pure::variants Setup Guide

Parametric Technology GmbH

Version 7.0.0.221 for pure::variants 7.0

Copyright © 2003-2025 Parametric Technology GmbH

2025

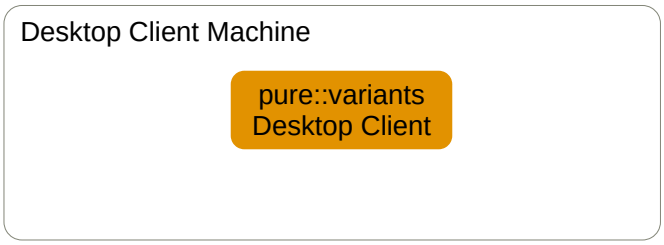
Table of Contents

1. Introduction

This setup guide describes how to install and update pure::variants Community on a personal computer.

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

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

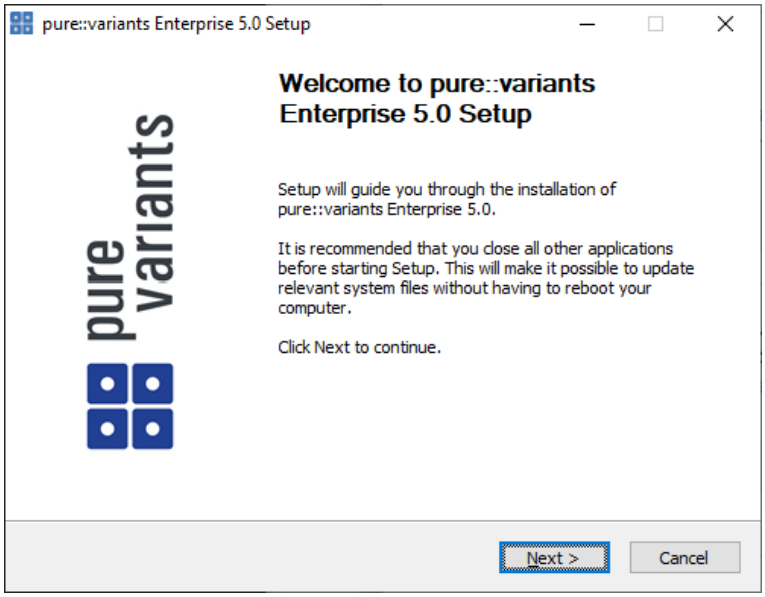
- supported 64-Bit Java version installed

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.

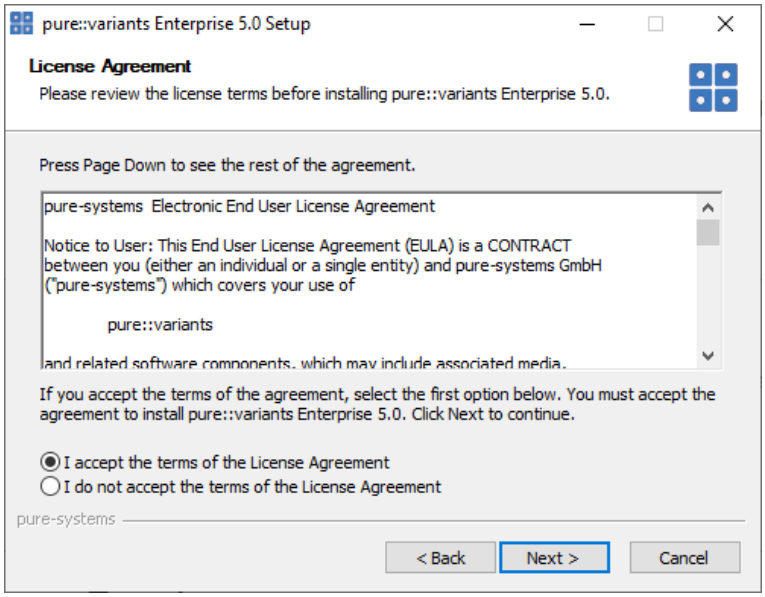
All pure::variants extensions available for the account are automatically included in the Windows Installer download. However, some may not be enabled by default in Installer. Make sure to select the desired extensions during the installation process. Later updates to the extension selection can be done 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



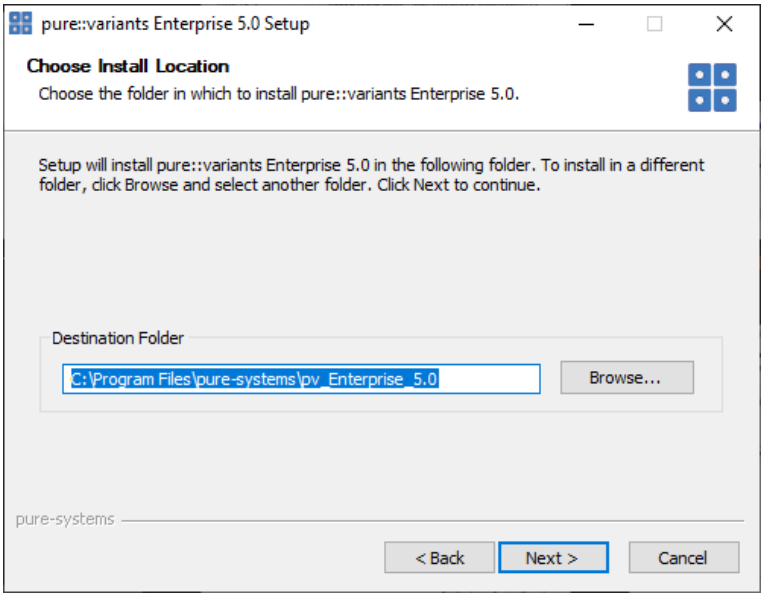
Click *Next*.

Figure 3. Setup pure::variants Desktop Client License



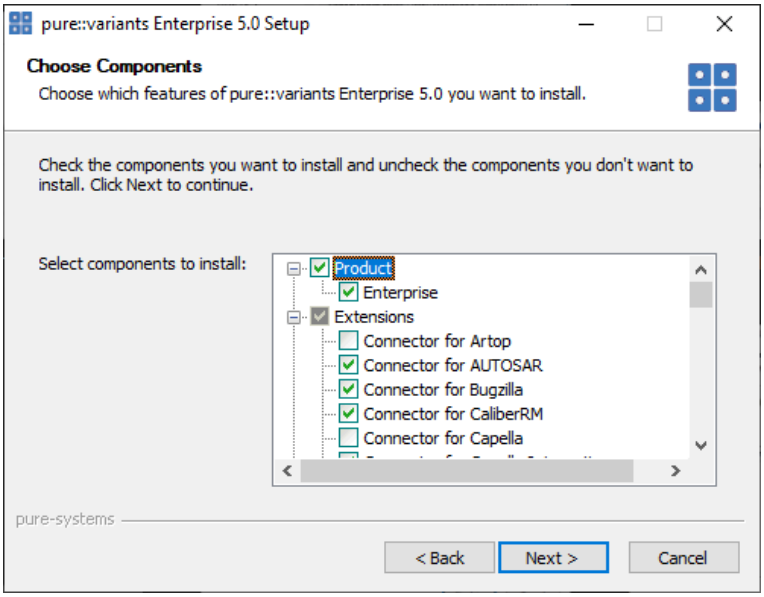
Read the license agreement and after accepting it click *Next*.

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



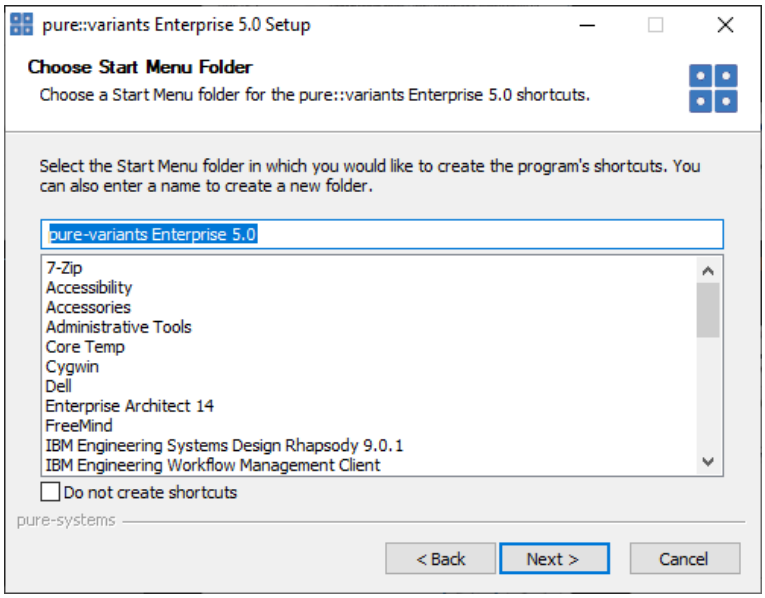
Select the folder where to install the pure::variants Desktop Client files. Click *Next*.

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

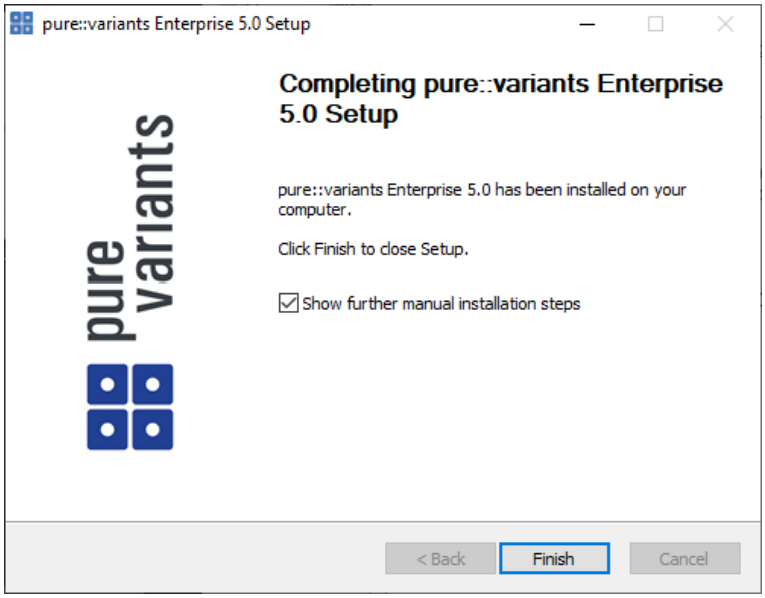


Enter the name for the Windows start menu entry, or disable the creation of the start menu entry. Click *Next*

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.

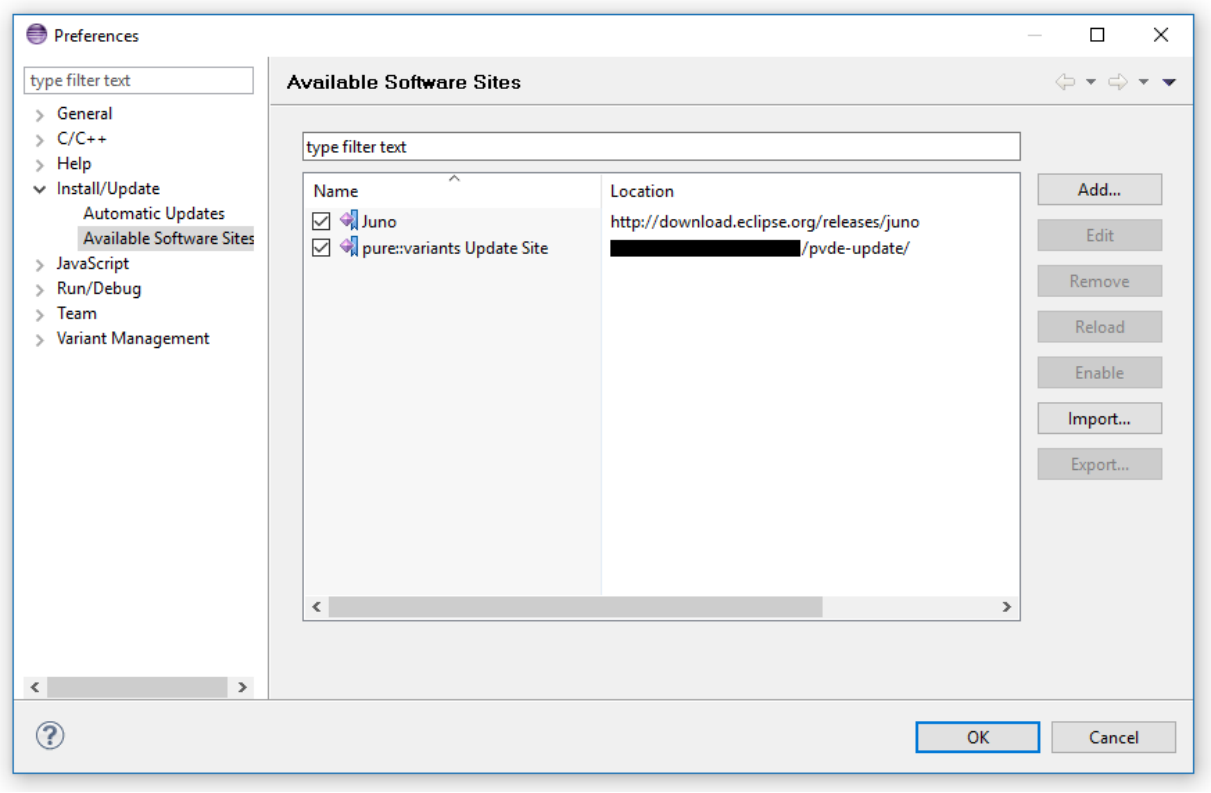
Using update site

- 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.

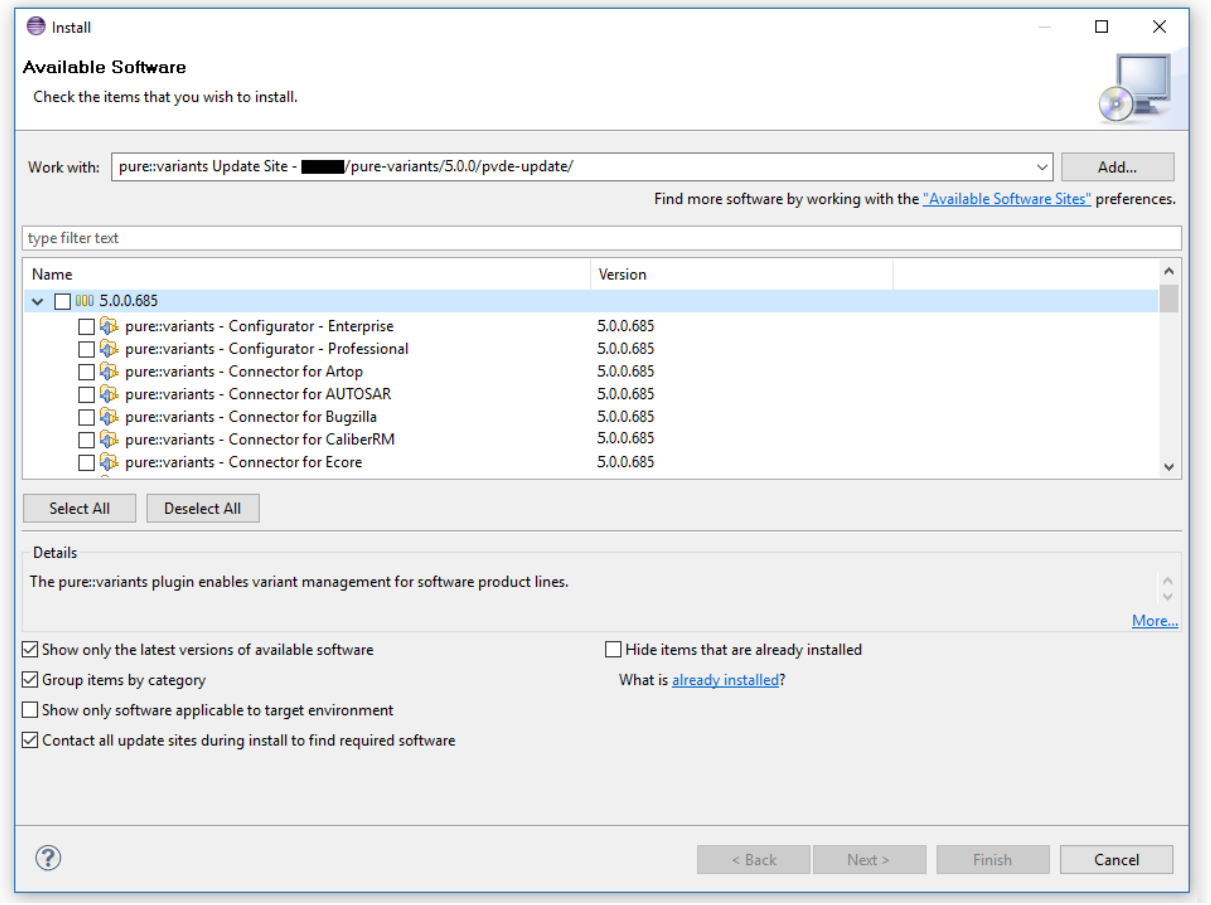
  The location of the site depends on the pure::variants product variant. Visit the Parametric Technology web site (https://www.pure-systems.com) or read your registration email to find out which site is relevant for the version of the software you are using.

Figure 8. Update Site Selection



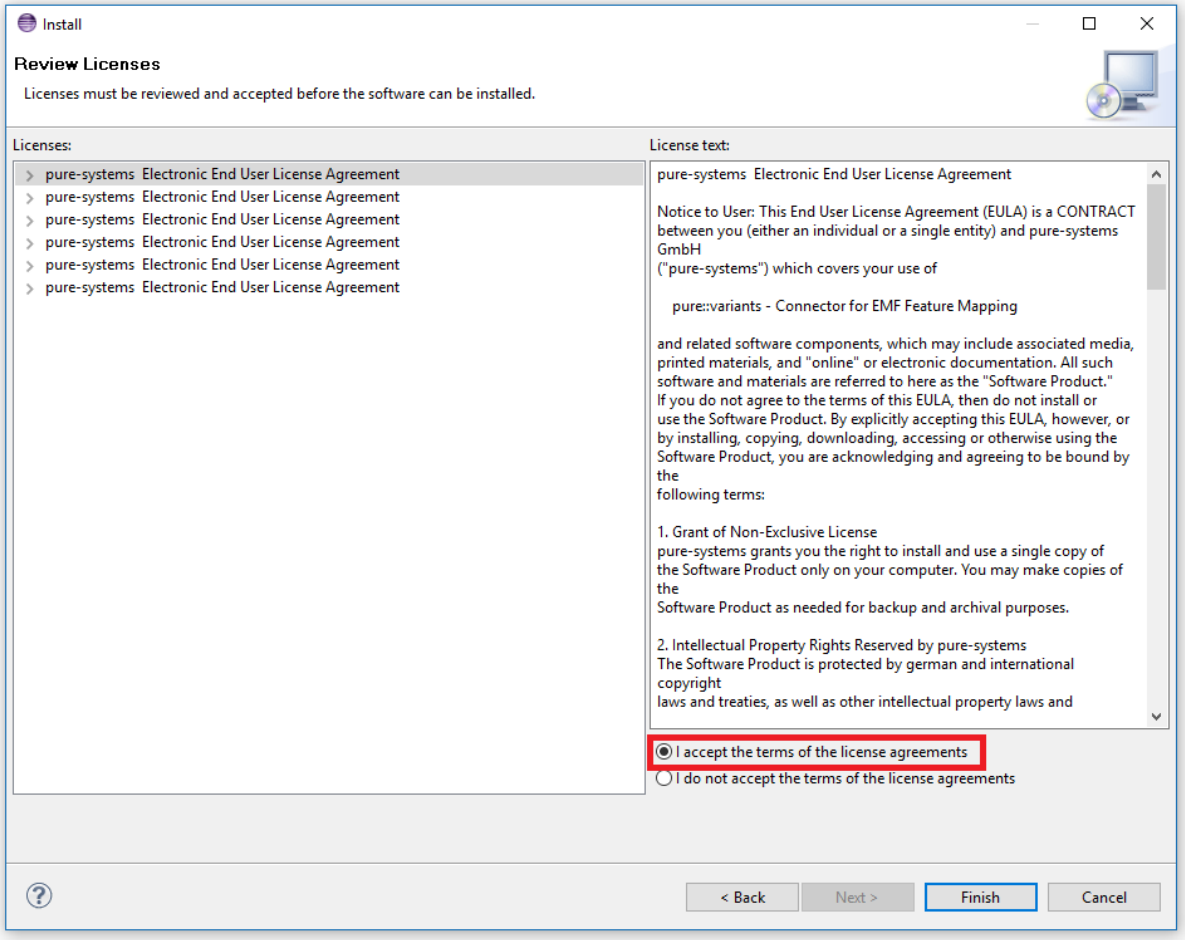
- Unfold the pure::variants update site and select all features to be updated. Select "Next".

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

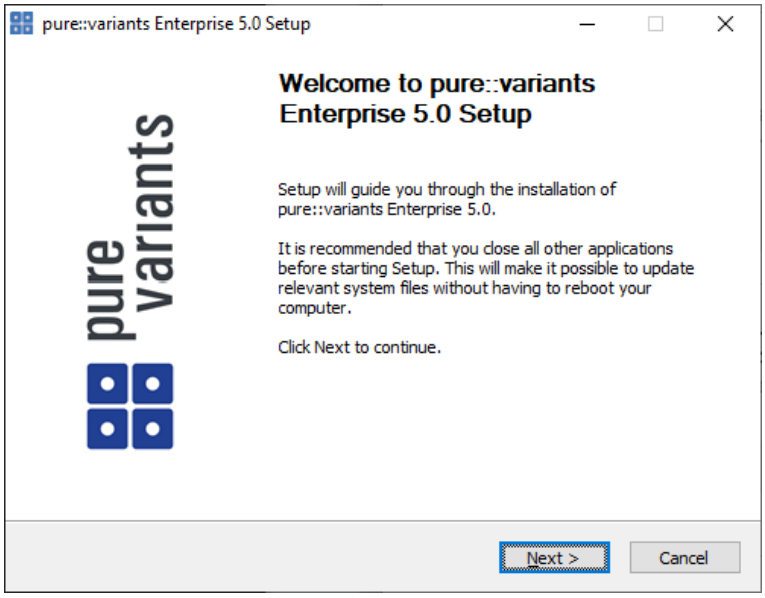
3.2.1. Update with pure::variants Installer

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.

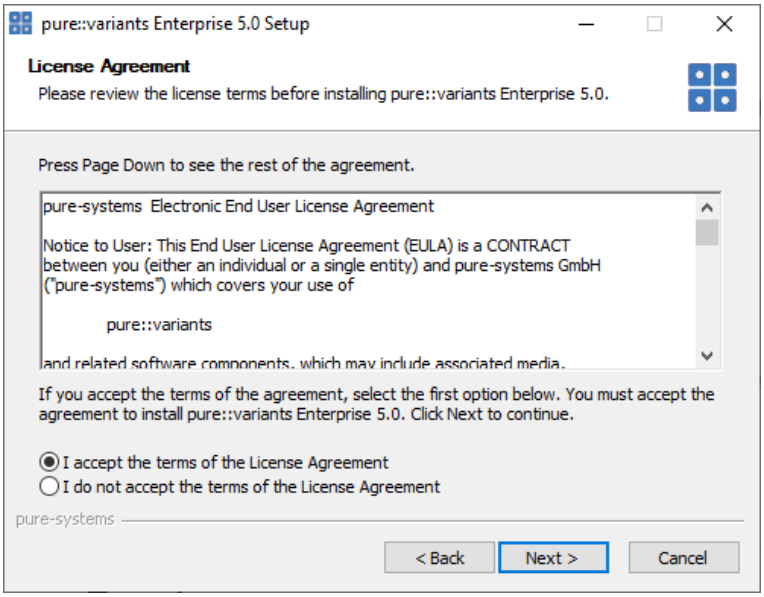
Download the installer package ("pure::variants Windows Installer Package") and extract it. The installer will check for an existing pure::variants Desktop Client installation and start in update mode if it finds one. Start the update by double-clicking "Setup Enterprise X.Y.ZZ.exe". Running the pure::variants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

Figure 11. pure::variants Desktop Client Installer



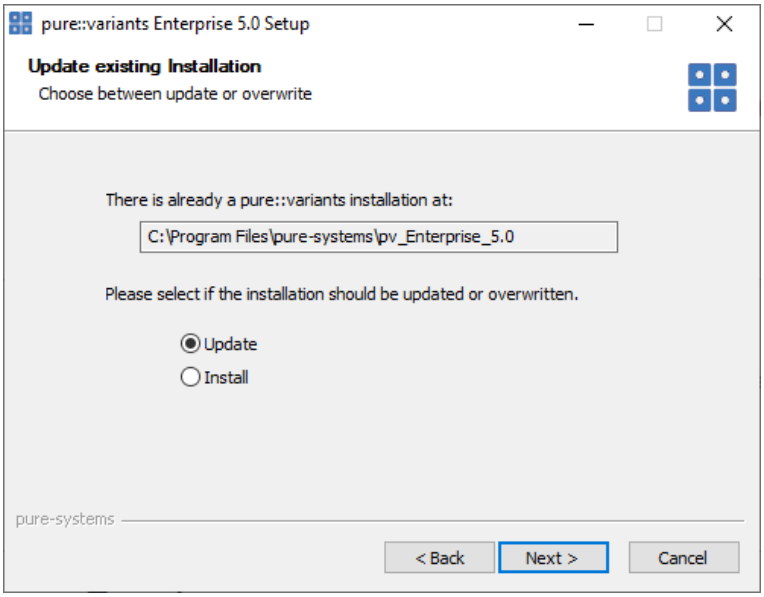
Click *Next*.

Figure 12. Setup pure::variants Desktop Client License



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

Figure 13. Choose Update Mode

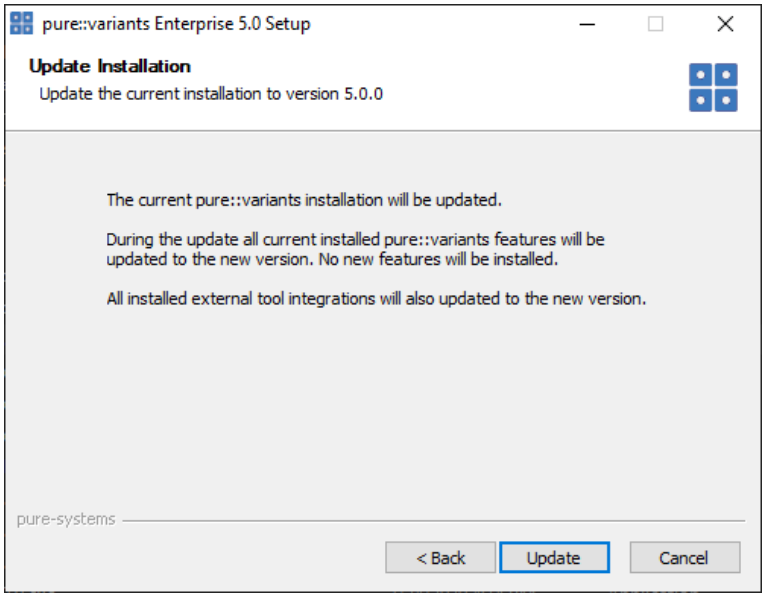


Choose *Update* if the current pure::variants Desktop Client installation shall just be updated with the same installed feature and settings. The installed pure::variants integrations will also be updated. The installed components cannot be changed. If a change of the installed components is wanted,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

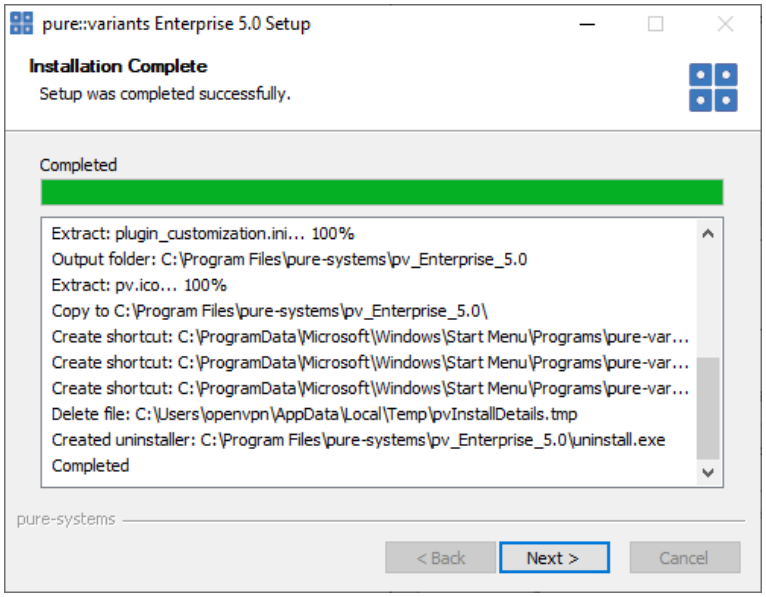
Click *Next*.

Figure 14. pure::variants Start Update



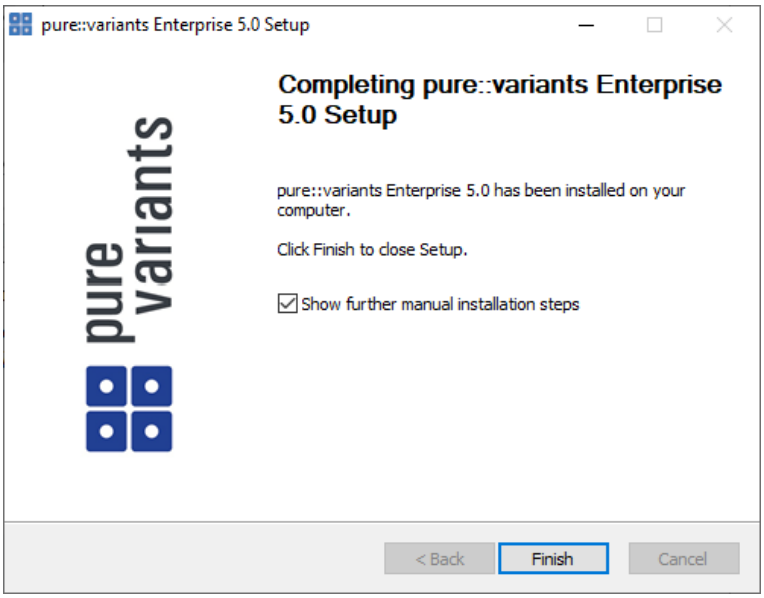
Click *Update* to start the update process.

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

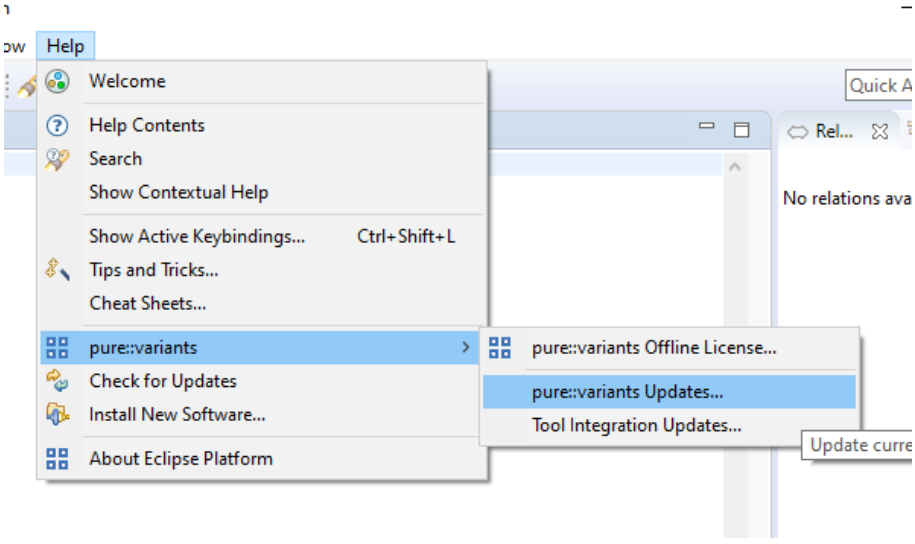
pure::variants has a built-in update action which can be used to perform an update with all the currently installed pure::variants extensions. This update action does not update the installed pure::variants integrations automatically. But they can be easily updated with the *Tool Integration Update*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

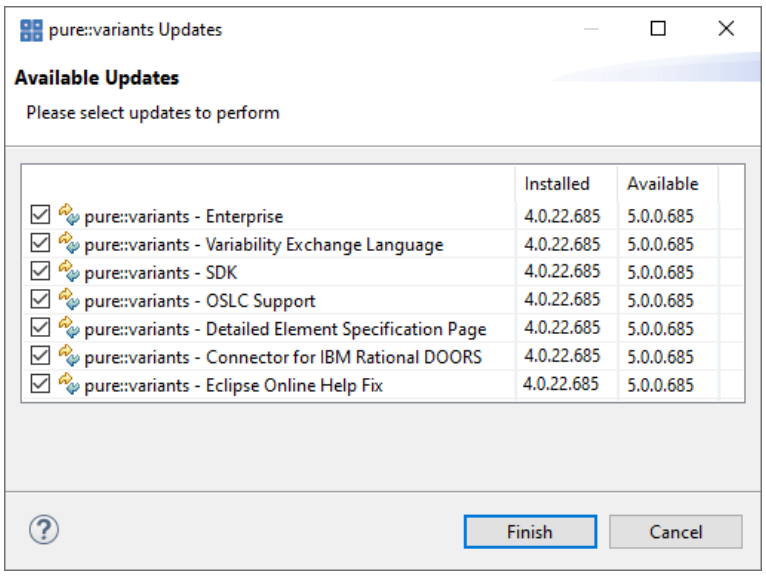
Figure 17. Start pure::variants Desktop Client Update



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

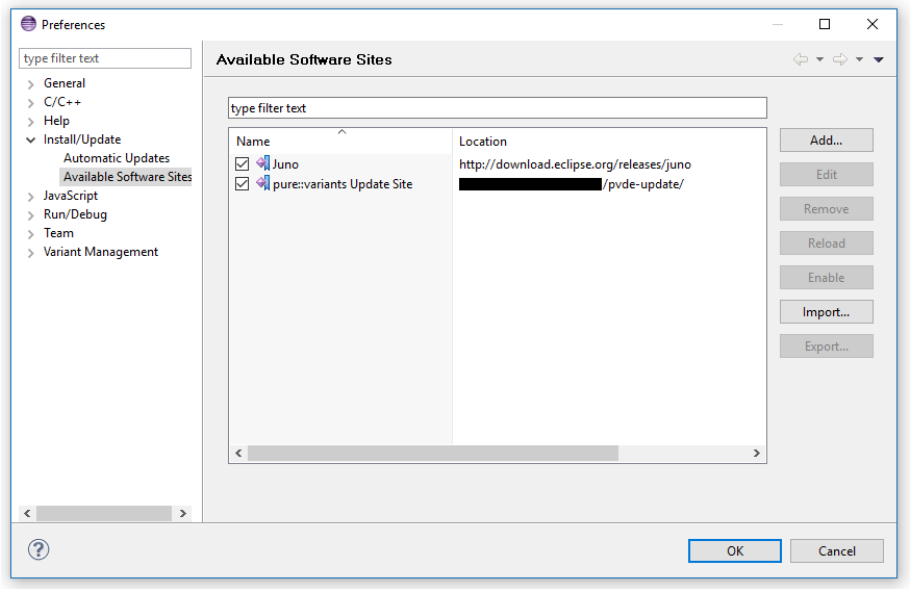
The quickest way to get a update for pure:.variants is to run the software updater inside pure::variants:

- 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.

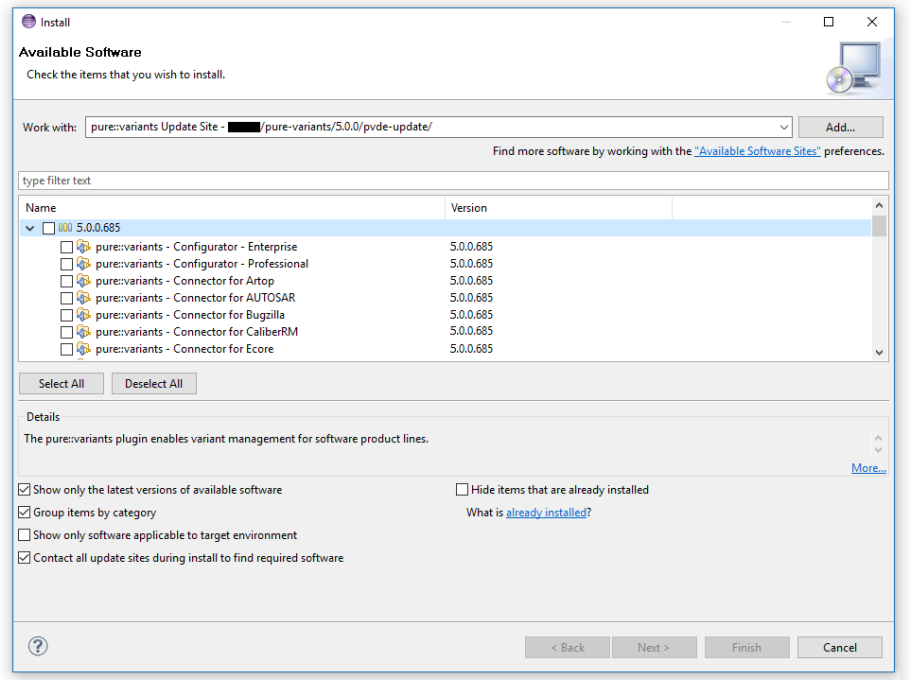
  The location of the site depends on the pure::variants product variant. Visit the Parametric Technology web site (https://www.pure-systems.com) or read your registration email to find out which site is relevant for the version of the software you are using.

Figure 20. Update Site Selection



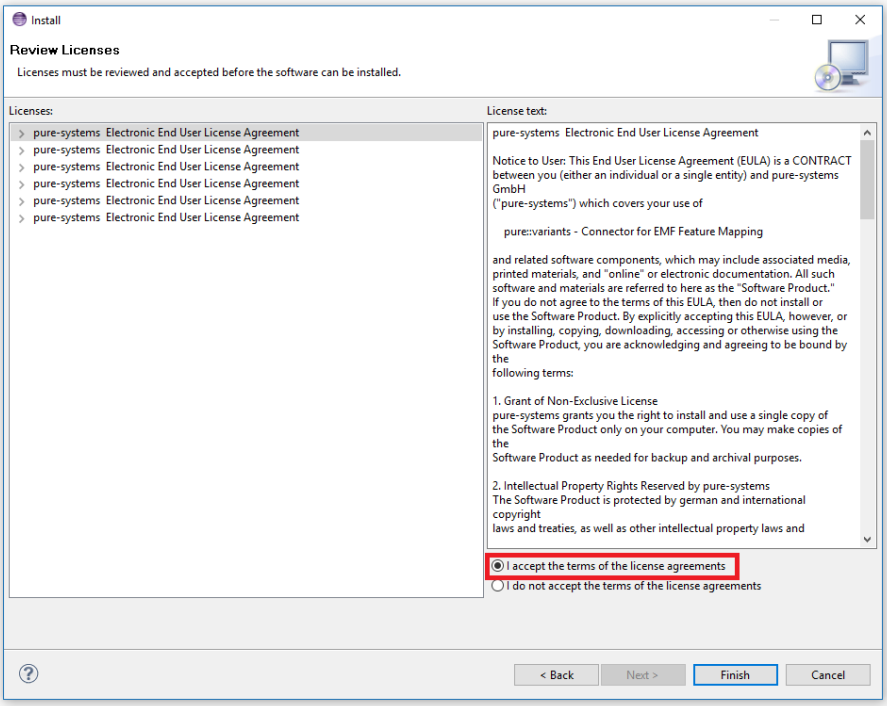
- Unfold the pure::variants update site and select all features to be updated. Select "Next".

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:

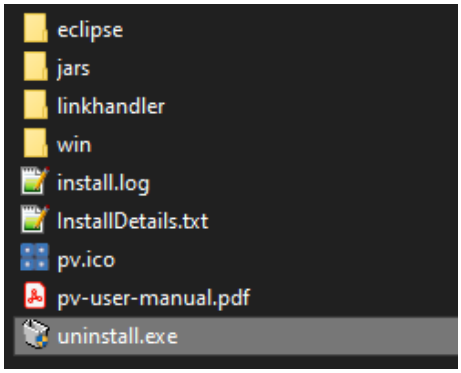
- Start pure::variants (or the Eclipse into which pure::variants has been installed).
- Select "Help"->"Software Updates"->"Find and Install...".
- Select "Search for new features to install" and "Next".
- Click on button "Archived Update Site" or "Local Update Site".
- Use "Browse" to select the downloaded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

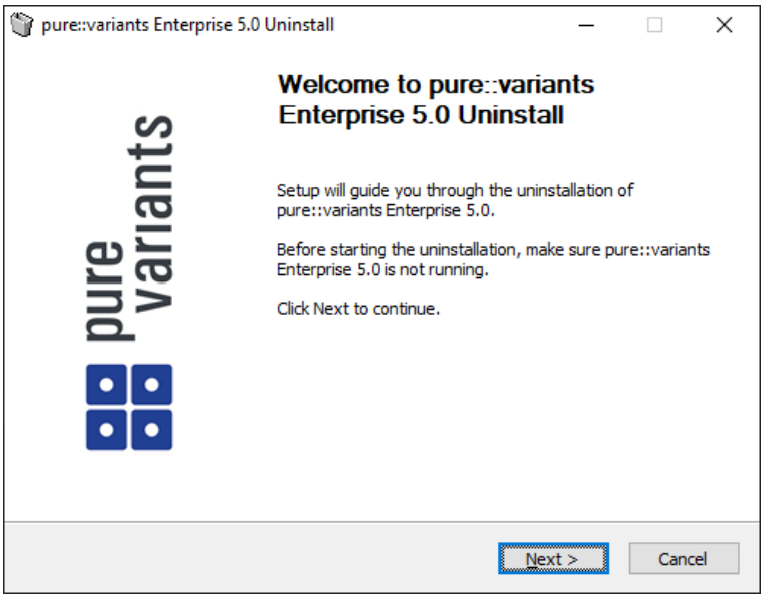
The uninstaller for the pure::variants Desktop Client can be started in two different ways. The first is to go to the Windows *Add or remove programs* application and search for *pure::variants Enterprise* and start the uninstaller by using the *Uninstall* action. The uninstaller requires Administrator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

Figure 23. pure::variants Desktop Client Uninstaller



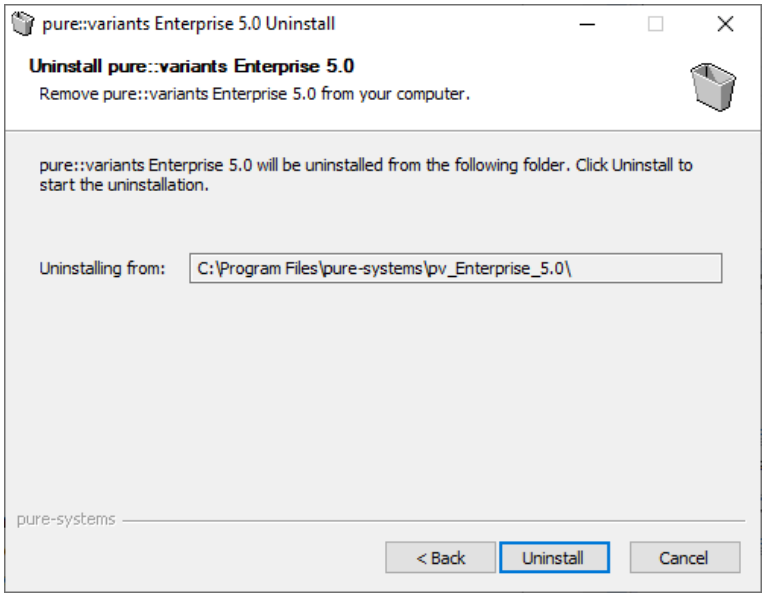
The second way is to navigate to the pure::variants Desktop Client installation folder and start the uninstaller by double clicking it.

Figure 24. pure::variants Desktop Client Uninstaller



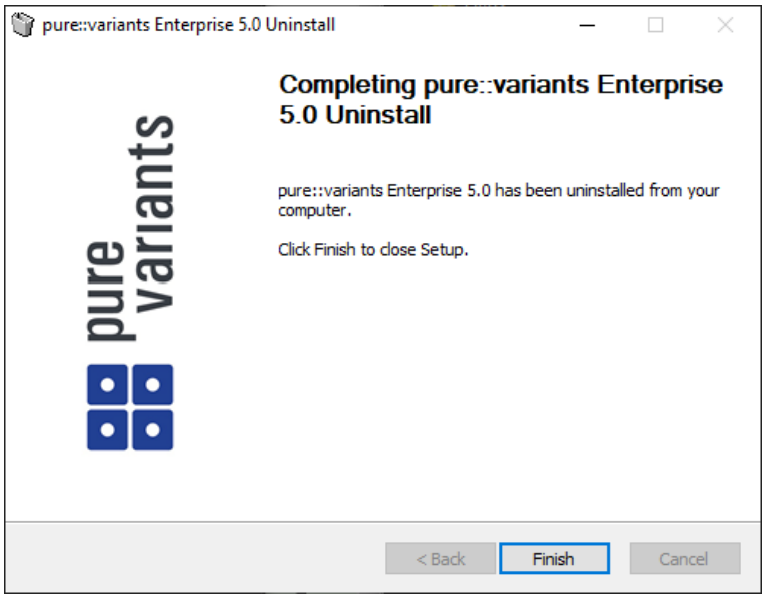
Click *Next*.

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

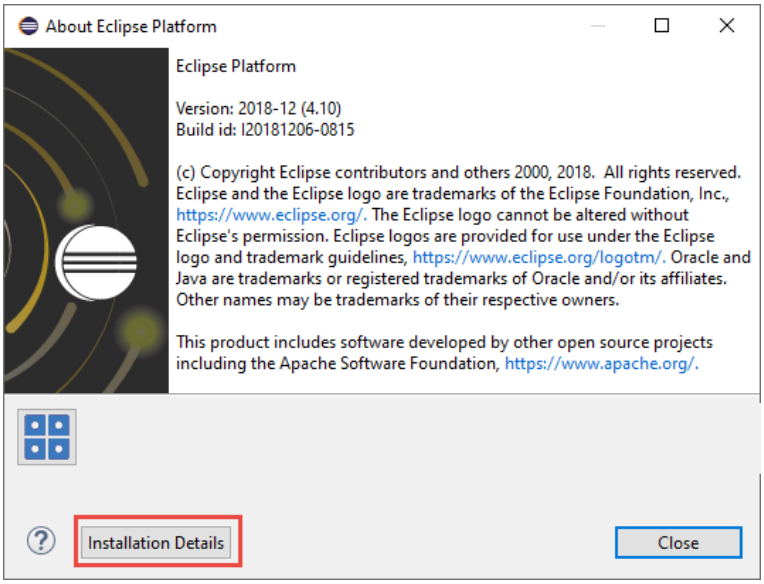
There are two ways to remove pure::variants from an Eclipse instance. You can use the Eclipse command line or remove the pure::variants features one by one in the running Eclipse Instance. Either way a cleanup of the Eclipse instance has to be performed afterwards.

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

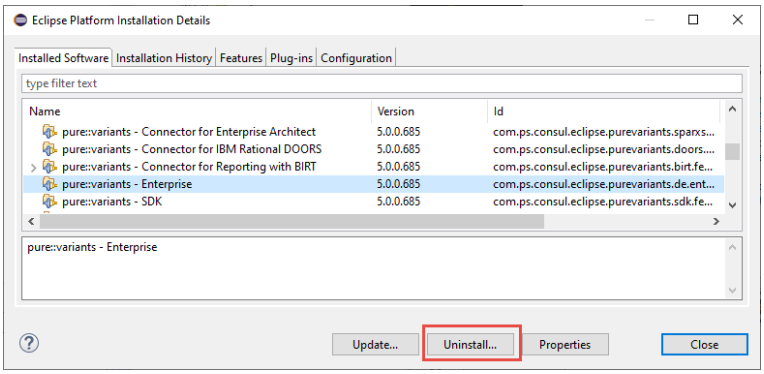
To remove an installed feature from Eclipse using the Eclipse client, open the *About Eclipse Platform* dialog with the *About Eclipse Platform* action in the *Help* menu.

Figure 27. Eclipse About Dialog



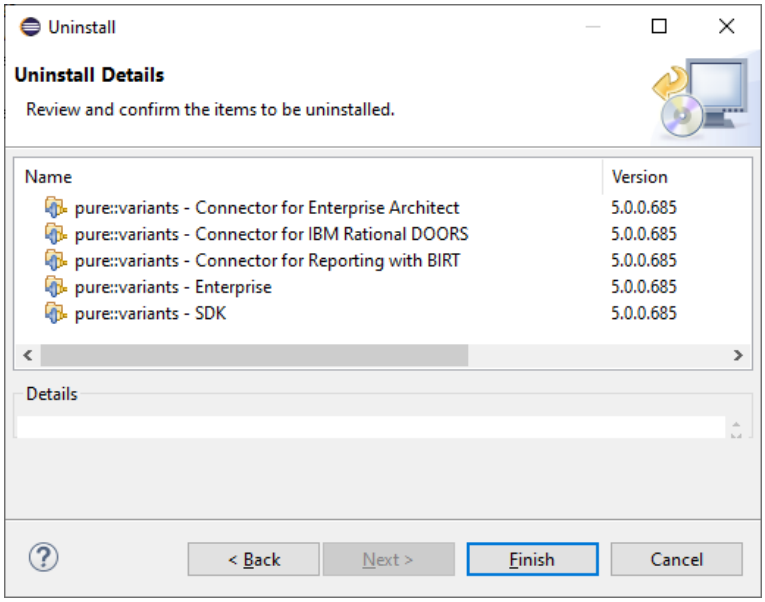
Use the *Installation Details* button to access the installation details.

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

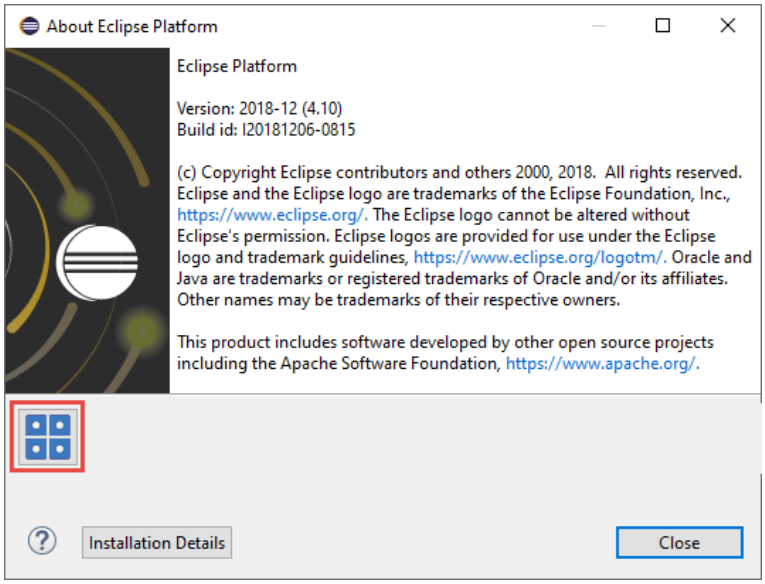


Click *Finish* to start the uninstall process. After it finished, Eclipse will prompt you to restart the application. Click *Restart* to finish the uninstallation.

Uninstall pure::variants using Eclipse uninstall application

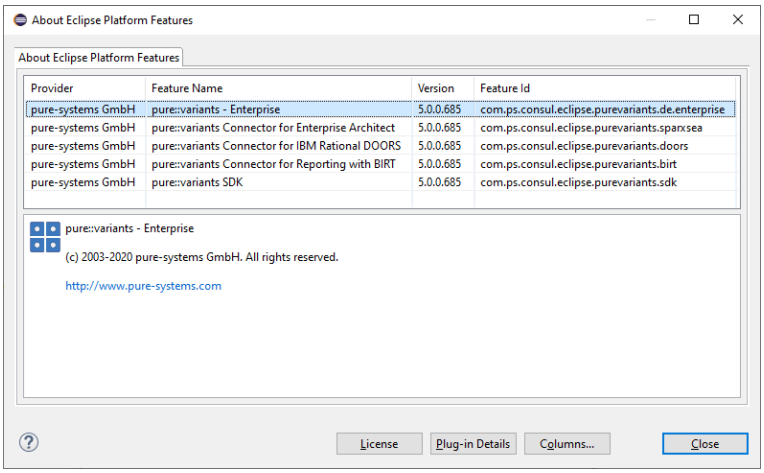
To use the uninstall application you need the feature ids of the features you want to uninstall. The feature ids can be found in the *About Eclipse Plattform* dialog. Open the dialog with the *About Eclipse Platform* action in the *Help* menu.

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
```

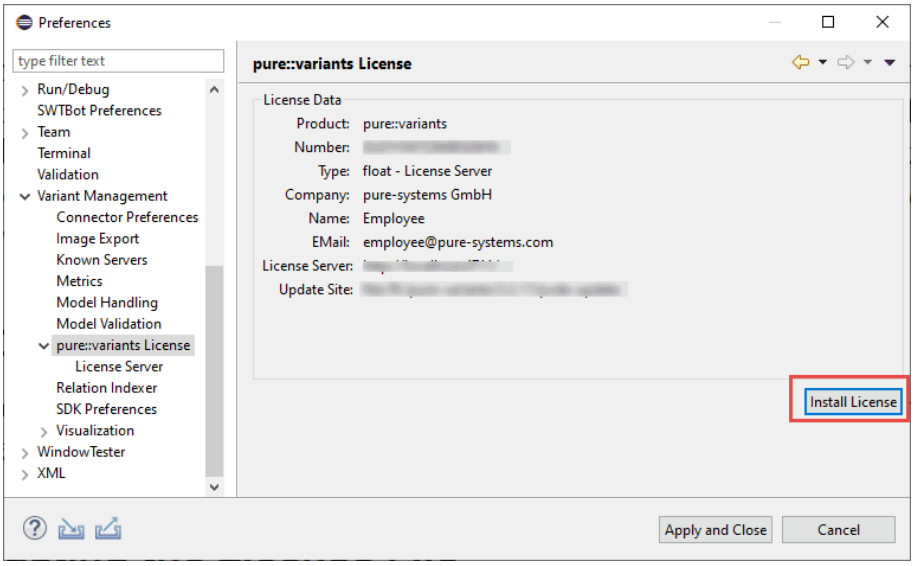
3.4. Basic Setup of the pure::variants Desktop Client

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

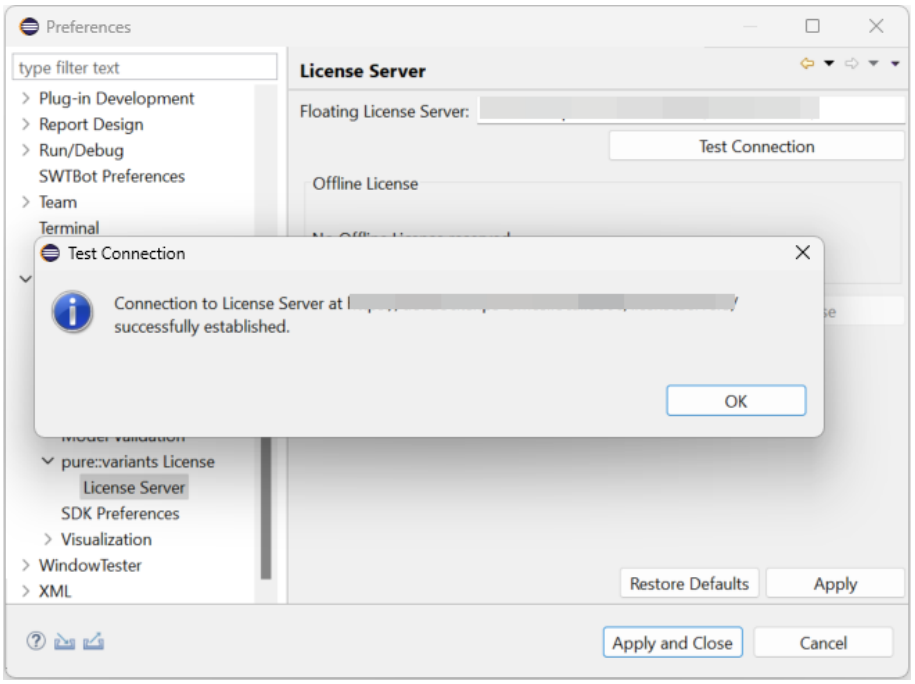


The pure::variants client license can also be defined using the *PVLICENSE* environment variable. This variable has to define the fullpath to a license file. If this variable is defined the given license file is used and the user does not need to define the license for pure::variants client instances.

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.

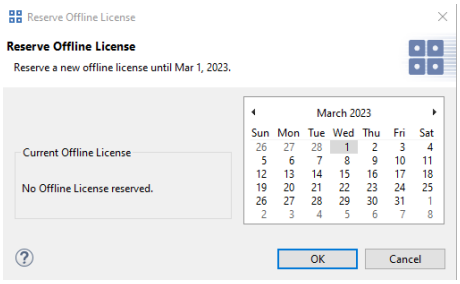
To set the license server URL, open the sub page **Variant Management -> pure::variants License -> License Server** and enter the URL of the license server into the **Floating License Server** text field, click button **Test Connection** to check that the connection is successful.

Figure 33. pure::variants License Server Preferences



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

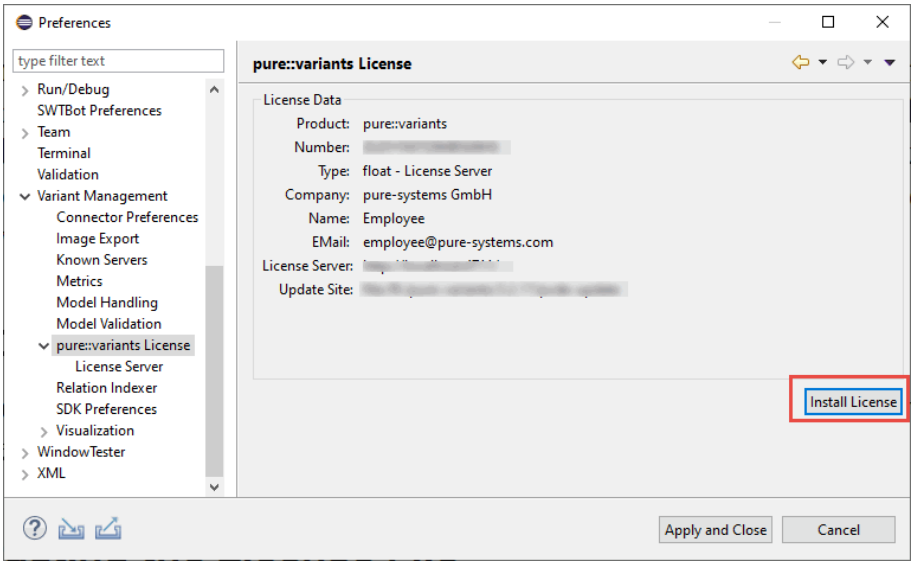
Figure 34. Reserving Offline License



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 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`.

4.1. 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.

4.2. Requirements for the Kubernetes Deployment

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

4.2.1. 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 ???)
- 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.

4.2.2. 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 ???

4.3. 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 ???.

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.

4.4. 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.

4.4.1. 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.

4.4.2. 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 ???

4.4.3. 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.

4.4.4. 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.

4.4.5. 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"}'
```

4.4.6. 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”

4.4.7. 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 ???

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 ???. 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.

4.4.8. 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.

4.5. 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`.

4.6. Validate Correct Operation of Deployment

4.6.1. 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.

4.6.2. 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).