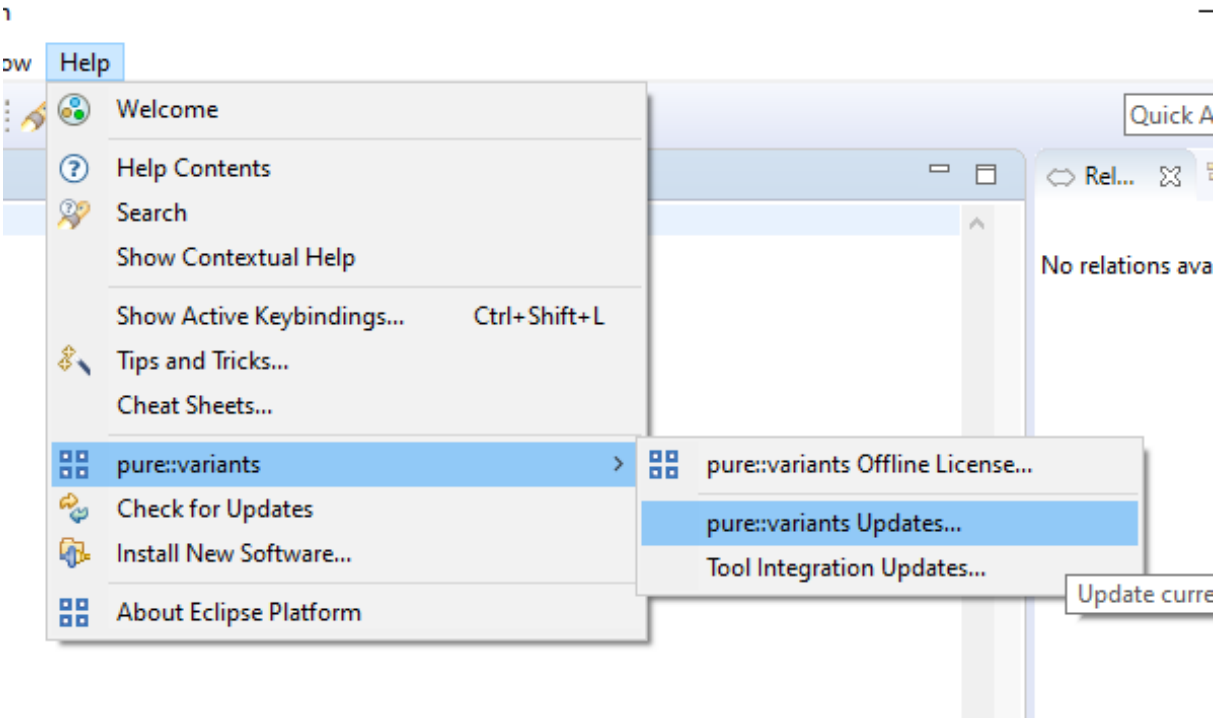
pure::variants has a built-in update action which can be used to perform an update with all the currently installed pure::variants extensions. This update action does not update the installed pure::variants integrations automatically. But they can be easily updated with the *Tool Integration Update*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

The update action requires administrator privileges.

### Note

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

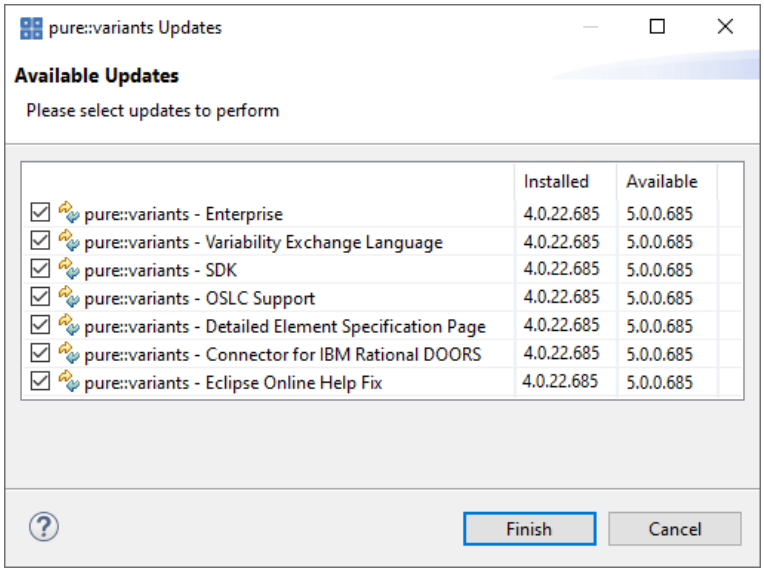
**Figure 17. Start pure::variants Client Update**




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

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

**Figure 18. Start pure::variants Client Update**




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

**Figure 19. Start pure::variants Client Update**


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

### 3.2.3. Update with Eclipse package manager

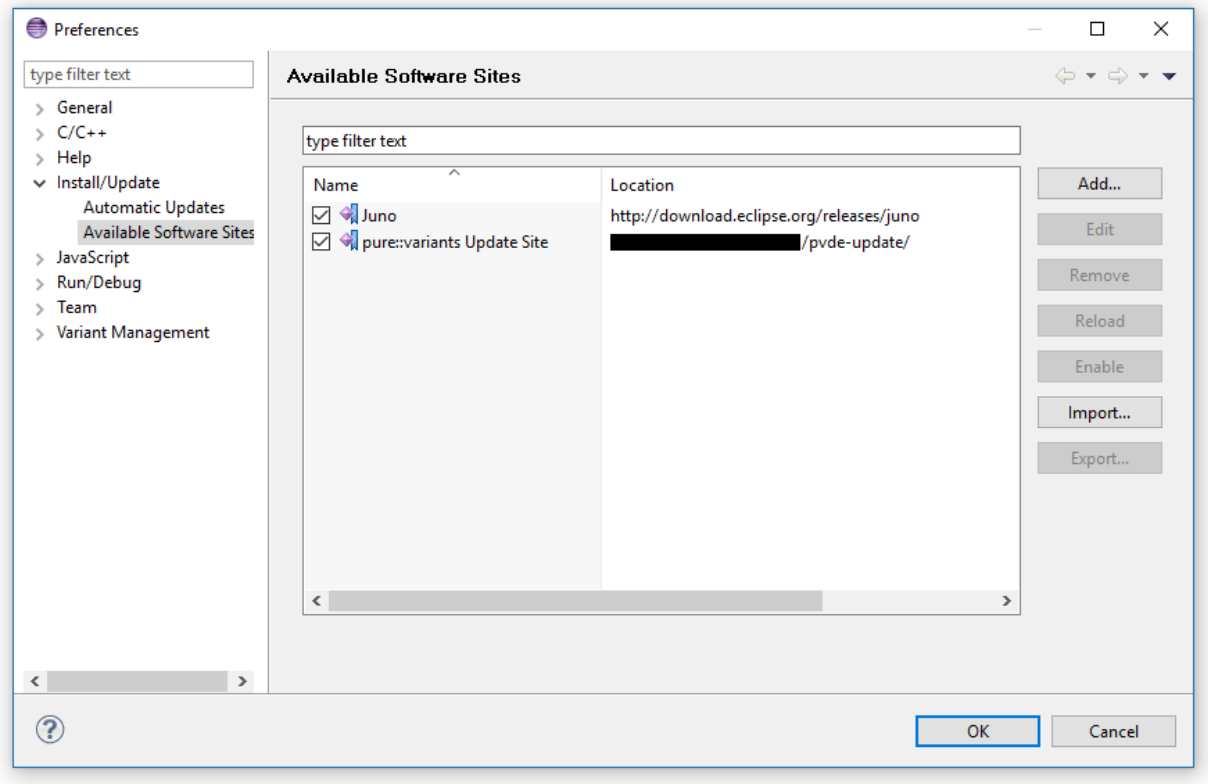
The quickest way to get a update for pure:.variants is to run the software updater inside pure::variants:

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

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

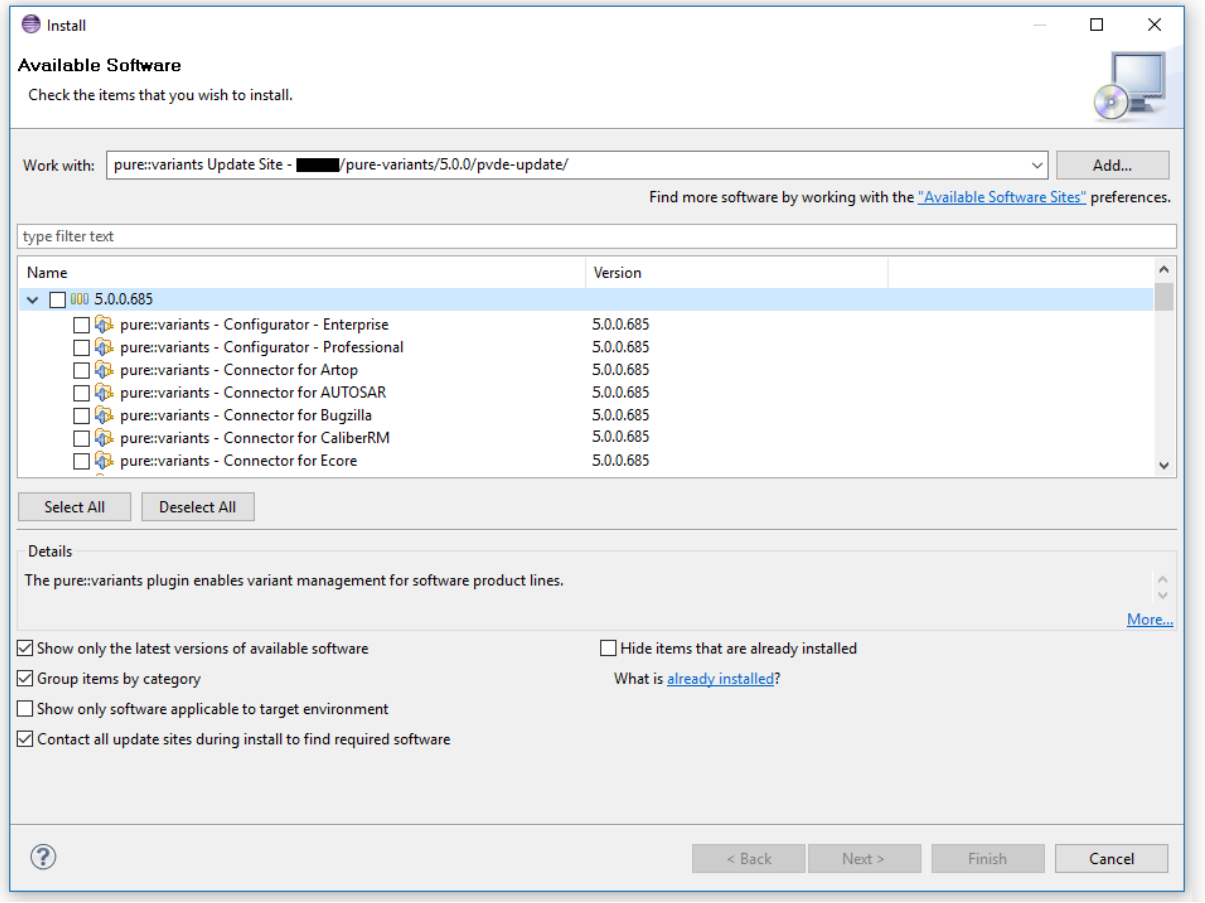
  The location of the site depends on the pure::variants product variant. Visit the pure-systems web site (https://www.pure-systems.com) or read your registration email to find out which site is relevant for the version of the software you are using.

**Figure 20. Update Site Selection**



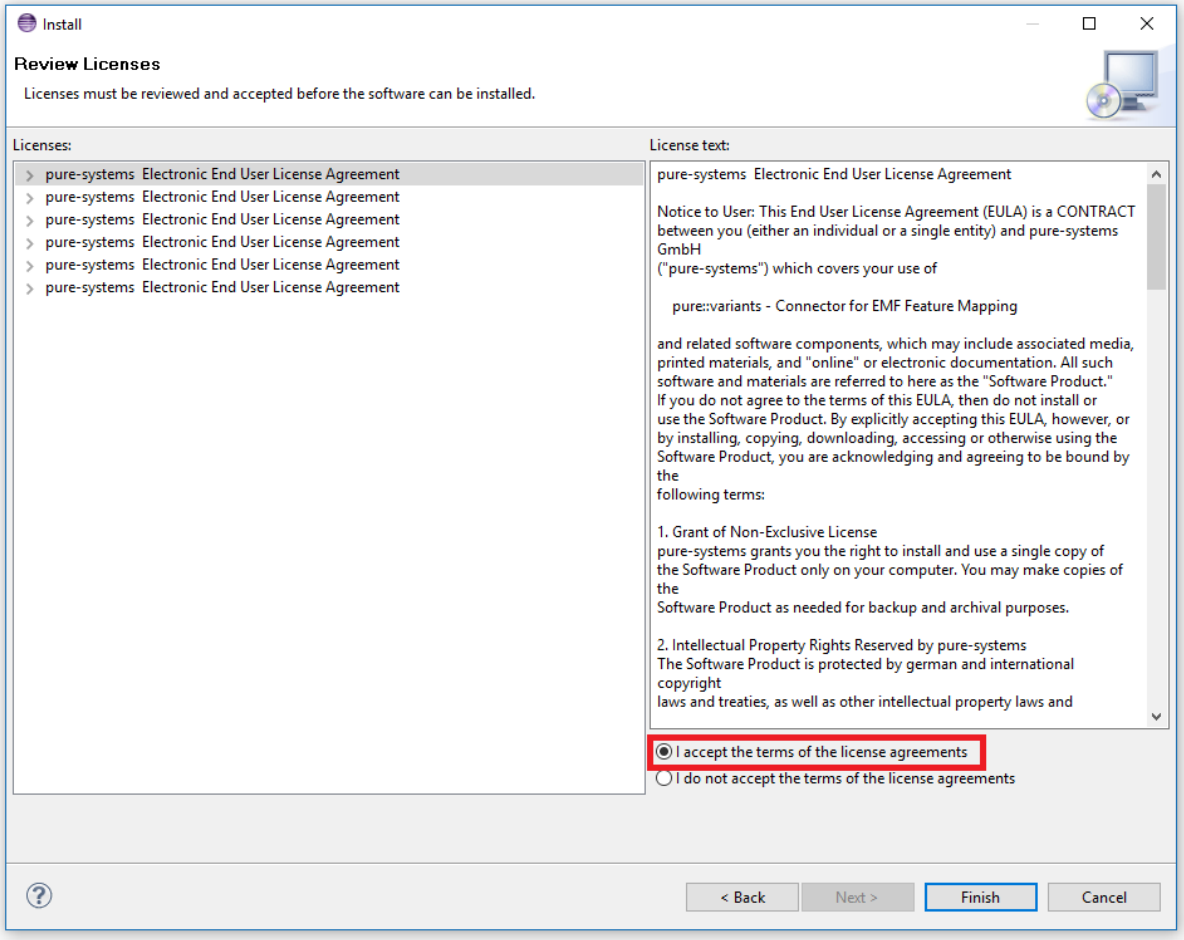

- Unfold the pure::variants update site and select all features to be updated. Select "Next".

**Figure 21. pure::variants Plugin Selection**




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

**Figure 22. Licence Agreement**




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

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

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

and download the "Complete Updatesite" archive:

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

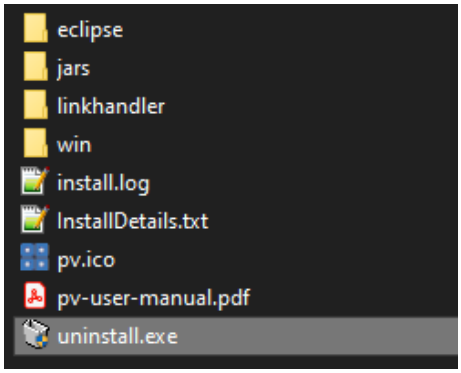
- Unfold everything below pure::variants update site and select the features to be updated. Select "Next".
- Accept the license, hit "Next" and then "Finish".
- In the dialog select "Install all".
- Restart pure::variants when asked for.

## 3.3. Uninstall pure::variants Client

### 3.3.1. Uninstall using pure::variants Uninstaller

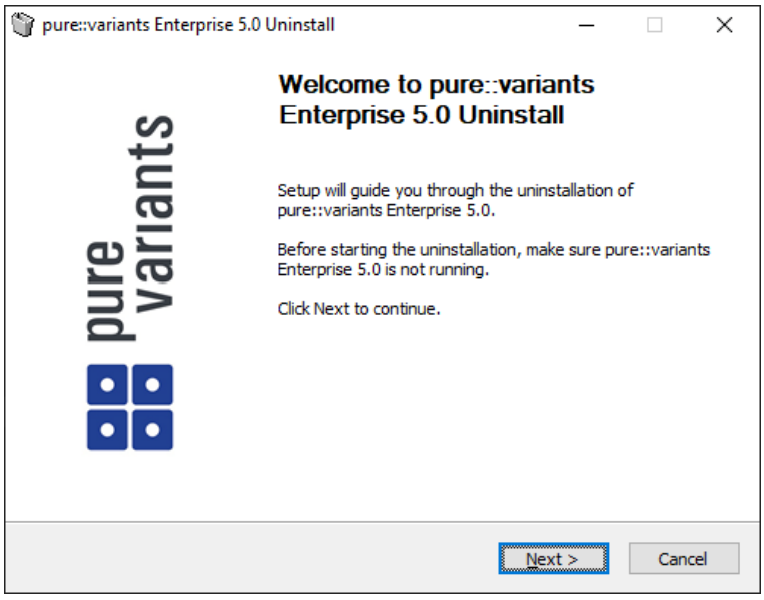
The uninstaller for the pure::variants client can be started in two different ways. The first is to go to the Windows *Add or remove programs* application and search for *pure::variants Enterprise* and start the uninstaller by using the *Uninstall* action. The uninstaller requires Administrator privileges.

**Figure 23. pure::variants Client Uninstaller**



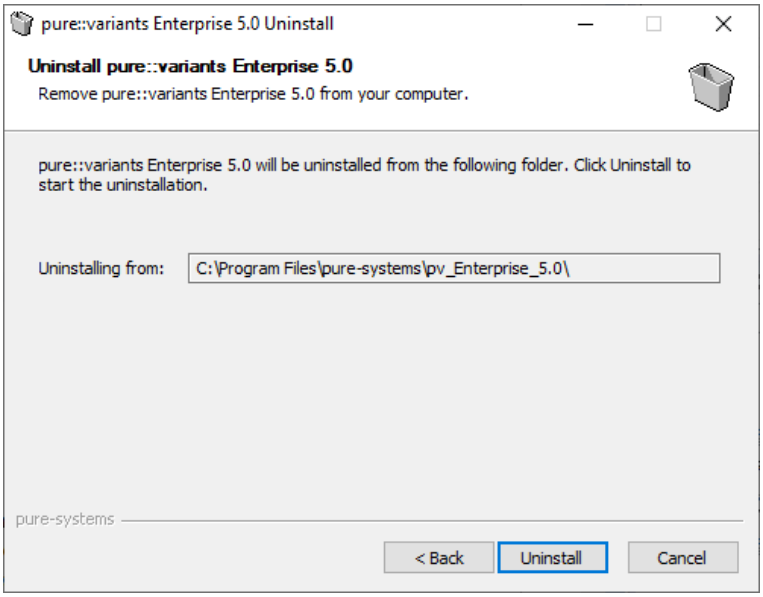

The second way is to navigate to the pure::variants client installation folder and start the uninstaller by double clicking it.

**Figure 24. pure::variants Client Uninstaller**



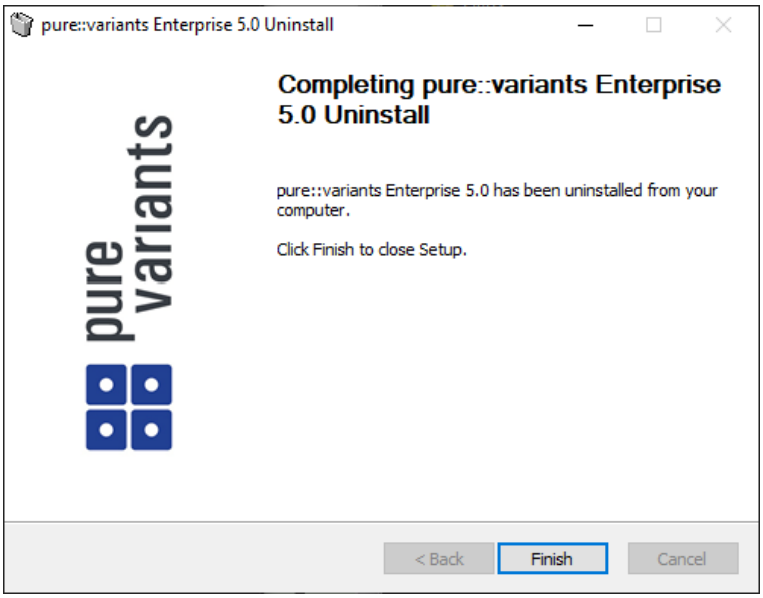

Click *Next*.

**Figure 25. Uninstall from**




Click *Uninstall* to start the uninstall process.

**Figure 26. Completing Uninstall**




Click *Finish* to close the uninstaller.

## 3.3.2. Uninstall pure::variants from existing Eclipse instance

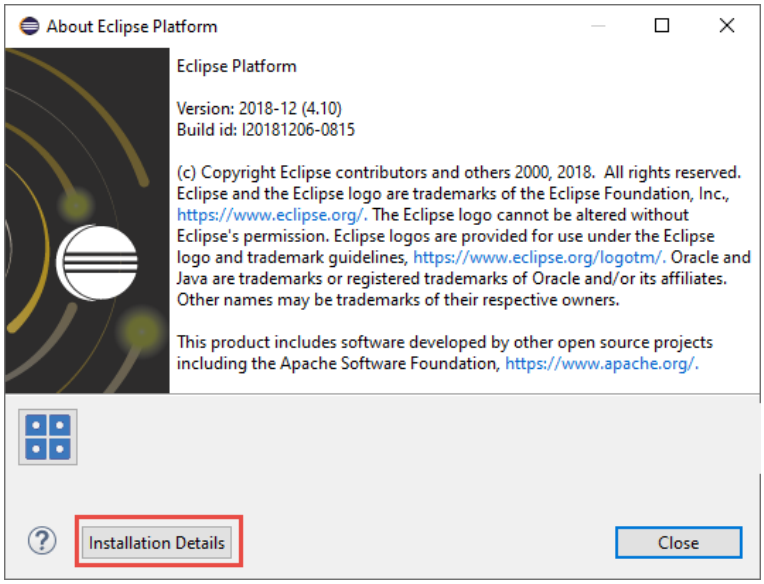
There are two ways to remove pure::variants from an Eclipse instance. You can use the Eclipse command line or remove the pure::variants features one by one in the running Eclipse Instance. Either way a cleanup of the Eclipse instance has to be performed afterwards.

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

### Uninstall pure::variants in running Eclipse Instance

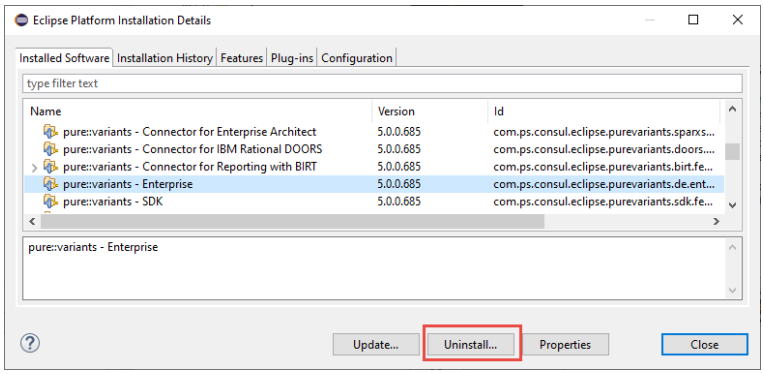
To remove an installed feature from Eclipse using the Eclipse client, open the *About Eclipse Platform* dialog with the *About Eclipse Platform* action in the *Help* menu.

**Figure 27. Eclipse About Dialog**



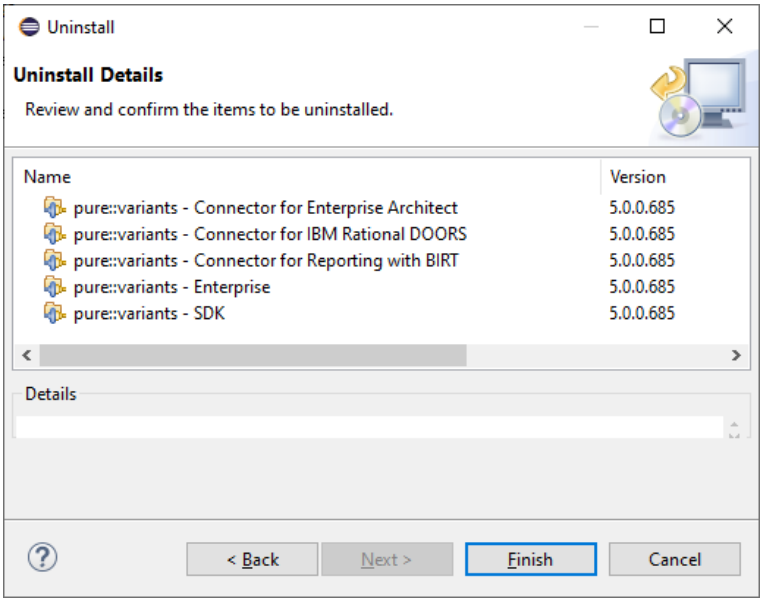

Use the *Installation Details* button to access the installation details.

**Figure 28. Eclipse About Dialog**




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

**Figure 29. Eclipse About Dialog**

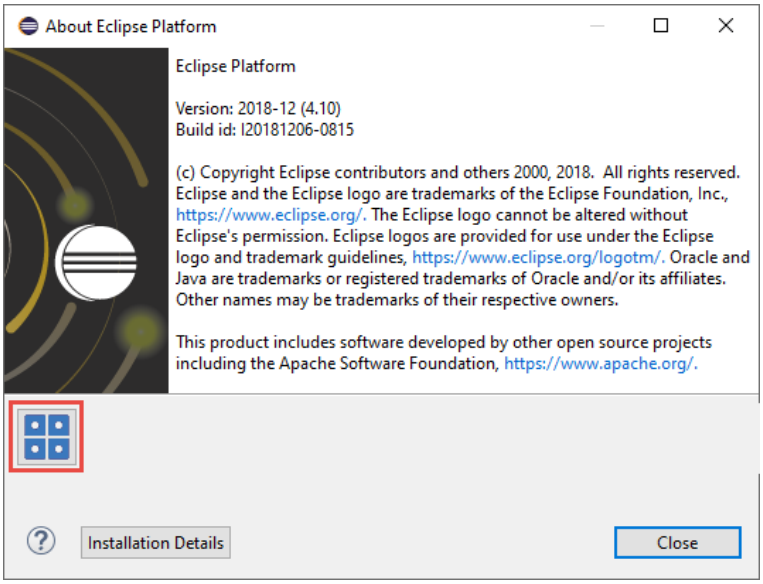



Click *Finish* to start the uninstall process. After it finished, Eclipse will prompt you to restart the application. Click *Restart* to finish the uninstallation.

## Uninstall pure::variants using Eclipse uninstall application

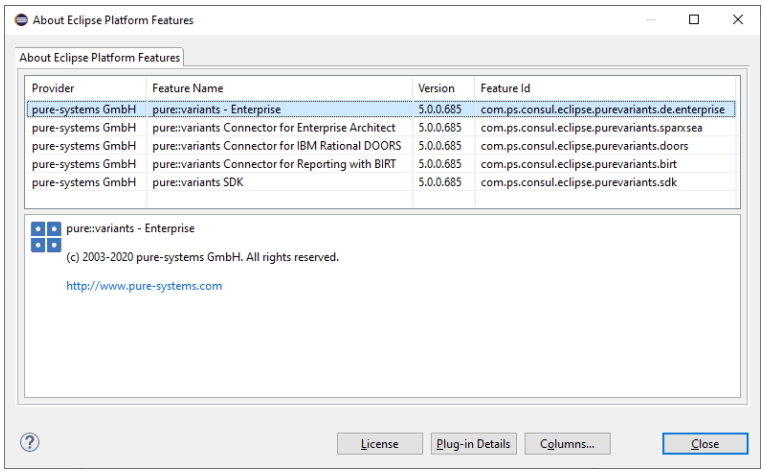
To use the uninstall application you need the feature ids of the features you want to uninstall. The feature ids can be found in the *About Eclipse Plattform* dialog. Open the dialog with the *About Eclipse Platform* action in the *Help* menu.

**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-5, and the second C:\ProgramData\pure-variants-5. On Linux based systems the pure-variants-5 folders are located in the users home directory and at */usr/share*. These folders should be removed after pure::variants has been completely removed from the computer.

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

```
"<Eclipse Installation Directory>\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
```

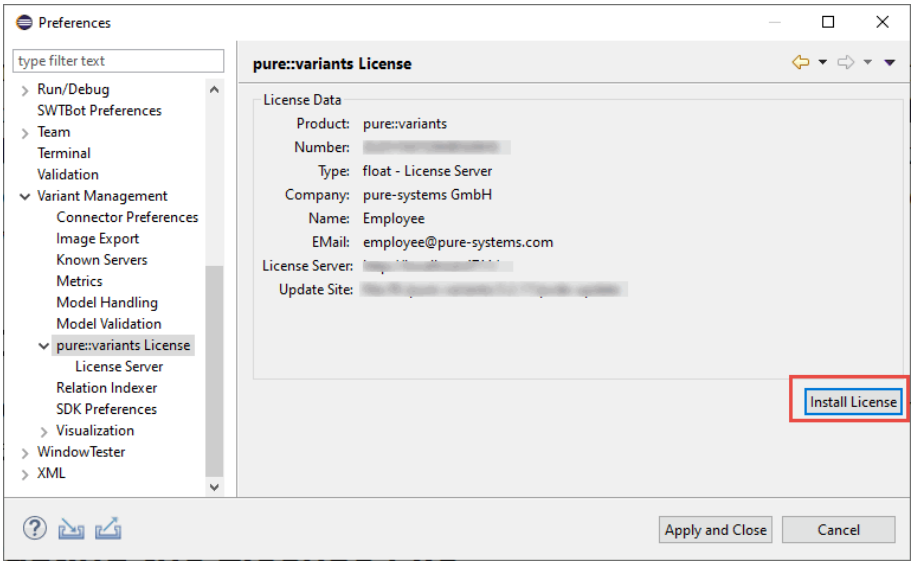
## 3.4. Basic Setup of the pure::variants Client

### 3.4.1. Setup a pure::variants Client License

A valid license file is required in order to use pure::variants. If pure::variants is started and no license is present, then the user is prompted to supply a license. Select the “Yes” button and use the file dialog to specify the license file delivered with pure::variants. The specified license will be stored in the user's application data directory. If you are using multiple workspaces then the license file has to be installed only once. The pure::variants integrations also use the installed license and thus no further setup step is needed here.

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

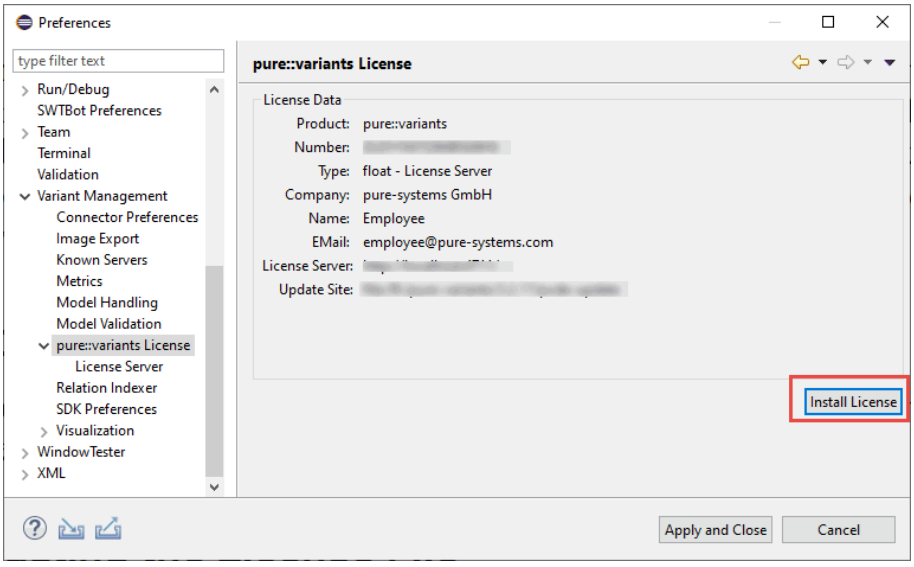
**Figure 32. pure::variants License Preferences**




## 3.4.2. Update a pure::variants Client License

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

**Figure 33. pure::variants License Preferences**




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

For central or automatic deployed pure::variants clients it may be necessary to also automatically deploy or update the pure::variants client license. For this use case the variable **PVLICENSE** can be used. This variable can either be introduced as an environment variable or just added as a Java property to the command line starting pure::variants. If this variable is set, the given license is used instead of a possibly previously installed license.

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

## 3.5. Trouble Shooting

### 3.5.1. pure::variants is low on memory

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

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

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

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

#### Note

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

# 4. pure::variants Connectors

## 4.1. Installation of pure::variants Connectors

Installing a connector into an existing pure::variants installation works the exact same way like installing the pure::variants client into an 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”.

## 4.2. Connector for Codebeamer

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

### 4.2.1. Installation of pure::variants Client

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

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

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

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

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

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

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

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

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

### 4.2.3. Installation without running in a docker container

Follow the steps to install:

1. Stop the codeBeamer server
2. Copy ‘com.ps.consul.codebeamer.vel.jar’ and 'pvcore.jar' found in the zip archive to <codeBeamer>/tomcat/webapps/cb/WEB-INF/lib
3. Copy following files included in 'pv_integration' to following locations:

   'pv_integration/widget' to <codebeamer>/tomcat/webapps/pv-widget/, e.g. /home/appuser/codebeamer/tomcat/webapps/pv-widget/
4. Restart the Codebeamer server

### 4.2.4. Installation in a docker image

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

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

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

```
volumes:
      - ./pv_integration/widget:<codebeamer>/tomcat/webapps/pv-widget/

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

Then follow the steps to install:

1. Shut down the docker container first
2. Copy ‘com.ps.consul.codebeamer.vel.jar’ and 'pvcore.jar' found in the zip archive to a location accessible by docker, and as defined in the volumes mapping (see above)
3. Copy the folder 'pv_integration' including all content to a location accessible by docker, and as defined in the volumes mapping (see above). When updating, please make sure to remove old content of the complete directory first.
4. Restart the docker container

5. In Codebeamer, add following the "externalWidgetExtensions" section to the Application Configuration (https://<codebeamer>/sysadmin/configConfiguration.spr) as System Administrator:

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

## 4.2.5. Permissions

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

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

## 4.2.6. Getting Version Information of the Server Component

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

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

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

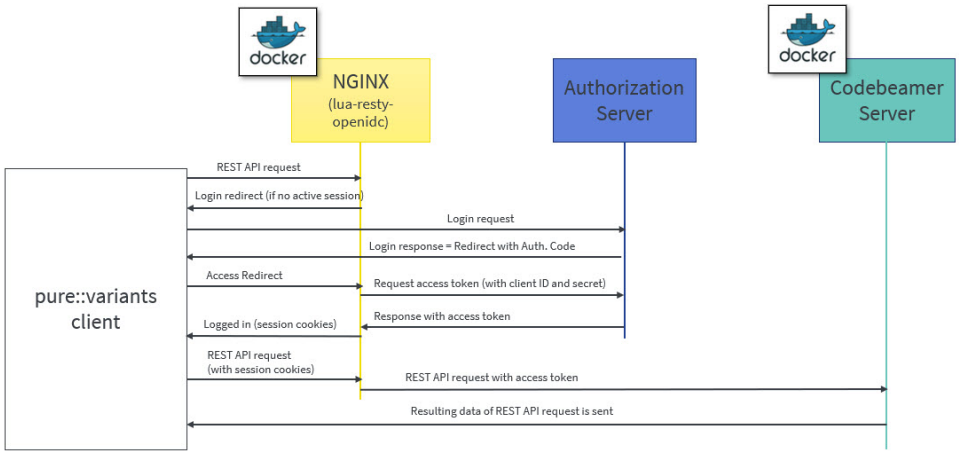
## 4.2.7. Configuration To Enable Open ID Connect (OIDC) Authentication

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

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

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

**Figure 34. Authorization process using a proxy**




Client registration steps:

1. During the client registration process, both codeBeamer and the Authentication Proxy needs to be registered.

2. The provided configuration files use the 'lua openresty' library for NGINX, implementing OIDC.

3. The registered client's Client ID and Client secret should be added in the configuration files of docker-compose and NGINX (see later).

Following chapter describe the docker configuration files and their parameters.

### 4.2.8. docker-compose.yml for the NGNIX Proxy

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

Following parameters are to be set:

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

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

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

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

Following code listing shows an example:

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

### 4.2.9. oidc-auth-proxy.dockerfile for the NGNIX Proxy

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

Following code listing shows an example:

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

### 4.2.10. oidc-auth-proxy-nginx.conf for the NGNIX Proxy

The directives that need to be adapted are as follows:

- Set **listen** port to which NGINX should listen to.

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

Following code listing shows an example:

```
events {
  worker_connections 128;
}

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

  server {
    listen 9943 ssl; ##mention the port to which nginx should listen to
    server_name codebeamer.example.com; ##domain name or ip address of the host on which
 docker is running
    ssl_certificate /usr/local/openresty/nginx/server.cert;
    ssl_certificate_key /usr/local/openresty/nginx/server.key;
    ssl_protocols TLSv1.2 TLSv1.3;

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

```
      proxy_pass http://codebeamer-app:8090;
    }
  }
}
```

## 4.2.11. Steps to Setup a Docker-container the NGNIX 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.