Installation Guide - UnifiedViews 2.X

There are basically three ways how to install UnifiedViews - from Debian packages, using Docker images, or from sources. If possible, installing from Debian packages is the preferred way of installing UnifiedViews. Docker images may contain a bit older versions of UnifiedViews.

General prerequisites

- Recommended RAM memory for the machine - at least 4 GB.
- Installed Java 7+ JDK. (Java 7+ JRE should be enough as well, if you do not plan to e.g. develop DPUs on that machine)
- Browser which supports HTML5 canvas element, see minimum versions of browsers supporting canvas element.
    - So that you can use Unifiedviews UI.
- If you plan to also develop DPUs, you should also have IDE, such as Netbeans or Eclipse or Intellij.

Installing from Debian packages

Supported distribution is **Debian Wheezy** (7.x).

Steps:

- Installing UnifiedViews Core
    - This will install UnifiedViews frontend (.war app running on Tomcat) and UnifiedViews task executor (JAVA console app)
    - You may decide which database (PostgreSQL or MySQL) you would like to use to hold configurations of tasks
- Installing UnifiedViews Core Plugins
    - This will install 35+ Core plugins you may use out of the box

Please see the details for these steps below.

Installing UnifiedViews Core

To install (or upgrade to newer version of UnifiedViews) packages from UnifiedViews Debian repository, please follow these steps:

1. Make sure that the following env. variables - JAVA_HOME, LANGUAGE, LANG, LC_ALL are properly set up (they are not empty), by running the following two commands and checking their outputs:

    ```
    echo JAVA_HOME;
    locale;
    ```

    If not, please export them before installation, e.g.:

    ```
    export JAVA_HOME=/usr/lib/jvm/java-7-oracle
    export LANGUAGE=en_US.UTF-8
    export LANG=en_US.UTF-8
    export LC_ALL=en_US.UTF-8
    ```

2. Add UnifiedViews packages repository into apt-sources-list:

    ```
    echo "deb http://packages.unifiedviews.eu/debian/ wheezy main" > /etc/apt/sources.list.d/unifiedviews.
    list
    ```


    Releases before 2.3.0 are also published at http://packages.comsode.eu/debian/. If you would like to use this alternative location, please use instead of the previous command:

    ```
    echo "deb http://packages.comsode.eu/debian/ wheezy main" > /etc/apt/sources.list.d/odn.list
    ```

3. Add UnifiedViews public key:

    ```
    wget -O - http://packages.unifiedviews.eu/key/unifiedviews.gpg.key | apt-key add -
    ```

> ⚠ Releases before 2.3.0 are also published at http://packages.comsode.eu/debian/. If you would like to use this alternative location, please use instead of the previous command:
>
> ```
> wget -O - http://packages.comsode.eu/key/odn.gpg.key | apt-key add -
> ```

4. Update apt sources:

```
aptitude update
```

5. Use either option a for new installation of UnifiedViews or option b for upgrade to newer version of UnifiedViews:
    a. install UnifiedViews with PostgreSQL as relational database for storing pipeline/DPU configurations:

    ```
    aptitude install unifiedviews-pgsql
    ```

    b. OR install UnifiedViews with mysql as relational database for storing pipeline/DPU configurations:

    ```
    aptitude install unifiedviews-mysql
    ```

    c. OR upgrade UnifiedViews:

    ```
    aptitude upgrade
    ```

    > ⚠ The default location "packages.unifiedviews.eu" contains packages for versions 2.3.0+

    Note: In case of UnifiedViews upgrade, user is required to confirm replacement of configuration files from previous installation

    - it should be confirmed by pressing *y* each time user input is required

6. To verify the installation:
    a. Go to `{domain}:28080/unifiedviews`, where you should be able to log with default credentials `admin/test`
        i. If you are not able to log in, make sure that unifiedviews backend is running - run /etc/init.d/unifiedviews-backend start
7. After successful installation, UnifiedViews Core is prepared, but **it does not contain any Plugins** - DPUs to be used on the pipelines. To install Core plugins, see section below

Installing UnifiedViews Core Plugins

To have all Core plugins in place, please run separately:

```
aptitude install unifiedviews-plugins
```

Maintenance of the Installation from Debian packages

If you need to start/stop/restart either UnifiedViews frontend (Administration interface) or backend, you can use scripts which were prepared for you in `/etc/init.d/` - `unifiedviews-backend, unifiedviews-frontend`. So for example, to restart backend, you may write: `service unifiedviews-backend restart`

Default location for logs:

- For frontend: `/var/log/unifiedviews/frontend`
- For backend: `/var/log/unifiedviews/backend`

Default location for UV configuration files:

- For frontend: `/etc/unifiedviews/frontend-config.properties`
- For backend: `/etc/unifiedviews/backend-config.properties`

Installing using Docker Images

See the images & readme: https://github.com/tenforce/docker-unified-views/tree/master

> ⚠ Docker images may contain a bit older versions of UnifiedViews as they are not created as the newest release of UnifiedViews is available

Installing from sources

When installing from sources, you have to install UnifiedViews - Core. Please note that after installing UnifiedViews - Core, there will be no plugins (DPUs) available - plugins must be installed separately, see below.

Prerequisities:

- Installed Java 7+ JDK. (Java 7+ JRE should be enough as well, if you do not plan to e.g. develop DPUs on that machine)
- Installed Apache Tomcat 7+
    - Installation (from Apache): https://tomcat.apache.org/tomcat-7.0-doc/setup.html
        - You can install multiple instances of Tomcat running as a service
- Installed Mysql for relational database used for storing configuration of pipelines, DPUs, ...
    - As an alternative, PostgreSQL may be also used
- Browser which supports HTML5 canvas element, see minimum versions of browsers supporting canvas element.
    - So that you can use Unifiedviews UI.
- Installed Apache Maven 3.
    - You may skip installation of Git, you can also just copy the relevant files from a different computer (but make sure that code build using Java 7 is not executed by Java 8 RE and vice versa.)
- Installed Git version control tool It may be provided together with certain IDEs
    - You may skip installation of Git, you can also just copy the relevant files from a different computer (but make sure that code build using Java 7 is not executed by Java 8 RE and vice versa.)
- If you plan to also develop DPUs on that machine, you should also have IDE, such as Netbeans or Eclipse or Intellij.

UnifiedViews - Plugins-devEnv + Core

1. Obtain the source code from UV/Plugins-devEnv (latest release can be found at releases page). Read the release notes.
    a. Clean and build the downloaded code by running mvn install inside root folder.

        i. Do not forget to use "mvn install -P java8" if you want to build code for Java8
2. Obtain the source code from UV/Core, latest release can be found at releases page. Read the release notes.
    a. Clean and build the downloaded code by running `mvn install` on the downloaded `Core` folder, further denoted as `{root}`.

        i. Do not forget to use "mvn install -P java8" if you want to build code for Java8
3. Prepare relational database. Use mysql.

    a. log to the mysql database console using `mysql -u{user}`
    b. create new database for UnifiedViews, switch to that database (i.e., type in the mysql console `create database unifiedviews; use unifiedviews;`, which will create new database unifiedviews)

    > ⚠ Starting with UnifiedViews Release 1.3, Virtuoso as relational database is no longer properly supported.
    >
    > For Virtuoso, there are alternative scripts in the virtuoso folder which can be executed by the Virtuoso command utility `isql -U {username} -P {password} -S {isqlPort} < {file}`

    c. For UnifiedViews prior to 2.3. release, it is necessary to create schema/insert initial data, see below. For UnifiedViews 2.3+, schema is automatically created and initial data automatically inserted as the backend is executed for the first time.

For UnifiedViews < 2.3 version

a. import core tables, i.e. type `source {root}/db/mysql/rdbms/schema.sql;`
b. import sample data, i.e. type `source {root}/db/mysql/rdbms/data.sql;`, (in newer versions, `data.sql` is split into `data-core.sql` and `data-permissions.sql`)

4. To run backend - engine executing data processing tasks:
    a. Copy `{root}/backend/conf/config.sample.properties` to the `{root}/backend/target` folder
        i. Create the folder if it does not exist. Take into account that if you rebuild the project, the target folder may be deleted.
        ii. Note: `config.properties` may be copied to an arbitrary location where backend can read files, `{root}/backend/target` folder is used for simplicity
    b. Adjust `{root}/backend/target/config.properties`, a configuration file for backend.
        i. Set at least the following properties:
            1. `general.workingdir` - the directory where the working directory of the executed pipelines is located, i.e., `{root}/backend/working`
                a. Please double-check that the user under which backend/frontend runs has write access to `{root}/backend/working`.
            2. `module.path` - the directory where the data processing units are located, i.e., `{root}/target`
                a. Warning: If you specify different directory then `{root}/target`, you have to copy `{root}/target/lib` folder (which is generated after running `mvn install`) to `{module.path}/lib` folder, so that shared libraries are available to DPUs.
                b. Please double-check that the user under which backend/frontend is running has write access to `{module.path}/dpu`, `{module.path}/lib folders.`
        ii. Specify the connection to the relational database where configuration of pipelines is stored using `database.sql.*` properties
        iii. Specify the connection to RDF store used for intermediate results of the pipelines' executions using `database.rdf.*` properties. There are three choices where intermediate RDF data may be stored: localRDF (openRDF Native repository), remoteRDF (using Sesame, native repositories), and virtuoso (using OpenLink Virtuoso); based on the selected choice, please uncomment the set of related `database.rdf.*` properties.
            1. It is highly recommend to use localRDF store. Support for Virtuoso is experimental!
    c. **Run backend** using `nohup java -DconfigFileLocation={root}/backend/target/config.properties -jar {root}/backend/target/backend-{version}.jar &`
        i. Note: Parameter `-DconfigFileLocation` specifies where the config.properties file is located.
    d. Check write access of backend to certain folders:
        i. Check that backend has write access to `{root}/backend/working, where backend stores data about executions.`
        ii. Check that backend has write access to the directory to which backend is logging (which is specified in `config.properties`).
    e. **Note: Running backend will also create DB schema** (from version 2.3, otherwise it has to be prepared manually, see above)
5. To deploy & run frontend - management GUI of the tool:
    a. Deploy frontend war file -- `{root}/frontend/target/unifiedviews.war –` to the application server (Tomcat), to the `{webapp}` folder.
        i. The web application unifiedviews will not start properly without steb 5b) accomplished.
    b. Copy `{webapp}/unifiedviews/WEB-INF/config.sample.properties` file to `{webapp}/unifiedviews/WEB-INF/config.properties`. Adjust `{webapp}/unifiedviews/WEB-INF/config.properties`. The file `config.properties` is a configuration file for frontend, which contains similar settings as the configuration file for backend. Use the template provided to adjust the configuration, see 4b.
        i. Alternatively, `config.properties` file may be placed to any location you need (similarly as backend). Using this approach, **you can also point both backend and frontend to the same config.properties file.** But in this case, you have to specify `-DconfigFileLocation` within `CATALINA_OPTS` in `{tomcatHome}/bin/setenv.sh`, so that tomcat knows where `config.properties` should be searched. You can also directly edit `{tomcatHome}/bin/catalina.bat/catalina.sh` (i.e., CATALINA_OPTS="$CATALINA_OPTS -DconfigFileLocation=/some/path/to/config.properties -server -Xms1024m -Xmx2048m")
    c. **Start the application unifiedviews**
    d. Check that the application is running under the context `/unifiedviews`. Google Chrome is the preferred browser. The default passwords are `admin/test` for admin account and `user/test` for user account
        i. From UnifiedViews 2.3, before being able to log into the UI, you have to run backend first to create the DB schema with initial data
    e. Check write access of frontend to certain folders:
        i. Check that frontend has write access to `{root}/target/dpu` (needed when frontend imports DPUs)
        ii. If you specified certain log directory to which frontend is logging (in config.propeties of fronted), check that frontend has write access to such folder.

UnifiedViews - Plugins

After finishing the steps above, the installed instance of UnifiedViews still does not contain any plugin (DPU). To install plugins please follow these steps:

1. Obtain the source code from UV/Plugins (latest release can be found at releases page). Read the release notes.
    a. Clean and build the downloaded code by running mvn install inside root folder.
        i. Do not forget to use "mvn install -P java8" if you want to build code for Java8
2. As a result, all DPUs should be built, the resulting JAR files (DPU bundles) are in {dpuName}/target folder.
3. You may then
    a. individually import selected DPUs (JAR files) via GUI of UnifiedViews (DPUTemplate -> Create DPU template") or
    b. pack the generated DPUs inside a zip archive, which may be then imported via GUI (DPUTemplate -> Create DPU template -> zip folder) - when importing zip archive, it basically tries to import all JAR files (DPUs) in the zip archive regardless of the folder in which they are

Advanced installation Options:

Installation (for DPU developers)

1. Obtain the source code from https://github.com/UnifiedViews/Plugin-DevEnv, release or master/develop branch (depends on to which version you want to develop).
2. Clean and build the project by running `mvn clean install.`
3. `Java IDE (Eclipse, Netbeans), New Project, from Maven archetype, uv-dpu-template-base. Define basic metadata for the DPU. As a result, skeleton for the DPU is prepared.`

RDF store - installation requirements.

There are couple of possibilities where working RDF data may be stored:

1. Local RDF (default) - no extra installation needed
2. Remote RDF - Sesame:
    a. Latest version of sesame RDF store (war file) must be installed from http://rdf4j.org/sesame/2.7/docs/users.docbook?view#chapter-server-install>
3. OpenLink Virtuoso (experimental)
    a. Virtuoso has to be installed.

It is recommended to use Local RDF (default) or Remote RDF settings for working RDF database of UnifiedViews.